CSIT REPORT

Release rls2210

Nov 23, 2022

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Report History

FD.io CSIT-2210 Report history and per .[ww] revision changes are listed below.

.[ww] Revision	Changes
.47	1. Added RCA.
.46	1. Added wireguard tests for 3n-icx and 3n-snr. 2. Added NAT44 ED TCP tput tests for 2n-dnv. 3. Added iterative and coverage data.
.45	Initial revision

FD.io CSIT Reports follow CSIT-[yy][mm].[ww] numbering format, with version denoted by concatenation of two digit year [yy] and two digit month [mm], and maintenance revision identified by two digit calendar week number [ww].

1.2 Report Structure

FD.io CSIT-2210 report contains system performance and functional testing data of VPP-22.10 release. PDF version of this report[1] is available for download.

CSIT-2210 report is structured as follows:

1. INTRODUCTION: General introduction to FD.io CSIT-2210.
    - **Introduction**: This section.
    - **Test Scenarios Overview**: A brief overview of test scenarios covered in this report.
    - **Physical Testbeds**: Description of physical testbeds.
    - **Test Methodology**: Performance benchmarking and functional test methodologies.
2. VPP PERFORMANCE: VPP performance tests executed in physical FD.io testbeds.
    - **Overview**: Tested logical topologies, test coverage and naming specifics.
    - **Release Notes**: Changes in CSIT-2210, added tests, environment or methodology changes, known issues.

[1] https://s3-docs.fd.io/csit/rls2210/report/_static/archive/csit_rls2210.47.pdf

- **Packet Throughput**: NDR, PDR throughput graphs based on results from repeated same test job executions to verify repeatibility of measurements.
- **Speedup Multi-Core**: NDR, PDR throughput multi-core speedup graphs based on results from test job executions.
- **Packet Latency**: Latency graphs based on results from test job executions.
- **Soak Tests**: Long duration soak tests are executed using PLRsearch algorithm.
- **NFV Service Density**: Network Function Virtualization (NFV) service density tests focus on measuring total per server throughput at varied NFV service "packing" densities with vswitch providing host dataplane.
- **Comparisons**: Performance comparisons between VPP releases and between different testbed types.
- **Throughput Trending**: References to continuous VPP performance trending.
- **Test Environment**: Performance test environment configuration.
- **Documentation**: Pointers to CSIT source code documentation for VPP performance tests.

3. DPDK PERFORMANCE: DPDK performance tests executed in physical FD.io testbeds.
   - **Overview**: Tested logical topologies, test coverage.
   - **Release Notes**: Changes in CSIT-2210, known issues.
   - **Packet Throughput**: NDR, PDR throughput graphs based on results from repeated same test job executions to verify repeatibility of measurements.
   - **Packet Latency**: Latency graphs based on results from test job executions.
   - **Comparisons**: Performance comparisons between DPDK releases and between different testbed types.
   - **Throughput Trending**: References to regular DPDK performance trending.
   - **Test Environment**: Performance test environment configuration.
   - **Documentation**: Pointers to CSIT source code documentation for DPDK performance tests.
4. TREX PERFORMANCE: TREX performance tests executed in physical FD.io testbeds.
   - **Overview**: Tested logical topologies, test coverage.
   - **Release Notes**: Changes in CSIT-2210, known issues.
   - **Packet Throughput**: NDR, PDR throughput graphs based on results from repeated same test job executions to verify repeatibility of measurements.
   - **Packet Latency**: Latency graphs based on results from test job executions.
   - **Throughput Trending**: References to regular TREX performance trending.
   - **Test Environment**: Performance test environment configuration.
5. VPP DEVICE: VPP functional tests executed in physical FD.io testbeds using containers.
   - **Overview**: Tested virtual topologies, test coverage and naming specifics;
   - **Release Notes**: Changes in CSIT-2210, added tests, environment or methodology changes, known issues.
   - **Integration Tests**: Functional test environment configuration.
   - **Documentation**: Pointers to CSIT source code documentation for VPP functional tests.
6. DETAILED RESULTS: Detailed result tables auto-generated from CSIT test job executions using RF (Robot Framework) output files as sources.
   - **VPP Performance NDR/PDR**: VPP NDR/PDR throughput and latency.

- **VPP Performance MRR**: VPP MRR throughput.
- **DPDK Performance**: DPDK Testpmd and L3fwd NDR/PDR throughput and latency.

7. TEST CONFIGURATION: VPP DUT configuration data based on VPP API Test (VAT) Commands History auto-generated from CSIT test job executions using RF output files as sources.
   - **VPP Performance NDR/PDR**: Configuration data.
   - **VPP Performance MRR**: Configuration data.
8. TEST OPERATIONAL DATA: VPP DUT operational data auto-generated from CSIT test job executions using RFoutput files as sources.
   - **VPP Performance NDR/PDR**: VPP *show run* outputs under test load.
9. CSIT FRAMEWORK DOCUMENTATION: Description of the overall FD.io CSIT framework.
   - **Design**: Framework modular design hierarchy.
   - **Test naming**: Test naming convention.
   - **CSIT RF Tags Descriptions**: CSIT RF Tags used for test suite and test case grouping and selection.

1.3 Test Scenarios

FD.io CSIT-2210 report includes multiple test scenarios of VPP centric applications, topologies and use cases. In addition it also covers baseline tests of DPDK sample applications. Tests are executed in physical (performance tests) and virtual environments (functional tests).

Brief overview of test scenarios covered in this report:

1. **VPP Performance**: VPP performance tests are executed in physical FD.io testbeds, focusing on VPP network data plane performance in NIC-to-NIC switching topologies. VPP application runs in bare-metal host user-mode handling NICs. TRex is used as a traffic generator.
2. **VPP Vhostuser Performance with KVM VMs**: VPP VM service switching performance tests using vhostuser virtual interface for interconnecting multiple NF-in-VM instances. VPP vswitch instance runs in bare-metal user-mode handling NICs and connecting over vhost-user interfaces to VM instances each running VPP with virtio virtual interfaces. Similarly to VPP Performance, tests are run across a range of configurations. TRex is used as a traffic generator.
3. **VPP Memif Performance with LXC and Docker Containers**: VPP Container service switching performance tests using memif virtual interface for interconnecting multiple VPP-in-container instances. VPP vswitch instance runs in bare-metal user-mode handling NICs and connecting over memif (Slave side) interfaces to more instances of VPP running in LXC or in Docker Containers, both with memif interfaces (Master side). Similarly to VPP Performance, tests are run across a range of configurations. TRex is used as a traffic generator.
4. **DPDK Performance**: VPP uses DPDK to drive the NICs and physical interfaces. DPDK performance tests are used as a baseline to profile performance of the DPDK sub-system. Two DPDK applications are tested: Testpmd and L3fwd. DPDK tests are executed in the same testing environment as VPP tests. DPDK Testpmd and L3fwd applications run in host user-mode. TRex is used as a traffic generator.
5. **T-Rex Performance**: T-Rex perfomance tests are executed in physical FD.io testbeds, focusing on T-Rex data plane performance in NIC-to-NIC loopback topologies.
6. **VPP Functional**: VPP functional tests are executed in virtual FD.io testbeds, focusing on VPP packet processing functionality, including both network data plane and in-line control plane. Tests cover vNIC-to-vNIC vNIC-to-nestedVM-to-vNIC forwarding topologies. Scapy is used as a traffic generator.

All CSIT test data included in this report is auto- generated from RF (Robot Framework) `output.xml` files produced by LF (Linux Foundation) FD.io Jenkins jobs executed against VPP-22.10 release artifacts. References are provided to the original FD.io Jenkins job results and all archived source files.

FD.io CSIT system is developed using two main coding platforms: RF and Python. CSIT-2210 source code for the executed test suites is available in CSIT branch rls2210 in the directory `./tests/<name_of_the_test_suite>`. A local copy of CSIT source code can be obtained by cloning CSIT git repository - **`git clone https://gerrit.fd.io/r/csit`**.

1.4 Performance Physical Testbeds

All FD.io (Fast Data Input/Ouput) CSIT (Continuous System Integration and Testing) performance test results included in this report are executed on the physical testbeds hosted by LF FD.io project, unless otherwise noted.

Two physical server topology types are used:

- **2-Node Topology**: Consists of one server acting as a System Under Test (SUT) and one server acting as a Traffic Generator (TG), with both servers connected into a ring topology. Used for executing tests that require frame encapsulations supported by TG.
- **3-Node Topology**: Consists of two servers acting as a Systems Under Test (SUTs) and one server acting as a Traffic Generator (TG), with all servers connected into a ring topology. Used for executing tests that require frame encapsulations not supported by TG e.g. certain overlay tunnel encapsulations and IPsec. Number of native Ethernet, IPv4 and IPv6 encapsulation tests are also executed on these testbeds, for comparison with 2-Node Topology.

Current FD.io production testbeds are built with SUT servers based on the following processor architectures:

- Intel Xeon: Cascadelake 6252N, Icelake 8358.
- Intel Atom: Denverton C3858, Snowridge P5362.
- Arm: TaiShan 2280, hip07-d05, Neoverse N1.
- AMD EPYC: Zen2 7532.

Server SUT performance depends on server and processor type, hence results for testbeds based on different servers must be reported separately, and compared if appropriate.

Complete technical specifications of compute servers used in CSIT physical testbeds are maintained in FD.io CSIT repository: https://git.fd.io/csit/tree/docs/lab/testbed_specifications.md.

1.4.1 Physical NICs and Drivers

SUT and TG servers are equipped with a number of different NIC models.

VPP is performance tested on SUTs with the following NICs and drivers:

1. 2p10GE: x550, x553 Intel (codename Niantic) - DPDK Poll Mode Driver (PMD).
2. 4p10GE: x710-DA4 Intel (codename Fortville, FVL) - DPDK PMD. - AVF in PMD mode. - AF_XDP in PMD mode.
3. 2p25GE: xxv710-DA2 Intel (codename Fortville, FVL) - DPDK PMD. - AVF in PMD mode. - AF_XDP in PMD mode.
4. 4p25GE: xxv710-DA4 Intel (codename Fortville, FVL) - DPDK PMD. - AVF in PMD mode. - AF_XDP in PMD mode.
5. 4p25GE: E822-CQDA4 Intel (codename Columbiaville, CVL) - DPDK PMD. - AVF in PMD mode.
6. 2p100GE: cx556a-edat Mellanox ConnectX5 - RDMA_core in PMD mode.

7. 2p100GE: E810-2CQDA2 Intel (codename Columbiaville, CVL) - DPDK PMD. - AVF in PMD mode.

DPDK applications, testpmd and l3fwd, are performance tested on the same SUTs exclusively with DPDK drivers for all NICs.

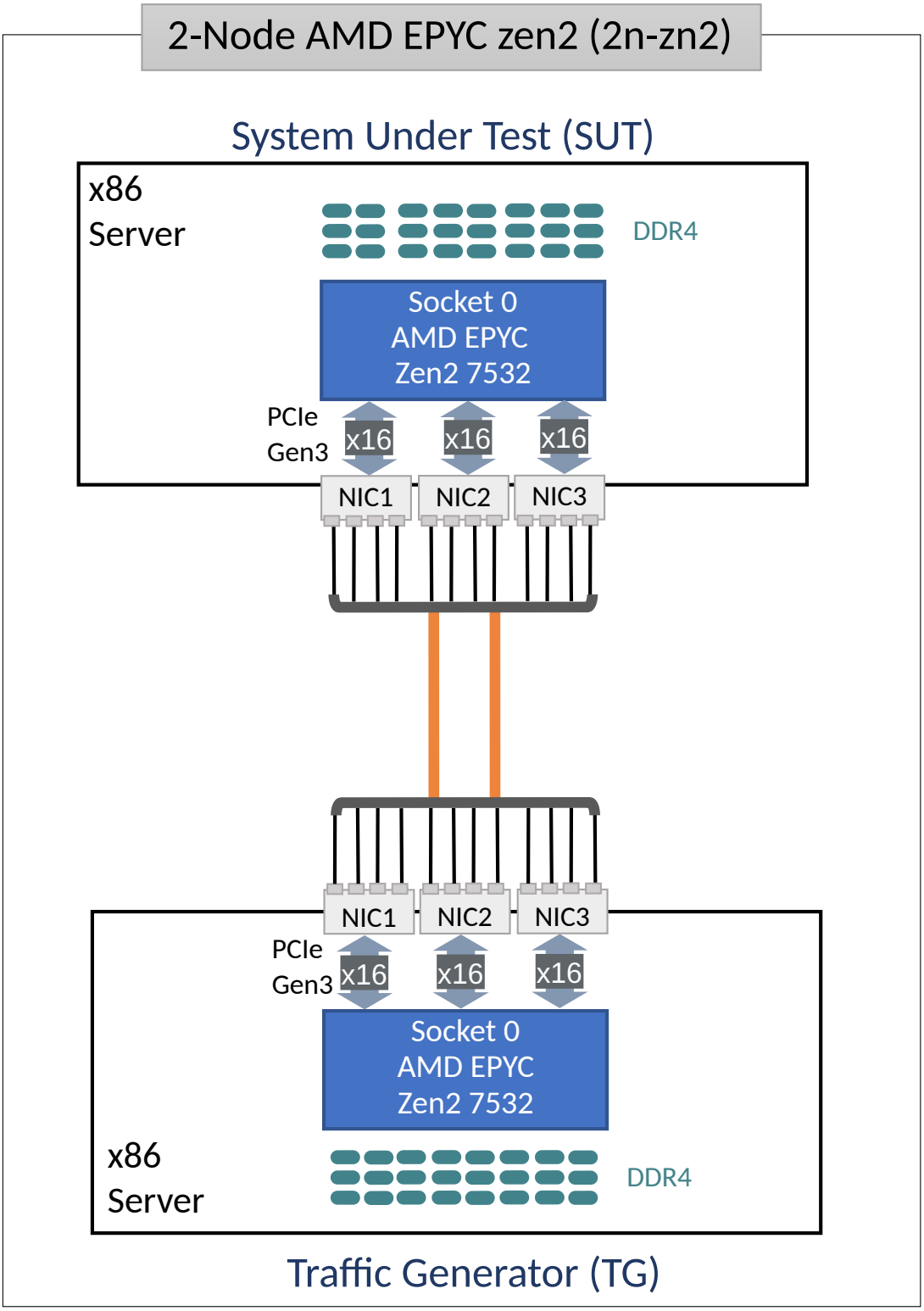
TRex running on TGs is using DPDK drivers for all NICs.

VPP hoststack tests utilize ab (Apache HTTP server benchmarking tool) running on TGs and using Linux drivers for all NICs.

For more information see *Test Environment* (page 1206) and *Test Environment* (page 1354).

1.4.2 2-Node AMD EPYC Zen2 (2n-zn2)

One 2n-zn2 testbed in in operation in FD.io labs. It is built based on two SuperMicro SuperMicro AS-1114S-WTRT servers, with SUT and TG servers equipped with one AMD EPYC Zen2 7532 processor each (256 MB Cache, 2.40 GHz, 32 cores). 2n-zn2 physical topology is shown below.



SUT NICs:

1. NIC-1: x710-DA4 4p10GE Intel.
2. NIC-2: xxv710-DA2 2p25GE Intel.
3. NIC-3: cx556a-edat ConnectX5 2p100GE Mellanox.

TG NICs:

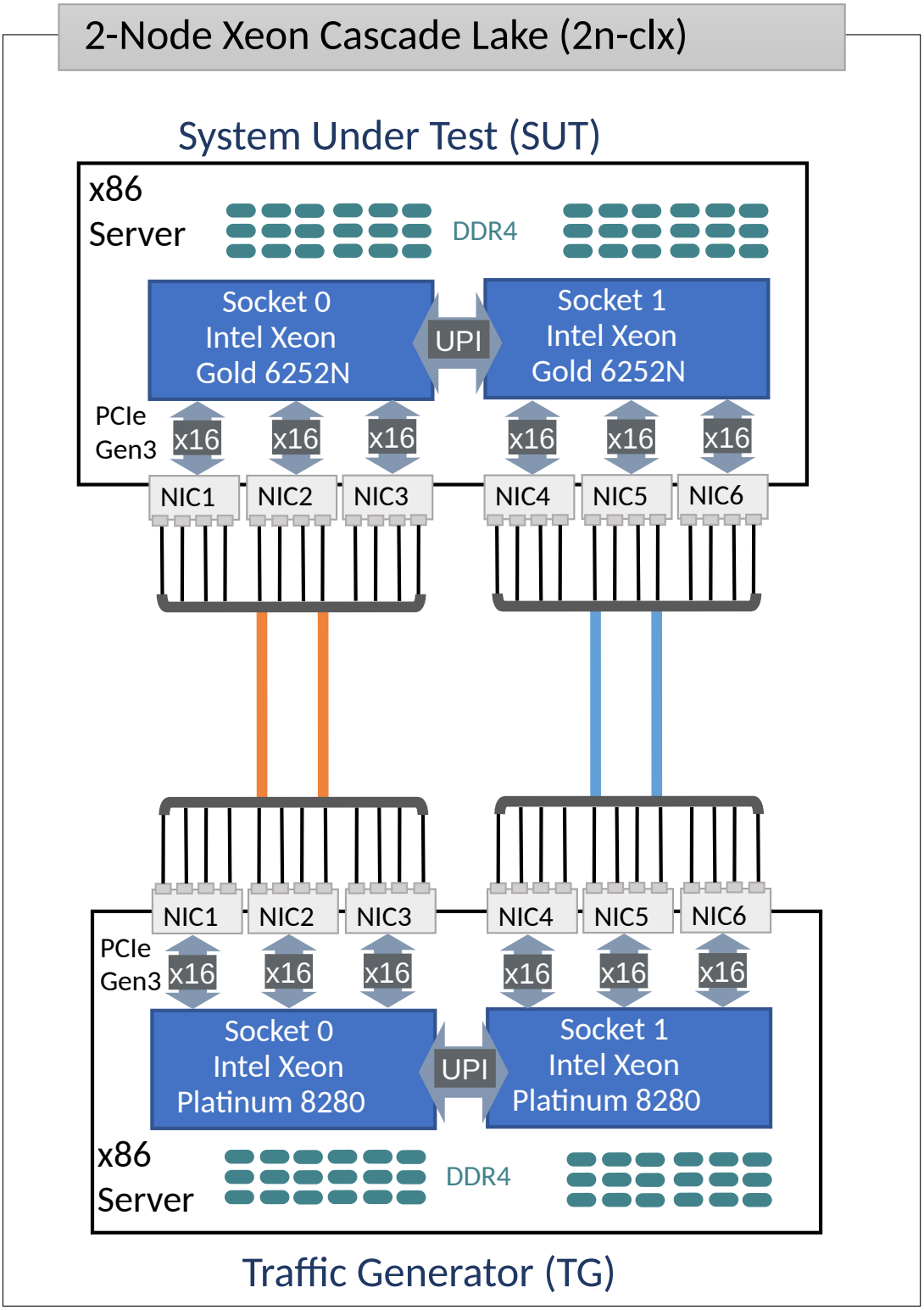
1. NIC-1: x710-DA4 4p10GE Intel.

2. NIC-2: xxv710-DA2 2p25GE Intel.
3. NIC-3: cx556a-edat ConnectX5 2p100GE Mellanox.

All AMD EPYC Zen2 7532 servers run with AMD SMT enabled, doubling the number of logical cores exposed to Linux.

1.4.3 2-Node Xeon Cascadelake (2n-clx)

Three 2n-clx testbeds are in operation in FD.io labs. Each 2n-clx testbed is built with two SuperMicro SYS-7049GP-TRT servers, SUTs are equipped with two Intel Xeon Gold 6252N processors (35.75 MB Cache, 2.30 GHz, 24 cores). TGs are equiped with Intel Xeon Cascade Lake Platinum 8280 processors (38.5 MB Cache, 2.70 GHz, 28 cores). 2n-clx physical topology is shown below.



SUT NICs:

1. NIC-1: x710-DA4 4p10GE Intel.
2. NIC-2: xxv710-DA2 2p25GE Intel.
3. NIC-3: cx556a-edat ConnectX5 2p100GE Mellanox.
4. NIC-4: empty, future expansion.
5. NIC-5: empty, future expansion.

6. NIC-6: empty, future expansion.

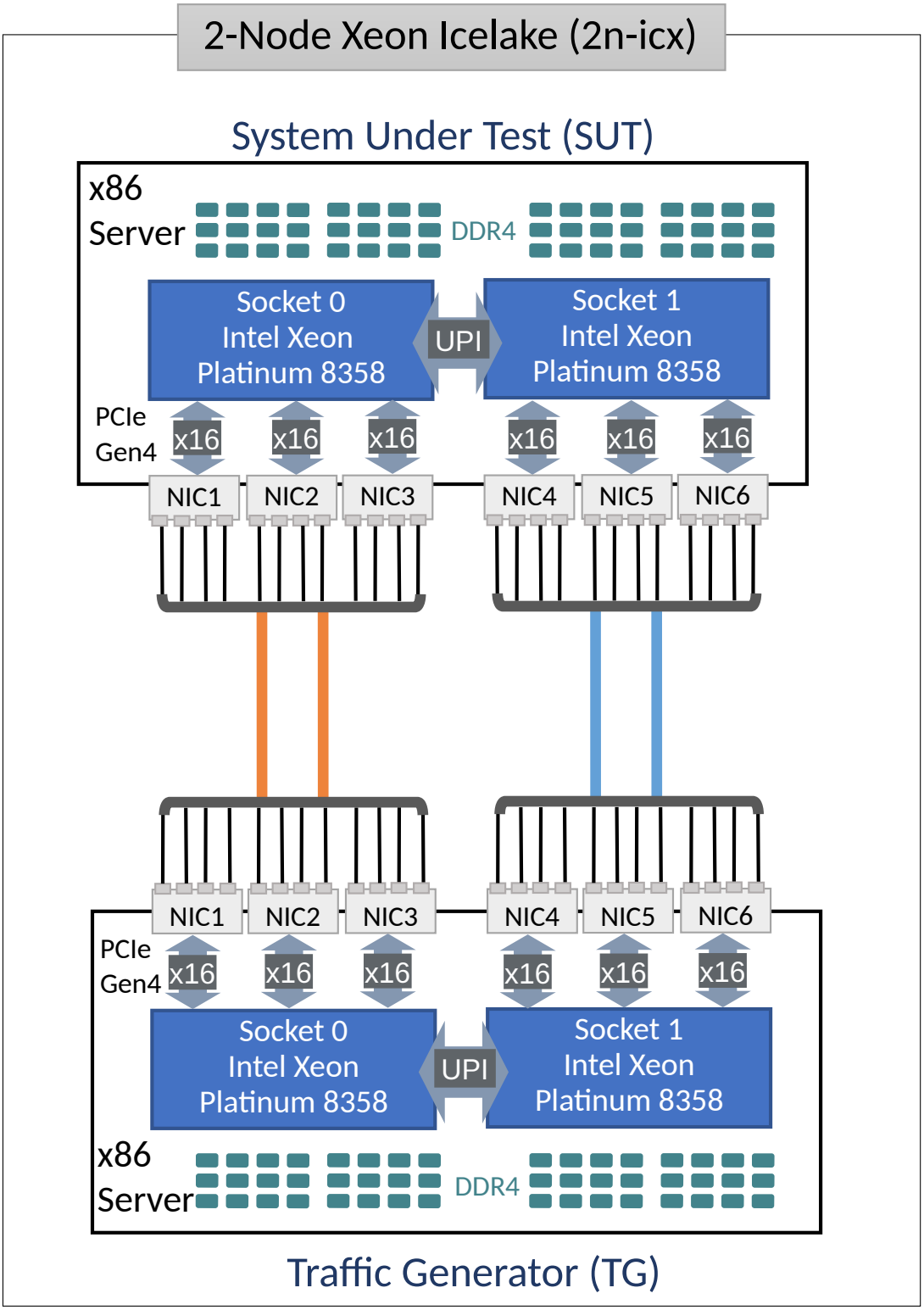
TG NICs:

1. NIC-1: x710-DA4 4p10GE Intel.
2. NIC-2: xxv710-DA2 2p25GE Intel.
3. NIC-3: cx556a-edat ConnectX5 2p100GE Mellanox.
4. NIC-4: empty, future expansion.
5. NIC-5: empty, future expansion.
6. NIC-6: x710-DA4 4p10GE Intel. (For self-tests.)

All Intel Xeon Cascadelake servers run with Intel Hyper-Threading enabled, doubling the number of logical cores exposed to Linux.

1.4.4 2-Node Xeon Icelake (2n-icx)

One 2n-icx testbed is in operation in FD.io labs. It is built with two SuperMicro SYS-740GP-TNRT servers, each in turn equipped with two Intel Xeon Platinum 8358 processors (48 MB Cache, 2.60 GHz, 32 cores).



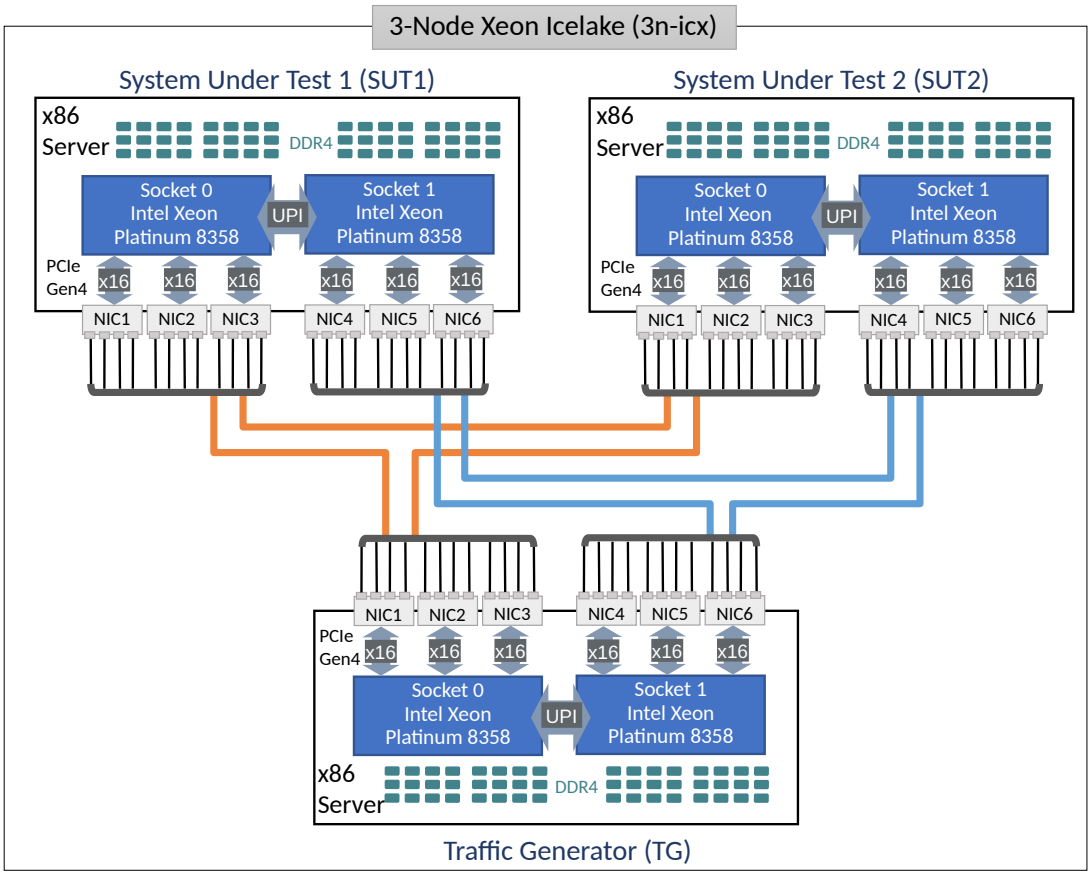
SUT and TG NICs:

1. NIC-1: xxv710-DA2 2p25GE Intel.
2. NIC-2: E810-2CQDA2 2p100GbE Intel (* to be added).
3. NIC-3: E810-CQDA4 4p100GbE Intel (* to be added).

All Intel Xeon Icelake servers run with Intel Hyper-Threading enabled, doubling the number of logical cores exposed to Linux.

1.4.5 3-Node Xeon Icelake (3n-icx)

One 3n-icx testbed is in operation in FD.io labs. It is built with three SuperMicro SYS-740GP-TNRT servers, each in turn equipped with two Intel Xeon Platinum 8358 processors (48 MB Cache, 2.60 GHz, 32 cores).



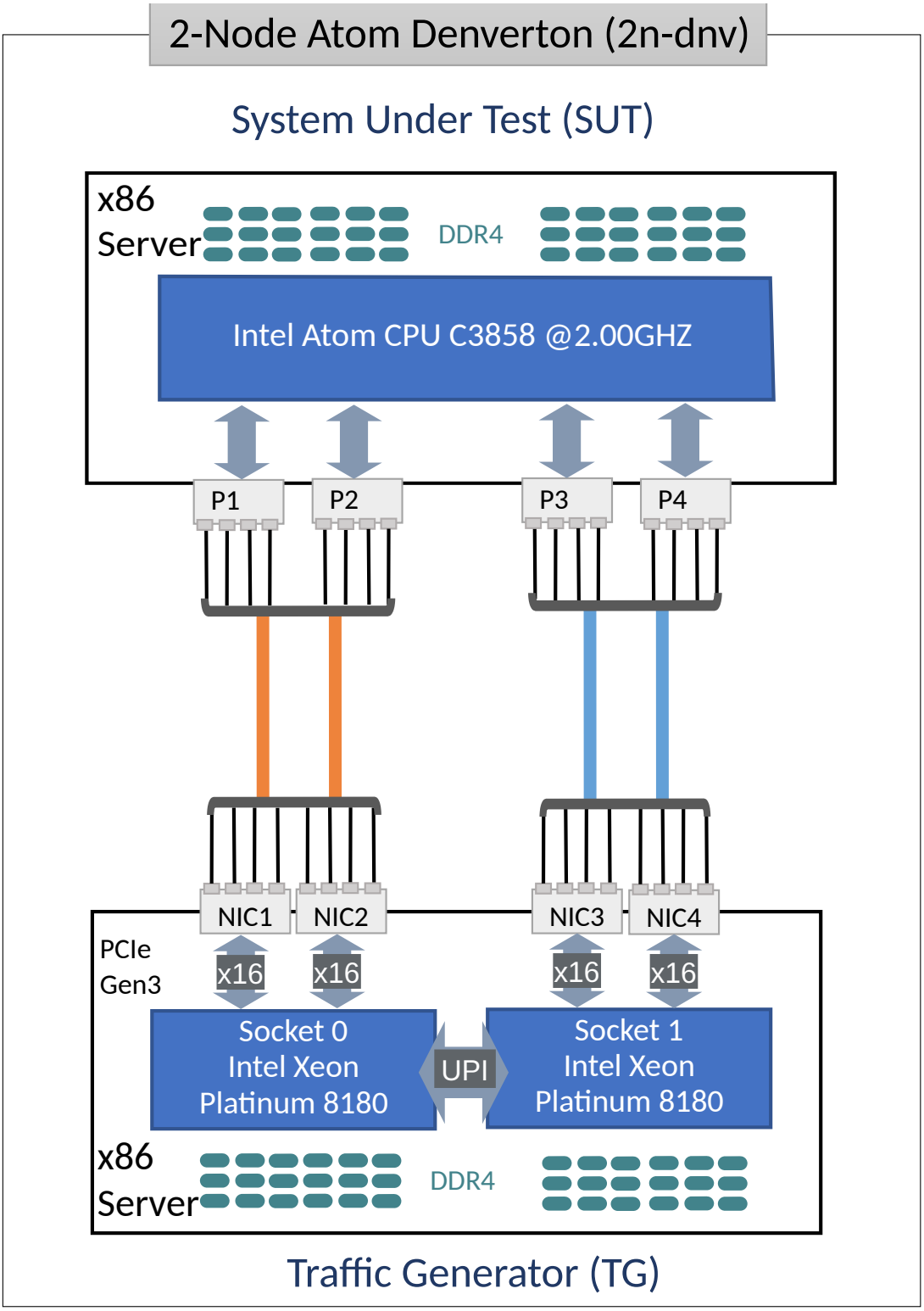
SUT and TG NICs:

1. NIC-1: xxv710-DA2 2p25GE Intel.
2. NIC-2: E810-2CQDA2 2p100GbE Intel (* to be added).
3. NIC-3: E810-CQDA4 4p100GbE Intel (* to be added).

All Intel Xeon Icelake servers run with Intel Hyper-Threading enabled, doubling the number of logical cores exposed to Linux.

1.4.6 2-Node Atom Denverton (2n-dnv)

2n-dnv testbed is built with: i) one Intel S2600WFT server acting as TG and equipped with two Intel Xeon Skylake Platinum 8180 processors (38.5 MB Cache, 2.50 GHz, 28 cores), and ii) one SuperMicro SYS-E300-9A server acting as SUT and equipped with one Intel Atom C3858 processor (12 MB Cache, 2.00 GHz, 12 cores). 2n-dnv physical topology is shown below.



SUT 10GE NIC ports:

1. P-1: x553 copper port.
2. P-2: x553 copper port.
3. P-3: x553 fiber port.
4. P-4: x553 fiber port.

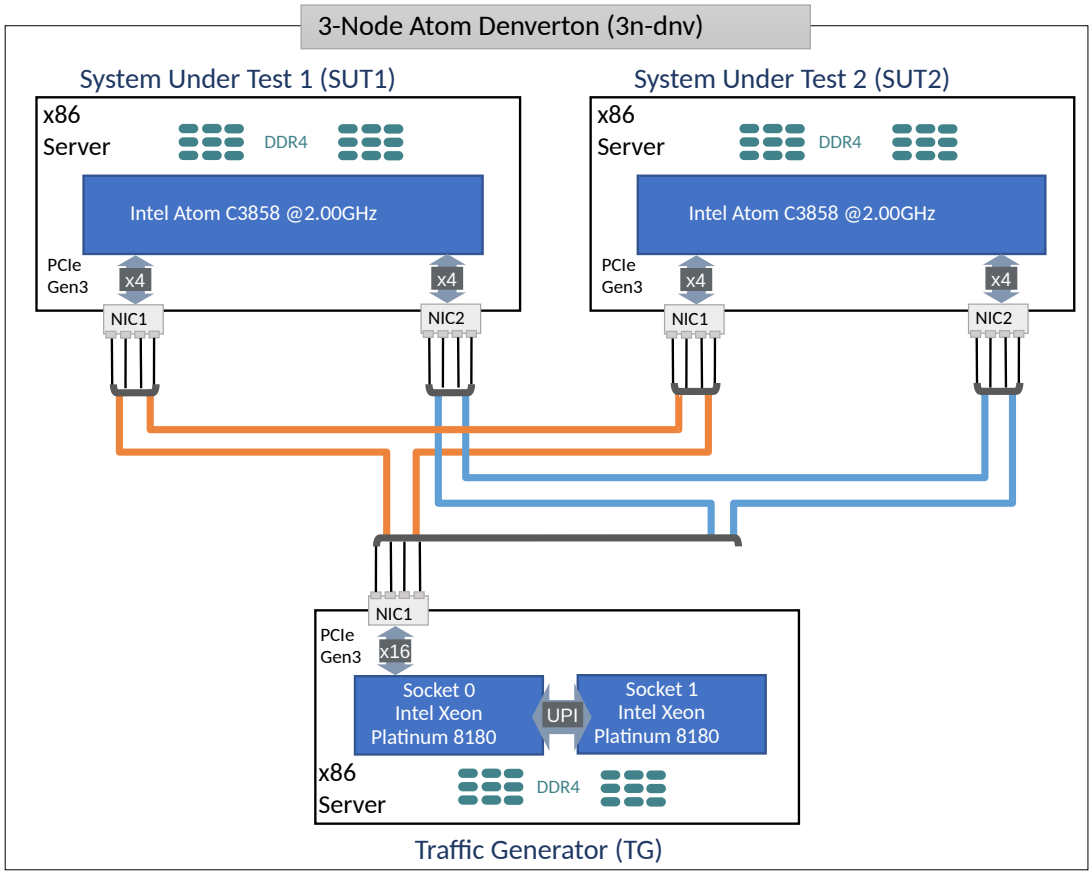
TG NICs:

1. NIC-1: x550-T2 2p10GE Intel.
2. NIC-2: x550-T2 2p10GE Intel.
3. NIC-3: x520-DA2 2p10GE Intel.
4. NIC-4: x520-DA2 2p10GE Intel.

The 2n-dnv testbed is in operation in Intel SH labs.

1.4.7 3-Node Atom Denverton (3n-dnv)

One 3n-dnv testbed is built with: i) one SuperMicro SYS-7049GP-TRT server acting as TG and equipped with two Intel Xeon Skylake Platinum 8180 processors (38.5 MB Cache, 2.50 GHz, 28 cores), and ii) one SuperMicro SYS-E300-9A server acting as SUT and equipped with one Intel Atom C3858 processor (12 MB Cache, 2.00 GHz, 12 cores). 3n-dnv physical topology is shown below.



SUT1 and SUT2 NICs:

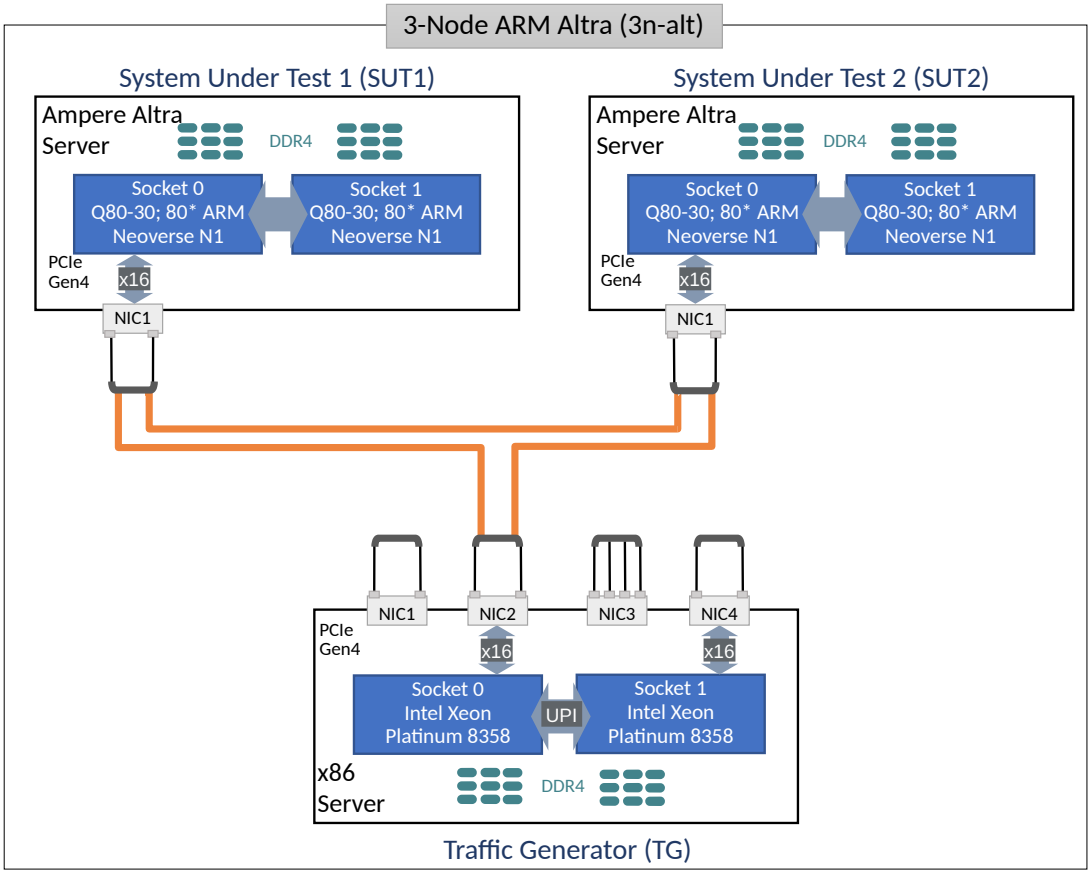
1. NIC-1: x553 2p10GE fiber Intel.
2. NIC-2: x553 2p10GE copper Intel.

TG NICs:

1. NIC-1: x710-DA4 4p10GE Intel.

1.4.8 3-Node ARM Altra (3n-alt)

One 3n-tsh testbed is built with: i) one SuperMicro SYS-740GP-TNRT server acting as TG and equipped with two Intel Xeon Icelake Platinum 8358 processors (80 MB Cache, 2.60 GHz, 32 cores), and ii) one Ampere Altra server acting as SUT and equipped with two Q80-30 processors (80* ARM Neoverse N1). 3n-alt physical topology is shown below.



SUT1 and SUT2 NICs:

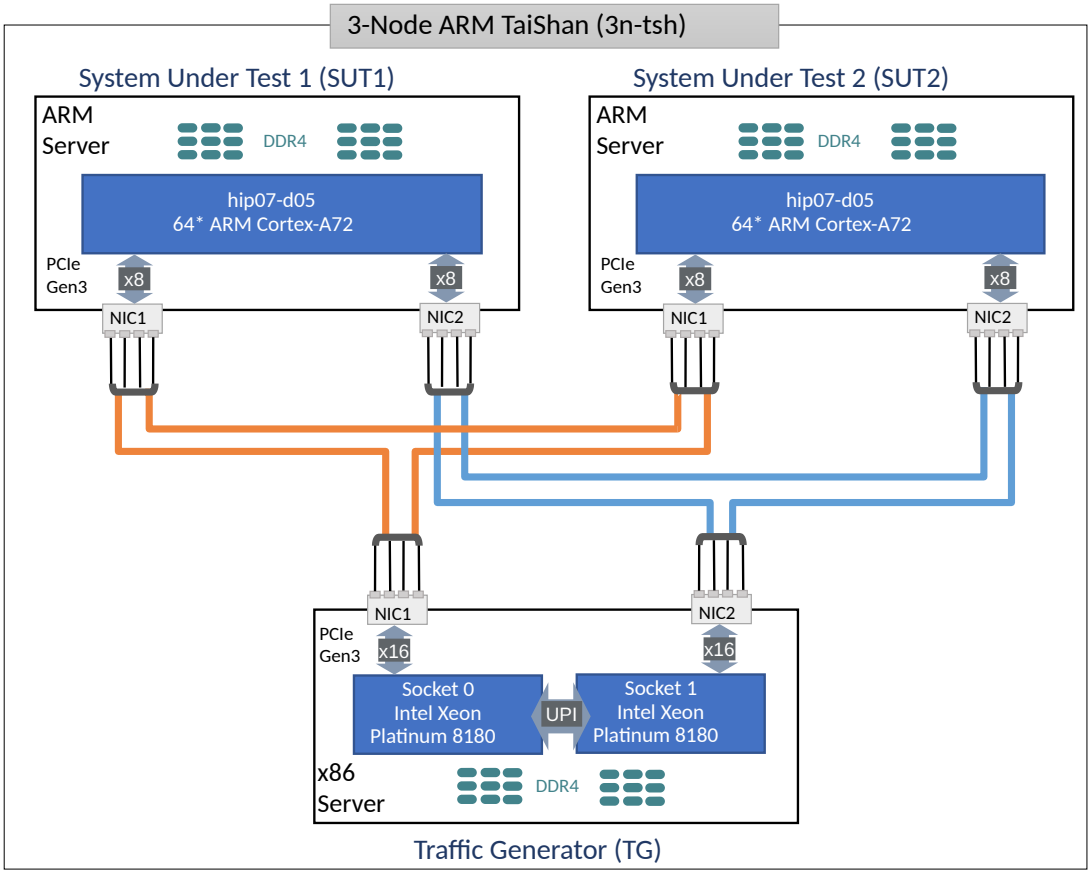
1. NIC-1: xl710-QDA2-2p40GE Intel.

TG NICs:

1. NIC-1: xxv710-DA2-2p25GE Intel.
2. NIC-2: xl710-QDA2-2p40GE Intel.
3. NIC-3: e810-XXVDA4-4p25GE Intel.
4. NIC-4: e810-2CQDA2-2p100GE Intel.

1.4.9 3-Node ARM TaiShan (3n-tsh)

One 3n-tsh testbed is built with: i) one SuperMicro SYS-7049GP-TRT server acting as TG and equipped with two Intel Xeon Skylake Platinum 8180 processors (38.5 MB Cache, 2.50 GHz, 28 cores), and ii) one Huawei TaiShan 2280 server acting as SUT and equipped with one hip07-d05 processor (64* ARM Cortex-A72). 3n-tsh physical topology is shown below.



SUT1 and SUT2 NICs:

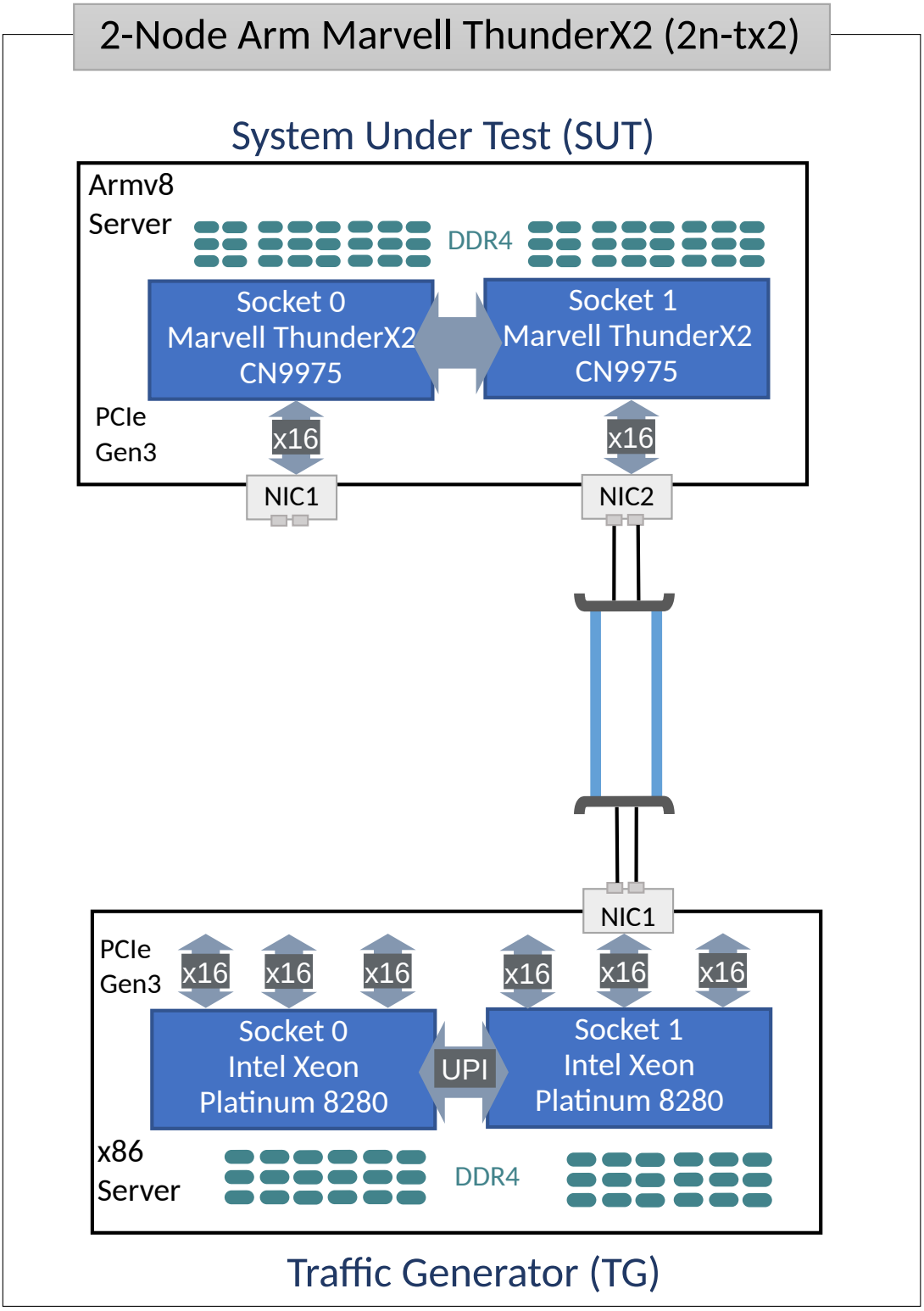
1. NIC-1: connectx4 2p25GE Mellanox.
2. NIC-2: x520 2p10GE Intel.

TG NICs:

1. NIC-1: x710-DA4 4p10GE Intel.
2. NIC-2: xxv710-DA2 2p25GE Intel.
3. NIC-3: xl710-QDA2 2p40GE Intel.

1.4.10 2-Node ARM ThunderX2 (2n-tx2)

One 2n-tx2 testbed is built with: i) one SuperMicro SYS-7049GP-TRT server acting as TG and equipped with two Intel Xeon Skylake Platinum 8180 processors (38.5 MB Cache, 2.50 GHz, 28 cores), and ii) one Marvell ThnderX2 9975 (28* ThunderX2) server acting as SUT and equipped with two ThunderX2 ARMv8 CN9975 processors. 2n-tx2 physical topology is shown below.



SUT NICs:

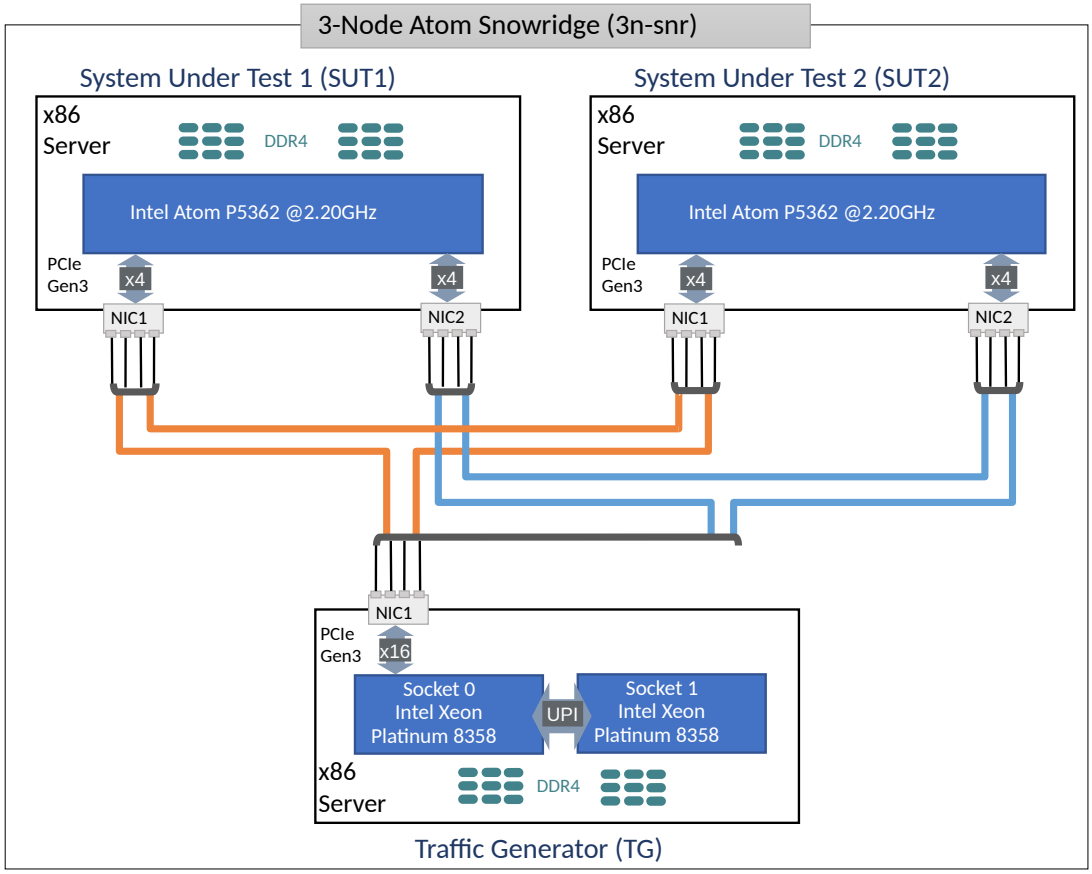
1. NIC-1: xl710-QDA2 2p40GE Intel (not connected).
2. NIC-2: xl710-QDA2 2p40GE Intel.

TG NICs:

1. NIC-1: xl710-QDA2 2p40GE Intel.

1.4.11 3-Node Atom Snowridge (3n-snr)

One 3n-snr testbed is built with: i) one SuperMicro SYS-740GP-TNRT server acting as TG and equipped with two Intel Xeon Icelake Platinum 8358 processors (48 MB Cache, 2.60 GHz, 32 cores), and ii) SUT equipped with one Intel Atom P5362 processor (27 MB Cache, 2.20 GHz, 24 cores). 3n-snr physical topology is shown below.



SUT1 and SUT2 NICs:

1. NIC-1: e822cq-DA4 4p25GE fiber Intel.

TG NICs:

1. NIC-1: e810xxv-DA4 4p25GE Intel.

1.5 Test Methodology

1.5.1 Terminology

- **Frame size**: size of an Ethernet Layer-2 frame on the wire, including any VLAN tags (dot1q, dot1ad) and Ethernet FCS, but excluding Ethernet preamble and inter-frame gap. Measured in Bytes.
- **Packet size**: same as frame size, both terms used interchangeably.
- **Inner L2 size**: for tunneled L2 frames only, size of an encapsulated Ethernet Layer-2 frame, preceded with tunnel header, and followed by tunnel trailer. Measured in Bytes.
- **Inner IP size**: for tunneled IP packets only, size of an encapsulated IPv4 or IPv6 packet, preceded with tunnel header, and followed by tunnel trailer. Measured in Bytes.
- **Device Under Test (DUT)**: In software networking, "device" denotes a specific piece of software tasked with packet processing. Such device is surrounded with other software components (such as operating system kernel). It is not possible to run devices without also running the other components, and hardware resources are shared between both. For purposes of testing, the whole set of hardware and software components is called "System Under Test" (SUT). As SUT is the part of the whole test setup performance of which can be measured with RFC 2544[2], using SUT instead of RFC 2544[3] DUT. Device under test (DUT) can be re-introduced when analyzing test results using whitebox techniques, but this document sticks to blackbox testing.
- **System Under Test (SUT)**: System under test (SUT) is a part of the whole test setup whose performance is to be benchmarked. The complete methodology contains other parts, whose performance is either already established, or not affecting the benchmarking result.
- **Bi-directional throughput tests**: involve packets/frames flowing in both east-west and west-east directions over every tested interface of SUT/DUT. Packet flow metrics are measured per direction, and can be reported as aggregate for both directions (i.e. throughput) and/or separately for each measured direction (i.e. latency). In most cases bi-directional tests use the same (symmetric) load in both directions.
- **Uni-directional throughput tests**: involve packets/frames flowing in only one direction, i.e. either east-west or west-east direction, over every tested interface of SUT/DUT. Packet flow metrics are measured and are reported for measured direction.
- **Packet Loss Ratio (PLR)**: ratio of packets received relative to packets transmitted over the test trial duration, calculated using formula: PLR = (pkts_transmitted - pkts_received) / pkts_transmitted. For bi-directional throughput tests aggregate PLR is calculated based on the aggregate number of packets transmitted and received.
- **Packet Throughput Rate**: maximum packet offered load DUT/SUT forwards within the specified Packet Loss Ratio (PLR). In many cases the rate depends on the frame size processed by DUT/SUT. Hence packet throughput rate MUST be quoted with specific frame size as received by DUT/SUT during the measurement. For bi-directional tests, packet throughput rate should be reported as aggregate for both directions. Measured in packets-per-second (pps) or frames-per-second (fps), equivalent metrics.
- **Bandwidth Throughput Rate**: a secondary metric calculated from packet throughput rate using formula: bw_rate = pkt_rate * (frame_size + L1_overhead) * 8, where L1_overhead for Ethernet includes preamble (8 Bytes) and inter-frame gap (12 Bytes). For bi-directional tests, bandwidth throughput rate should be reported as aggregate for both directions. Expressed in bits-per-second (bps).
- **Non Drop Rate (NDR)**: maximum packet/bandwith throughput rate sustained by DUT/SUT at PLR equal zero (zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet NDR measured in packets-per-second (or fps), bandwidth NDR expressed in bits-per-second (bps).

[2] https://datatracker.ietf.org/doc/html/rfc2544.html
[3] https://datatracker.ietf.org/doc/html/rfc2544.html

- **Partial Drop Rate (PDR)**: maximum packet/bandwith throughput rate sustained by DUT/SUT at PLR greater than zero (non-zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet PDR measured in packets-per-second (or fps), bandwidth PDR expressed in bits-per-second (bps).
- **Maximum Receive Rate (MRR)**: packet/bandwidth rate regardless of PLR sustained by DUT/SUT under specified Maximum Transmit Rate (MTR) packet load offered by traffic generator. MUST be quoted with both specific packet size and MTR as received by DUT/SUT during the measurement. Packet MRR measured in packets-per-second (or fps), bandwidth MRR expressed in bits-per-second (bps).
- **Trial**: a single measurement step.
- **Trial duration**: amount of time over which packets are transmitted and received in a single measurement step.

1.5.2 Per Thread Resources

CSIT test framework is managing mapping of the following resources per thread:

1. Cores, physical cores (pcores) allocated as pairs of sibling logical cores (lcores) if server in Hyper-Threading/SMT mode, or as single lcores if server not in HyperThreading/SMT mode. Note that if server's processors are running in HyperThreading/SMT mode sibling lcores are always used.
2. Receive Queues (RxQ), packet receive queues allocated on each physical and logical interface tested.
3. Transmit Queues(TxQ), packet transmit queues allocated on each physical and logical interface tested.

Approach to mapping per thread resources depends on the application/DUT tested (VPP or DPDK apps) and associated thread types, as follows:

1. Data-plane workers, used for data-plane packet processing, when no feature workers present.
   - Cores: data-plane workers are typically tested in 1, 2 and 4 pcore configurations, running on single lcore per pcore or on sibling lcores per pcore. Result is a set of {T}t{C}c thread-core configurations, where{T} stands for a total number of threads (lcores), and {C} for a total number of pcores. Tested configurations are encoded in CSIT test case names, e.g. "1c", "2c", "4c", and test tags "2T1C"(or "1T1C"), "4T2C" (or "2T2C"), "8T4C" (or "4T4C").
   - Interface Receive Queues (RxQ): as of CSIT-2106 release, number of RxQs used on each physical or virtual interface is equal to the number of data-plane workers. In other words each worker has a dedicated RxQ on each interface tested. This ensures packet processing load to be equal for each worker, subject to RSS flow load balancing efficacy. Note: Before CSIT-2106 total number of RxQs across all interfaces of specific type was equal to the number of data-plane workers.
   - Interface Transmit Queues (TxQ): number of TxQs used on each physical or virtual interface is equal to the number of data-plane workers. In other words each worker has a dedicated TxQ on each interface tested.
   - Applies to VPP and DPDK Testpmd and L3Fwd.
2. Data-plane and feature workers (e.g. IPsec async crypto workers), the latter dedicated to specific feature processing.
   - Cores: data-plane and feature workers are tested in 2, 3 and 4 pcore configurations, running on single lcore per pcore or on sibling lcores per pcore. This results in a two sets of thread-core combinations separated by "-", {T}t{C}c-{T}t{C}c, with the leading set denoting total number of threads (lcores) and pcores used for data-plane workers, and the trailing set denoting total number of lcores and pcores used for feature workers. Accordingly, tested configurations are encoded in CSIT test case names, e.g. "1c-1c", "1c-2c", "1c-3c", and test tags "2T1C_2T1C" (or "1T1C_1T1C"), "2T1C_4T2C"(or "1T1C_2T2C"), "2T1C_6T3C" (or "1T1C_3T3C").
   - RxQ and TxQ: no RxQs and no TxQs are used by feature workers.

- Applies to VPP only.

3. Management/main worker, control plane and management.
   - Cores: single lcore.
   - RxQ: not used (VPP default behaviour).
   - TxQ: single TxQ per interface, allocated but not used (VPP default behaviour).
   - Applies to VPP only.

VPP Thread Configuration

Mapping of cores and RxQs to VPP data-plane worker threads is done in the VPP startup.conf during test suite setup:

1. *corelist-workers <list_of_cores>*: List of logical cores to run VPP data-plane workers and feature workers. The actual lcores' allocations depends on HyperThreading/SMT server configuration and per test core configuration.
   - For tests without feature workers, by default, all CPU cores configured in startup.conf are used for data-plane workers.
   - For tests with feature workers, CSIT code distributes lcores across data-plane and feature workers.
2. *num-rx-queues <value>*: Number of Rx queues used per interface.

Mapping of TxQs to VPP data-plane worker threads uses the default VPP setting of one TxQ per interface per data-plane worker.

DPDK Thread Configuration

Mapping of cores and RxQs to DPDK Testpmd/L3Fwd data-plane worker threads is done in the startup CLI:

1. *-l <list_of_cores>* - List of logical cores to run DPDK application.
2. *nb-cores=<N>* - Number of forwarding cores.
3. *rxq=<N>* - Number of Rx queues used per interface.

1.5.3 VPP Forwarding Modes

VPP is tested in a number of L2, IPv4 and IPv6 packet lookup and forwarding modes. Within each mode baseline and scale tests are executed, the latter with varying number of FIB entries.

L2 Ethernet Switching

VPP is tested in three L2 forwarding modes:

- *l2patch*: L2 patch, the fastest point-to-point L2 path that loops packets between two interfaces without any Ethernet frame checks or lookups.
- *l2xc*: L2 cross-connect, point-to-point L2 path with all Ethernet frame checks, but no MAC learning and no MAC lookup.
- *l2bd*: L2 bridge-domain, multipoint-to-multipoint L2 path with all Ethernet frame checks, with MAC learning (unless static MACs are used) and MAC lookup.

l2bd tests are executed in baseline and scale configurations:

- *l2bdbase*: Two MAC FIB entries are learned by VPP to enable packet switching between two interfaces in two directions. VPP L2 switching is tested with 254 IPv4 unique flows per direction, varying IPv4 source address per flow in order to invoke RSS based packet distribution across VPP workers. The same source and destination MAC address is used for all flows per direction. IPv4 source address is incremented for every packet.
- *l2bdscale*: A high number of MAC FIB entries are learned by VPP to enable packet switching between two interfaces in two directions. Tested MAC FIB sizes include: i) 10k with 5k unique flows per direction, ii) 100k with 2 x 50k flows and iii) 1M with 2 x 500k flows. Unique flows are created by using distinct source and destination MAC addresses that are changed for every packet using incremental ordering, making VPP learn (or refresh) distinct src MAC entries and look up distinct dst MAC entries for every packet. For details, see *Packet Flow Ordering* (page 44).

Ethernet wire encapsulations tested include: untagged, dot1q, dot1ad.

IPv4 Routing

IPv4 routing tests are executed in baseline and scale configurations:

- *ip4base*: Two /32 IPv4 FIB entries are configured in VPP to enable packet routing between two interfaces in two directions. VPP routing is tested with 253 IPv4 unique flows per direction, varying IPv4 source address per flow in order to invoke RSS based packet distribution across VPP workers. IPv4 source address is incremented for every packet.
- *ip4scale*: A high number of /32 IPv4 FIB entries are configured in VPP. Tested IPv4 FIB sizes include: i) 20k with 10k unique flows per direction, ii) 200k with 2 * 100k flows and iii) 2M with 2 * 1M flows. Unique flows are created by using distinct IPv4 destination addresses that are changed for every packet, using incremental or random ordering. For details, see *Packet Flow Ordering* (page 44).

IPv6 Routing

Similarly to IPv4, IPv6 routing tests are executed in baseline and scale configurations:

- *ip6base*: Two /128 IPv4 FIB entries are configured in VPP to enable packet routing between two interfaces in two directions. VPP routing is tested with 253 IPv6 unique flows per direction, varying IPv6 source address per flow in order to invoke RSS based packet distribution across VPP workers. IPv6 source address is incremented for every packet.
- *ip4scale*: A high number of /128 IPv6 FIB entries are configured in VPP. Tested IPv6 FIB sizes include: i) 20k with 10k unique flows per direction, ii) 200k with 2 * 100k flows and iii) 2M with 2 * 1M flows. Unique flows are created by using distinct IPv6 destination addresses that are changed for every packet, using incremental or random ordering. For details, see *Packet Flow Ordering* (page 44).

SRv6 Routing

SRv6 routing tests are executed in a number of baseline configurations, in each case SR policy and steering policy are configured for one direction and one (or two) SR behaviours (functions) in the other directions:

- *srv6enc1sid*: One SID (no SRH present), one SR function - End.
- *srv6enc2sids*: Two SIDs (SRH present), two SR functions - End and End.DX6.
- *srv6enc2sids-nodecaps*: Two SIDs (SRH present) without decapsulation, one SR function - End.
- *srv6proxy-dyn*: Dynamic SRv6 proxy, one SR function - End.AD.
- *srv6proxy-masq*: Masquerading SRv6 proxy, one SR function - End.AM.
- *srv6proxy-stat*: Static SRv6 proxy, one SR function - End.AS.

In all listed cases low number of IPv6 flows (253 per direction) is routed by VPP.

1.5.4 Data Plane Throughput

Data Plane Throughput Tests

Network data plane throughput is measured using multiple test methods in order to obtain representative and repeatable results across the large set of performance test cases implemented and executed within CSIT.

Following throughput test methods are used:

- MLRsearch - Multiple Loss Ratio search
- MRR - Maximum Receive Rate
- PLRsearch - Probabilistic Loss Ratio search

Description of each test method is followed by generic test properties shared by all methods.

MLRsearch Tests

Description

Multiple Loss Ratio search (MLRsearch) tests discover multiple packet throughput rates in a single search, reducing the overall test execution time compared to a binary search. Each rate is associated with a distinct Packet Loss Ratio (PLR) criteria. In FD.io CSIT two throughput rates are discovered: Non-Drop Rate (NDR, with zero packet loss, PLR=0) and Partial Drop Rate (PDR, with PLR<0.5%). MLRsearch is compliant with **RFC 2544**[4].

Usage

MLRsearch tests are run to discover NDR and PDR rates for each VPP and DPDK release covered by CSIT report. Results for small frame sizes (64b/78B, IMIX) are presented in packet throughput graphs (Box-and-Whisker Plots) with NDR and PDR rates plotted against the test cases covering popular VPP packet paths.

Each test is executed at least 10 times to verify measurements repeatability and results are compared between releases and test environments. NDR and PDR packet and bandwidth throughput results for all frame sizes and for all tests are presented in detailed results tables.

Details

See *MLRsearch Tests* (page 24) section for more detail. MLRsearch is being standardized in IETF in draft-ietf-bmwg-mlrsearch[5].

MRR Tests

Description

Maximum Receive Rate (MRR) tests are complementary to MLRsearch tests, as they provide a maximum "raw" throughput benchmark for development and testing community.

MRR tests measure the packet forwarding rate under the maximum load offered by traffic generator (dependent on link type and NIC model) over a set trial duration, regardless of packet loss. Maximum load for specified Ethernet frame size is set to the bi-directional link rate.

[4] https://datatracker.ietf.org/doc/html/rfc2544.html
[5] https://datatracker.ietf.org/doc/html/draft-ietf-bmwg-mlrsearch-01

Usage

MRR tests are much faster than MLRsearch as they rely on a single trial or a small set of trials with very short duration. It is this property that makes them suitable for continuous execution in daily performance trending jobs enabling detection of performance anomalies (regressions, progressions) resulting from data plane code changes.

MRR tests are also used for VPP per patch performance jobs verifying patch performance vs parent. CSIT reports include MRR throughput comparisons between releases and test environments. Small frame sizes only (64b/78B, IMIX).

Details

See *MRR Throughput* (page 25) section for more detail about MRR tests configuration.

FD.io CSIT performance dashboard includes complete description of daily performance trending tests[6] and VPP per patch tests[7].

PLRsearch Tests

Description

Probabilistic Loss Ratio search (PLRsearch) tests discovers a packet throughput rate associated with configured Packet Loss Ratio (PLR) criteria for tests run over an extended period of time a.k.a. soak testing. PLRsearch assumes that system under test is probabilistic in nature, and not deterministic.

Usage

PLRsearch are run to discover a sustained throughput for PLR=10^-7 (close to NDR) for VPP release covered by CSIT report. Results for small frame sizes (64b/78B) are presented in packet throughput graphs (Box Plots) for a small subset of baseline tests.

Each soak test lasts 30 minutes and is executed at least twice. Results are compared against NDR and PDR rates discovered with MLRsearch.

Details

See *PLRsearch* (page 26) methodology section for more detail. PLRsearch is being standardized in IETF in draft-vpolak-bmwg-plrsearch[8].

Generic Test Properties

All data plane throughput test methodologies share following generic properties:

- Tested L2 frame sizes (untagged Ethernet):
  - IPv4 payload: 64B, IMIX (28x64B, 16x570B, 4x1518B), 1518B, 9000B.
  - IPv6 payload: 78B, IMIX (28x78B, 16x570B, 4x1518B), 1518B, 9000B.
  - All quoted sizes include frame CRC, but exclude per frame transmission overhead of 20B (preamble, inter frame gap).

[6] https://s3-docs.fd.io/csit/master/trending/methodology/performance_tests.html
[7] https://s3-docs.fd.io/csit/master/trending/methodology/perpatch_performance_tests.html
[8] https://tools.ietf.org/html/draft-vpolak-bmwg-plrsearch

- Offered packet load is always bi-directional and symmetric.
- All measured and reported packet and bandwidth rates are aggregate bi-directional rates reported from external Traffic Generator perspective.

MLRsearch Tests

Overview

Multiple Loss Ratio search (MLRsearch) tests use an optimized search algorithm implemented in FD.io CSIT project. MLRsearch discovers any number of loss ratio loads in a single search.

Two loss ratio goals are of interest in FD.io CSIT, leading to Non-Drop Rate (NDR, loss ratio goal is exact zero) and Partial Drop Rate (PDR, non-zero loss ratio goal, currently 0.5%).

MLRsearch discovers all the loads in a single pass, reducing required time duration compared to separate `binary search`_es for each rate. Overall search time is reduced even further by relying on shorter trial durations of intermediate steps, with only the final measurements conducted at the specified final trial duration. This results in the shorter overall execution time when compared to standard NDR/PDR binary search, while guaranteeing similar results.

Note: All throughput rates are *always* bi-directional aggregates of two equal (symmetric) uni-directional packet rates received and reported by an external traffic generator, unless the test specifically requires unidirectional traffic.

Search Implementation

Detailed description of the MLRsearch algorithm is included in the IETF draft draft-ietf-bmwg-mlrsearch-02[9] that is in the process of being standardized in the IETF Benchmarking Methodology Working Group (BMWG). (Newer version is published in IETF, describing improvements not yet used in CSIT production.)

MLRsearch is also available as a PyPI (Python Package Index) library[10].

Algorithm highlights

MRR and receive rate at MRR load are used as initial guesses for the search.

All previously measured trials (except the very first one which can act as a warm-up) are taken into consideration, unless superseded by a trial at the same load but higher duration.

For every loss ratio goal, tightest upper and lower bound (from results of large enough trial duration) form an interval. Exit condition is given by that interval reaching low enough relative width. Small enough width is achieved by bisecting the current interval. The bisection can be uneven, to save measurements based on information theory.

Switching to higher trial duration generally requires a re-measure at a load from previous trial duration. When the re-measurement does not confirm previous bound classification (e.g. tightest lower bound at shorter trial duration becomes a newest tightest upper bound upon re-measurement), external search is used to find close enough bound of the lost type. External search is a generalization of the first stage of exponential search[11].

Shorter trial durations use double width goal, because one bisection is always safe before risking external search.

[9] https://datatracker.ietf.org/doc/html/draft-ietf-bmwg-mlrsearch-02
[10] https://pypi.org/project/MLRsearch/
[11] https://en.wikipedia.org/wiki/Exponential_search

Within an iteration for a specific trial duration, smaller loss ratios (NDR) are narrowed down first before search continues with higher loss ratios (PDR).

Other heuristics are there, aimed to prevent unneccessarily narrow intervals, and to handle corner cases around min and max load.

Deviations from RFC 2544

CSIT does not have any explicit wait times before and after trial traffic.

Small differences between intended and offered load are tolerated, mainly due to various time overheads preventing precise measurement of the traffic duration (and TRex can sometimes suffer from duration stretching).

The final trial duration is only 30s (10s for reconf tests).

MRR Throughput

Maximum Receive Rate (MRR) tests are complementary to MLRsearch tests, as they provide a maximum "raw" throughput benchmark for development and testing community. MRR tests measure the packet forwarding rate under the maximum load offered by traffic generator over a set trial duration, regardless of packet loss.

MRR tests are currently used for following test jobs:

- Report performance comparison: 64B, IMIX for vhost, memif.
- Daily performance trending: 64B, IMIX for vhost, memif.
- Per-patch performance verification: 64B.
- Initial iterations of MLRsearch and PLRsearch: 64B.

Maximum offered load for specific L2 Ethernet frame size is set to either the maximum bi-directional link rate or tested NIC model capacity, as follows:

- For 10GE NICs the maximum packet rate load is 2x14.88 Mpps for 64B, a 10GE bi-directional link rate.
- For 25GE NICs the maximum packet rate load is 2x18.75 Mpps for 64B, a 25GE bi-directional link sub-rate limited by 25GE NIC used on TRex TG, XXV710.
- For 40GE NICs the maximum packet rate load is 2x18.75 Mpps for 64B, a 40GE bi-directional link sub-rate limited by 40GE NIC used on TRex TG,XL710. Packet rate for other tested frame sizes is limited by PCIeGen3 x8 bandwidth limitation of ~50Gbps.

MRR test code implements multiple bursts of offered packet load and has two configurable burst parameters: individual trial duration and number of trials in a single burst. This enables more precise performance trending by providing more results data for analysis.

Burst parameter settings vary between different tests using MRR:

- MRR individual trial duration:
  - Report performance comparison: 1 sec.
  - Daily performance trending: 1 sec.
  - Per-patch performance verification: 10 sec.
  - Initial iteration for MLRsearch: 1 sec.
  - Initial iteration for PLRsearch: 5.2 sec.
- Number of MRR trials per burst:
  - Report performance comparison: 10.

  - Daily performance trending: 10.
  - Per-patch performance verification: 5.
  - Initial iteration for MLRsearch: 1.
  - Initial iteration for PLRsearch: 1.

PLRsearch

Motivation for PLRsearch

Network providers are interested in throughput a system can sustain.

RFC 2544[12] assumes loss ratio is given by a deterministic function of offered load. But NFV software systems are not deterministic enough. This makes deterministic algorithms (such as binary search[13] per RFC 2544 and MLRsearch with single trial) to return results, which when repeated show relatively high standard deviation, thus making it harder to tell what "the throughput" actually is.

We need another algorithm, which takes this indeterminism into account.

Generic Algorithm

Detailed description of the PLRsearch algorithm is included in the IETF draft draft-vpolak-bmwg-plrsearch-02[14] that is in the process of being standardized in the IETF Benchmarking Methodology Working Group (BMWG).

Terms

The rest of this page assumes the reader is familiar with the following terms defined in the IETF draft:

- Trial Order Independent System
- Duration Independent System
- Target Loss Ratio
- Critical Load
- Offered Load regions
  - Zero Loss Region
  - Non-Deterministic Region
  - Guaranteed Loss Region
- Fitting Function
  - Stretch Function
  - Erf Function
- Bayesian Inference
  - Prior distribution
  - Posterior Distribution
- Numeric Integration
  - Monte Carlo

[12] https://tools.ietf.org/html/rfc2544
[13] https://en.wikipedia.org/wiki/Binary_search_algorithm
[14] https://tools.ietf.org/html/draft-vpolak-bmwg-plrsearch-02

- Importance Sampling

FD.io CSIT Implementation Specifics

The search receives min_rate and max_rate values, to avoid measurements at offered loads not supporeted by the traffic generator.

The implemented tests cases use bidirectional traffic. The algorithm stores each rate as bidirectional rate (internally, the algorithm is agnostic to flows and directions, it only cares about aggregate counts of packets sent and packets lost), but debug output from traffic generator lists unidirectional values.

Measurement Delay

In a sample implemenation in FD.io CSIT project, there is roughly 0.5 second delay between trials due to restrictons imposed by packet traffic generator in use (T-Rex).

As measurements results come in, posterior distribution computation takes more time (per sample), although there is a considerable constant part (mostly for inverting the fitting functions).

Also, the integrator needs a fair amount of samples to reach the region the posterior distribution is concentrated at.

And of course, the speed of the integrator depends on computing power of the CPU the algorithm is able to use.

All those timing related effects are addressed by arithmetically increasing trial durations with configurable coefficients (currently 5.1 seconds for the first trial, each subsequent trial being 0.1 second longer).

Rounding Errors and Underflows

In order to avoid them, the current implementation tracks natural logarithm (instead of the original quantity) for any quantity which is never negative. Logarithm of zero is minus infinity (not supported by Python), so special value "None" is used instead. Specific functions for frequent operations (such as "logarithm of sum of exponentials") are defined to handle None correctly.

Fitting Functions

Current implementation uses two fitting functions, called "stretch" and "erf". In general, their estimates for critical rate differ, which adds a simple source of systematic error, on top of randomness error reported by integrator. Otherwise the reported stdev of critical rate estimate is unrealistically low.

Both functions are not only increasing, but also convex (meaning the rate of increase is also increasing).

Both fitting functions have several mathematically equivalent formulas, each can lead to an arithmetic overflow or underflow in different sub-terms. Overflows can be eliminated by using different exact formulas for different argument ranges. Underflows can be avoided by using approximate formulas in affected argument ranges, such ranges have their own formulas to compute. At the end, both fitting function implementations contain multiple "if" branches, discontinuities are a possibility at range boundaries.

Prior Distributions

The numeric integrator expects all the parameters to be distributed (independently and) uniformly on an interval (-1, 1).

As both “mrr” and “spread” parameters are positive and not dimensionless, a transformation is needed. Dimentionality is inherited from max_rate value.

The “mrr” parameter follows a Lomax distribution[15] with alpha equal to one, but shifted so that mrr is always greater than 1 packet per second.

The “stretch” parameter is generated simply as the “mrr” value raised to a random power between zero and one; thus it follows a reciprocal distribution[16].

Integrator

After few measurements, the posterior distribution of fitting function arguments gets quite concentrated into a small area. The integrator is using Monte Carlo[17] with importance sampling[18] where the biased distribution is bivariate Gaussian[19] distribution, with deliberately larger variance. If the generated sample falls outside (-1, 1) interval, another sample is generated.

The center and the covariance matrix for the biased distribution is based on the first and second moments of samples seen so far (within the computation). The center is used directly, covariance matrix is scaled up by a heurictic constant (8.0 by default). The following additional features are applied designed to avoid hyper-focused distributions.

Each computation starts with the biased distribution inherited from the previous computation (zero point and unit covariance matrix is used in the first computation), but the overal weight of the data is set to the weight of the first sample of the computation. Also, the center is set to the first sample point. When additional samples come, their weight (including the importance correction) is compared to sum of the weights of data seen so far (within the iteration). If the new sample is more than one e-fold more impactful, both weight values (for data so far and for the new sample) are set to (geometric) average of the two weights.

This combination showed the best behavior, as the integrator usually follows two phases. First phase (where inherited biased distribution or single big sample are dominating) is mainly important for locating the new area the posterior distribution is concentrated at. The second phase (dominated by whole sample population) is actually relevant for the critical rate estimation.

Offered Load Selection

First two measurements are hardcoded to happen at the middle of rate interval and at max_rate. Next two measurements follow MRR-like logic, offered load is decreased so that it would reach target loss ratio if offered load decrease lead to equal decrease of loss rate.

The rest of measurements start directly in between erf and stretch estimate average. There is one workaround implemented, aimed at reducing the number of consequent zero loss measurements (per fitting function). The workaround first stores every measurement result which loss ratio was the targed loss ratio or higher. Sorted list (called lossy loads) of such results is maintained.

When a sequence of one or more zero loss measurement results is encountered, a smallest of lossy loads is drained from the list. If the estimate average is smaller than the drained value, a weighted average of this estimate and the drained value is used as the next offered load. The weight of the estimate decreases exponentially with the length of consecutive zero loss results.

[15] https://en.wikipedia.org/wiki/Lomax_distribution
[16] https://en.wikipedia.org/wiki/Reciprocal_distribution
[17] https://en.wikipedia.org/wiki/Monte_Carlo_integration
[18] https://en.wikipedia.org/wiki/Importance_sampling
[19] https://en.wikipedia.org/wiki/Multivariate_normal_distribution

This behavior helps the algorithm with convergence speed, as it does not need so many zero loss result to get near critical region. Using the smallest (not drained yet) of lossy loads makes it sure the new offered load is unlikely to result in big loss region. Draining even if the estimate is large enough helps to discard early measurements when loss hapened at too low offered load. Current implementation adds 4 copies of lossy loads and drains 3 of them, which leads to fairly stable behavior even for somewhat inconsistent SUTs.

Caveats

As high loss count measurements add many bits of information, they need a large amount of small loss count measurements to balance them, making the algorithm converge quite slowly. Typically, this happens when few initial measurements suggest spread way bigger then later measurements. The workaround in offered load selection helps, but more intelligent workarounds could get faster convergence still.

Some systems evidently do not follow the assumption of repeated measurements having the same average loss rate (when the offered load is the same). The idea of estimating the trend is not implemented at all, as the observed trends have varied characteristics.

Probably, using a more realistic fitting functions will give better estimates than trend analysis.

Bottom Line

The notion of Throughput is easy to grasp, but it is harder to measure with any accuracy for non-deterministic systems.

Even though the notion of critical rate is harder to grasp than the notion of throughput, it is easier to measure using probabilistic methods.

In testing, the difference between througput measurements and critical rate measurements is usually small, see *Soak Tests vs NDR Tests* (page 1205).

In pactice, rules of thumb such as “send at max 95% of purported throughput” are common. The correct benchmarking analysis should ask “Which notion is 95% of throughput an approximation to?” before attempting to answer “Is 95% of critical rate safe enough?”.

Algorithmic Analysis

Motivation

While the estimation computation is based on hard probability science; the offered load selection part of PLRsearch logic is pure heuristics, motivated by what would a human do based on measurement and computation results.

The quality of any heuristic is not affected by soundness of its motivation, just by its ability to achieve the intended goals. In case of offered load selection, the goal is to help the search to converge to the long duration estimates sooner.

But even those long duration estimates could still be of poor quality. Even though the estimate computation is Bayesian (so it is the best it could be within the applied assumptions), it can still of poor quality when compared to what a human would estimate.

One possible source of poor quality is the randomnes inherently present in Monte Carlo numeric integration, but that can be supressed by tweaking the time related input parameters.

The most likely source of poor quality then are the assumptions. Most importantly, the number and the shape of fitting functions; but also others, such as trial order independence and duration independence.

The result can have poor quality in basically two ways. One way is related to location. Both upper and lower bounds can be overestimates or underestimates, meaning the entire estimated interval between lower bound and upper bound lays above or below (respectively) of human-estimated interval. The other

way is related to the estimation interval width. The interval can be too wide or too narrow, compared to human estimation.

An estimate from a particular fitting function can be classified as an overestimate (or underestimate) just by looking at time evolution (without human examining measurement results). Overestimates decrease by time, underestimates increase by time (assuming the system performance stays constant).

Quality of the width of the estimation interval needs human evaluation, and is unrelated to both rate of narrowing (both good and bad estimate intervals get narrower at approximately the same relative rate) and relatative width (depends heavily on the system being tested).

Graphical Examples

The following pictures show the upper (red) and lower (blue) bound, as well as average of Stretch (pink) and Erf (light green) estimate, and offered load chosen (grey), as computed by PLRsearch, after each trial measurement within the 30 minute duration of a test run.

Both graphs are focusing on later estimates. Estimates computed from few initial measurements are wildly off the y-axis range shown.

The following analysis will rely on frequency of zero loss measurements and magnitude of loss ratio if nonzero.

The offered load selection strategy used implies zero loss measurements can be gleaned from the graph by looking at offered load points. When the points move up farther from lower estimate, it means the previous measurement had zero loss. After non-zero loss, the offered load starts again right between (the previous values of) the estimate curves.

The very big loss ratio results are visible as noticeable jumps of both estimates downwards. Medium and small loss ratios are much harder to distinguish just by looking at the estimate curves, the analysis is based on raw loss ratio measurement results.

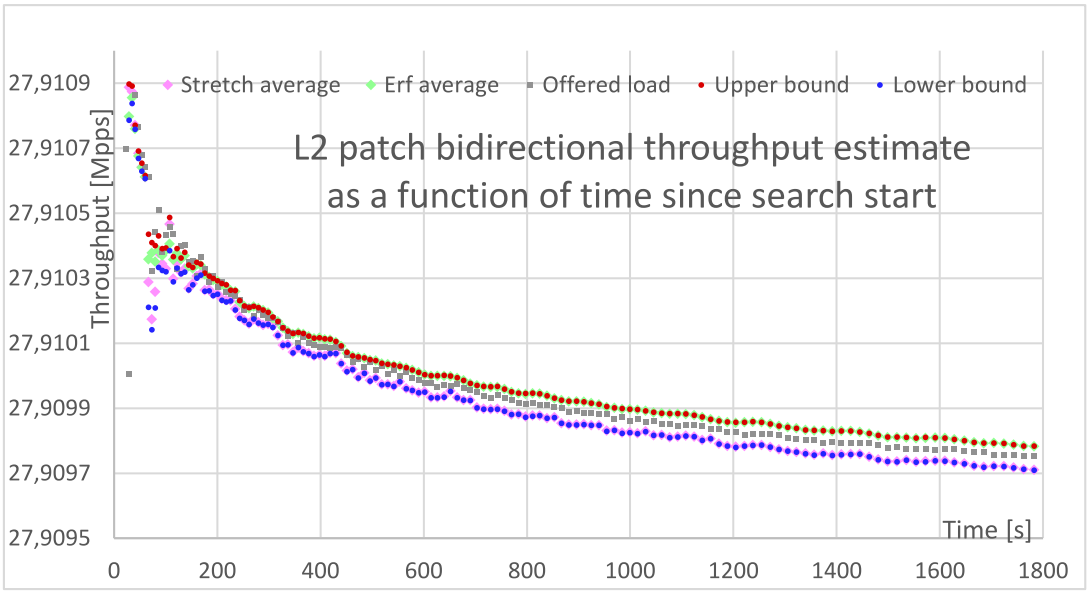
The following descriptions should explain why the graphs seem to signal low quality estimate at first sight, but a more detailed look reveals the quality is good (considering the measurement results).

L2 patch

Both fitting functions give similar estimates, the graph shows “stochasticity” of measurements (estimates increase and decrease within small time regions), and an overall trend of decreasing estimates.

On the first look, the final interval looks fairly narrow, especially compared to the region the estimates have travelled during the search. But the look at the frequency of zero loss results shows this is not a case of overestimation. Measurements at around the same offered load have higher probability of zero loss earlier (when performed farther from upper bound), but smaller probability later (when performed closer to upper bound). That means it is the performance of the system under test that decreases (slightly) over time.

With that in mind, the apparent narrowness of the interval is not a sign of low quality, just a consequence of PLRsearch assuming the performance stays constant.

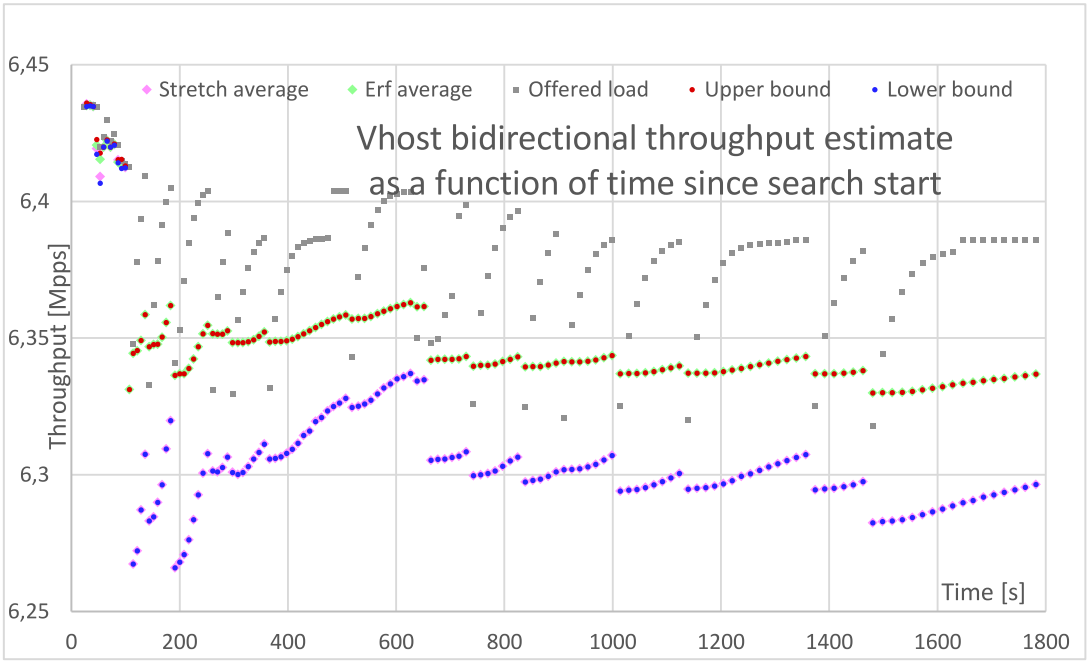


Vhost

This test case shows what looks like a quite broad estimation interval, compared to other test cases with similarly looking zero loss frequencies. Notable features are infrequent high-loss measurement results causing big drops of estimates, and lack of long-term convergence.

Any convergence in medium-sized intervals (during zero loss results) is reverted by the big loss results, as they happen quite far from the critical load estimates, and the two fitting functions extrapolate differently.

In other words, human only seeing estimates from one fitting function would expect narrower end interval, but human seeing the measured loss ratios agrees that the interval should be wider than that.



Summary

The two graphs show the behavior of PLRsearch algorithm applied to soaking test when some of PLRsearch assumptions do not hold:

- L2 patch measurement results violate the assumption of performance not changing over time.
- Vhost measurement results violate the assumption of Poisson distribution matching the loss counts.

The reported upper and lower bounds can have distance larger or smaller than a first look by a human would expect, but a more closer look reveals the quality is good, considering the circumstances.

The usefullness of the critical load estimate is of questionable value when the assumptions are violated.

Some improvements can be made via more specific workarounds, for example long term limit of L2 patch performance could be estmated by some heuristic.

Other improvements can be achieved only by asking users whether loss patterns matter. Is it better to have single digit losses distributed fairly evenly over time (as Poisson distribution would suggest), or is it better to have short periods of medium losses mixed with long periods of zero losses (as happens in Vhost test) with the same overall loss ratio?

1.5.5 TRex Traffic Generator

Usage

TRex traffic generator[20] is used for majority of CSIT performance tests. TRex is used in multiple types of performance tests, see *Data Plane Throughput Tests* (page 22) for more detail.

TRex is installed and run on the TG compute node. Versioning, installation and startup is documented in *TG Settings - TRex* (page 1211).

Traffic modes

TRex is primarily used in two (mutually incompatible) modes.

Stateless mode

Sometimes abbreviated as STL. A mode with high performance, which is unable to react to incoming traffic. We use this mode whenever it is possible. Typical test where this mode is not applicable is NAT44ED, as DUT does not assign deterministic outside address+port combinations, so we are unable to create traffic that does not lose packets in out2in direction.

Measurement results are based on simple L2 counters (opackets, ipackets) for each traffic direction.

Stateful mode

A mode capable of reacting to incoming traffic. Contrary to the stateless mode, only UDP and TCP is supported (carried over IPv4 or IPv6 packets). Performance is limited, as TRex needs to do more CPU processing. TRex suports two subtypes of stateful traffic, CSIT uses ASTF (Advanced STateFul mode).

This mode is suitable for NAT44ED tests, as clients send packets from inside, and servers react to it, so they see the outside address and port to respond to. Also, they do not send traffic before NAT44ED has created the corresponding translation entry.

When possible, L2 counters (opackets, ipackets) are used. Some tests need L7 counters, which track protocol state (e.g. TCP), but those values are less than reliable on high loads.

[20] https://trex-tgn.cisco.com

Traffic Continuity

Generated traffic is either continuous, or limited (by number of transactions). Both modes support both continuities in principle.

Continuous traffic

Traffic is started without any data size goal. Traffic is ended based on time duration, as hinted by search algorithm. This is useful when DUT behavior does not depend on the traffic duration. The default for stateless mode.

Limited traffic

Traffic has defined data size goal (given as number of transactions), duration is computed based on this goal. Traffic is ended when the size goal is reached, or when the computed duration is reached. This is useful when DUT behavior depends on traffic size, e.g. target number of NAT translation entries, each to be hit exactly once per direction. This is used mainly for stateful mode.

Traffic synchronicity

Traffic can be generated synchronously (test waits for duration) or asynchronously (test operates during traffic and stops traffic explicitly).

Synchronous traffic

Trial measurement is driven by given (or precomputed) duration, no activity from test driver during the traffic. Used for most trials.

Asynchronous traffic

Traffic is started, but then the test driver is free to perform other actions, before stopping the traffic explicitly. This is used mainly by reconf tests, but also by some trials used for runtime telemetry.

Trafic profiles

TRex supports several ways to define the traffic. CSIT uses small Python modules based on Scapy as definitions. Details of traffic profiles depend on modes (STL or ASTF), but some are common for both modes.

Search algorithms are intentionally unaware of the traffic mode used, so CSIT defines some terms to use instead of mode-specific TRex terms.

Transactions

TRex traffic profile defines a small number of behaviors, in CSIT called transaction templates. Traffic profiles also instruct TRex how to create a large number of transactions based on the templates.

Continuous traffic loops over the generated transactions. Limited traffic usually executes each transaction once (typically as constant number of loops over source addresses, each loop with different source ports).

Currently, ASTF profiles define one transaction template each. Number of packets expected per one transaction varies based on profile details, as does the criterion for when a transaction is considered successful.

Stateless transactions are just one packet (sent from one TG port, successful if received on the other TG port). Thus unidirectional stateless profiles define one transaction template, bidirectional stateless profiles define two transaction templates.

TPS multiplier

TRex aims to open transaction specified by the profile at a steady rate. While TRex allows the transaction template to define its intended "cps" value, CSIT does not specify it, so the default value of 1 is applied, meaning TRex will open one transaction per second (and transaction template) by default. But CSIT invocation uses "multiplier" (mult) argument when starting the traffic, that multiplies the cps value, meaning it acts as TPS (transactions per second) input.

With a slight abuse of nomenclature, bidirectional stateless tests set "packets per transaction" value to 2, just to keep the TPS semantics as a unidirectional input value.

Duration stretching

TRex can be IO-bound, CPU-bound, or have any other reason why it is not able to generate the traffic at the requested TPS. Some conditions are detected, leading to TRex failure, for example when the bandwidth does not fit into the line capacity. But many reasons are not detected.

Unfortunately, TRex frequently reacts by not honoring the duration in synchronous mode, taking longer to send the traffic, leading to lower then requested load offered to DUT. This usualy breaks assumptions used in search algorithms, so it has to be avoided.

For stateless traffic, the behavior is quite deterministic, so the workaround is to apply a fictional TPS limit (max_rate) to search algorithms, usually depending only on the NIC used.

For stateful traffic the behavior is not deterministic enough, for example the limit for TCP traffic depends on DUT packet loss. In CSIT we decided to use logic similar to asynchronous traffic. The traffic driver sleeps for a time, then stops the traffic explicitly. The library that parses counters into measurement results than usually treats unsent packets/transactions as lost/failed.

We have added a IP4base tests for every NAT44ED test, so that users can compare results. If the results are very similar, it is probable TRex was the bottleneck.

Startup delay

By investigating TRex behavior, it was found that TRex does not start the traffic in ASTF mode immediately. There is a delay of zero traffic, after which the traffic rate ramps up to the defined TPS value.

It is possible to poll for counters during the traffic (fist nonzero means traffic has started), but that was found to influence the NDR results.

Thus "sleep and stop" stategy is used, which needs a correction to the computed duration so traffic is stopped after the intended duration of real traffic. Luckily, it turns out this correction is not dependend on traffic profile nor CPU used by TRex, so a fixed constant (0.112 seconds) works well. Unfortunately, the constant may depend on TRex version, or execution environment (e.g. TRex in AWS).

The result computations need a precise enough duration of the real traffic, luckily server side of TRex has precise enough counter for that.

It is unknown whether stateless traffic profiles also exhibit a startup delay. Unfortunately, stateless mode does not have similarly precise duration counter, so some results (mostly MRR) are affected by less precise duration measurement in Python part of CSIT code.

Measuring Latency

If measurement of latency is requested, two more packet streams are created (one for each direction) with TRex flow_stats parameter set to STLFlowLatencyStats. In that case, returned statistics will also include min/avg/max latency values and encoded HDRHistogram data.

1.5.6 DUT state considerations

This page discusses considerations for Device Under Test (DUT) state. DUTs such as VPP require configuration, to be provided before the aplication starts (via config files) or just after it starts (via API or CLI access).

During operation DUTs gather various telemetry data, depending on configuration. This internal state handling is part of normal operation, so any performance impact is included in the test results. Accessing telemetry data is additional load on DUT, so we are not doing that in main trial measurements that affect results, but we include separate trials specifically for gathering runtime telemetry.

But there is one kind of state that needs specific handling. This kind of DUT state is dynamically created based on incoming traffic, it affects how DUT handles the traffic, and (unlike telemetry counters) it has uneven impact on CPU load. Typical example is NAT, where detecting new sessions takes more CPU than forwarding packet on existing (open or recently closed) sessions. We call DUT configurations with this kind of state "stateful", and configurations without them "stateless". (Even though stateless configurations contain state described in previous paragraphs, and some configuration items may have "stateful" in their name, such as stateful ACLs.)

Stateful DUT configurations

Typically, the level of CPU impact of traffic depends on DUT state. The first packets causing DUT state to change have higher impact, subsequent packets matching that state have lower impact.

From performance point of view, this is similar to traffic phases for stateful protocols, see *NGFW draft <https://tools.ietf.org/html/draft-ietf-bmwg-ngfw-performance-05#section-4.3.4>*. In CSIT we borrow the terminology (even if it does not fit perfectly, see discussion below). Ramp-up traffic causes the state change, sustain traffic does not change the state.

As the performance is different, each test has to choose which traffic it wants to test, and manipulate the DUT state to achieve the intended impact.

Ramp-up trial

Tests aiming at sustain performance need to make sure DUT state is created. We achieve this via a ramp-up trial, specific purpose of which is to create the state.

Subsequent trials need no specific handling, as long as the state remains the same. But some state can time-out, so additional ramp-up trials are inserted whenever the code detects the state can time-out. Note that a trial with zero loss refreshes the state, so only the time since the last non-zero loss trial is tracked.

For the state to be set completely, it is important both DUT and TG do not lose any packets. We achieve this by setting the profile multiplier (TPS from now on) to low enough value.

It is also important each state-affecting packet is sent. For size-limited traffic profile it is guaranteed by the size limit. For continuous traffic, we set a long enough duration (based on TPS).

At the end of the ramp-up trial, we check DUT state to confirm it has been created as expected. Test fails if the state is not (completely) created.

State Reset

Tests aiming at ramp-up performance do not use ramp-up trial, and they need to reset the DUT state before each trial measurement. The way of resetting the state depends on test, usually an API call is used to partially de-configure the part that holds the state, and then re-configure it back.

In CSIT we control the DUT state behavior via a test variable "resetter". If it is not set, DUT state is not reset. If it is set, each search algorithm (including MRR) will invoke it before all trial measurements (both main and telemetry ones). Any configuration keyword enabling a feature with DUT state will check whether a test variable for ramp-up rate is present. If it is present, resetter is not set. If it is not present, the keyword sets the apropriate resetter value. This logic makes sure either ramp-up or state reset are used.

Notes: If both ramp-up and state reset were used, the DUT behavior would be identical to just reset, while test would take longer to execute. If neither were used, DUT will show different performance in subsequent trials, violating assumptions of search algorithms.

DUT versus protocol ramp-up

There are at least three different causes for bandwidth possibly increasing within a single measurement trial.

The first is DUT switching from state modification phase to constant phase, it is the primary focus of this document. Using ramp-up traffic before main trials eliminates this cause for tests wishing to measure the performance of the next phase. Using size-limited profiles eliminates the next phase for tests wishing to measure performance of this phase.

The second is protocol such as TCP ramping up their throughput to utilize the bandwidth available. This is the original meaning of "ramp up" in the NGFW draft (see above). In existing tests we are not using this meaning of TCP ramp-up. Instead we use only small transactions, and large enough initial window so TCP acts as ramped-up already.

The third is TCP increasing offered load due to retransmissions triggered by packet loss. In CSIT we again try to avoid this behavior by using small enough data to transfer, so overlap of multiple transactions (primary cause of packet loss) is unlikely. But in MRR tests, packet loss and non-constant offered load are still expected.

Stateless DUT configuratons

These are simple configurations, which do not set any resetter value (even if ramp-up duration is not configured). Majority of existing tests are of this type, using continuous traffic profiles.

In order to identify limits of Trex performance, we have added suites with stateless DUT configuration (VPP ip4base) subjected to size-limited ASTF traffic. The discovered rates serve as a basis of comparison for evaluating the results for stateful DUT configurations (VPP NAT44ed) subjected to the same traffic profiles.

DUT versus TG state

Traffic Generator profiles can be stateful (ASTF) or stateless (STL). DUT configuration can be stateful or stateless (with respect to packet traffic).

In CSIT we currently use all four possible configurations:

- Regular stateless VPP tests use stateless traffic profiles.
- Stateless VPP configuration with stateful profile is used as a base for comparison.
- Some stateful DUT configurations (NAT44DET, NAT44ED unidirectional) are tested using stateless traffic profiles and continuous traffic.

- The rest of stateful DUT configurations (NAT44ED bidirectional) are tested using stateful traffic profiles and size limited traffic.

1.5.7 Network Address Translation IPv4 to IPv4

NAT44 Prefix Bindings

NAT44 prefix bindings should be representative to target applications, where a number of private IPv4 addresses from the range defined by RFC 1918[21] is mapped to a smaller set of public IPv4 addresses from the public range.

Following quantities are used to describe inside to outside IP address and port bindings scenarios:

- Inside-addresses, number of inside source addresses (representing inside hosts).
- Ports-per-inside-address, number of TCP/UDP source ports per inside source address.
- Outside-addresses, number of outside (public) source addresses allocated to NAT44.
- Ports-per-outside-address, number of TCP/UDP source ports per outside source address. The maximal number of ports-per-outside-address usable for NAT is 64 512 (in non-reserved port range 1024-65535, RFC 4787[22]).
- Sharing-ratio, equal to inside-addresses divided by outside-addresses.

CSIT NAT44 tests are designed to take into account the maximum number of ports (sessions) required per inside host (inside-address) and at the same time to maximize the use of outside-address range by using all available outside ports. With this in mind, the following scheme of NAT44 sharing ratios has been devised for use in CSIT:

ports-per-inside-address	sharing-ratio
63	1024
126	512
252	256
504	128

Initial CSIT NAT44 tests, including associated TG/TRex traffic profiles, are based on ports-per-inside-address set to 63 and the sharing ratio of 1024. This approach is currently used for all NAT44 tests including NAT44det (NAT44 deterministic used for Carrier Grade NAT applications) and NAT44ed (Endpoint Dependent).

Private address ranges to be used in tests:

- 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)
  - Total of 2^16 (65 536) of usable IPv4 addresses.
  - Used in tests for up to 65 536 inside addresses (inside hosts).
- 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
  - Total of 2^20 (1 048 576) of usable IPv4 addresses.
  - Used in tests for up to 1 048 576 inside addresses (inside hosts).

[21] https://datatracker.ietf.org/doc/html/rfc1918.html
[22] https://datatracker.ietf.org/doc/html/rfc4787.html

NAT44 Session Scale

NAT44 session scale tested is govern by the following logic:

- Number of inside-addresses(hosts) H[i] = (H[i-1] x 2^2) with H(0)=1 024, i = 1,2,3, ...
  - H[i] = 1 024, 4 096, 16 384, 65 536, 262 144, ...
- Number of sessions S[i] = H[i] * ports-per-inside-address
  - ports-per-inside-address = 63

i	hosts	sessions
0	1 024	64 512
1	4 096	258 048
2	16 384	1 032 192
3	65 536	4 128 768
4	262 144	16 515 072

NAT44 Deterministic

NAT44det performance tests are using TRex STL (Stateless) API and traffic profiles, similar to all other stateless packet forwarding tests like ip4, ip6 and l2, sending UDP packets in both directions inside-to-outside and outside-to-inside. See *Data Plane Throughput Tests* (page 22) for more detail.

The inside-to-outside traffic uses single destination address (20.0.0.0) and port (1024). The inside-to-outside traffic covers whole inside address and port range, the outside-to-inside traffic covers whole outside address and port range.

NAT44det translation entries are created during the ramp-up phase, followed by verification that all entries are present, before proceeding to the main measurements of the test. This ensures session setup does not impact the forwarding performance test.

Associated CSIT test cases use the following naming scheme to indicate NAT44det scenario tested:

- ethip4udp-nat44det-h{H}-p{P}-s{S}-[mrr|ndrpdr|soak]
  - {H}, number of inside hosts, H = 1024, 4096, 16384, 65536, 262144.
  - {P}, number of ports per inside host, P = 63.
  - {S}, number of sessions, S = 64512, 258048, 1032192, 4128768, 16515072.
  - [mrr|ndrpdr|soak], MRR, NDRPDR or SOAK test.

NAT44 Endpoint-Dependent

In order to excercise NAT44ed ability to translate based on both source and destination address and port, the inside-to-outside traffic varies also destination address and port. Destination port is the same as source port, destination address has the same offset as the source address, but applied to different subnet (starting with 20.0.0.0).

As the mapping is not deterministic (for security reasons), we cannot easily use stateless bidirectional traffic profiles. Inside address and port range is fully covered, but we do not know which outside-to-inside source address and port to use to hit an open session.

Therefore, NAT44ed is benchmarked using following methodologies:

- Unidirectional throughput using *stateless* traffic profile.
- Connections-per-second (CPS) using *stateful* traffic profile.
- Bidirectional throughput (TPUT, see below) using *stateful* traffic profile.

Unidirectional NAT44ed throughput tests are using TRex STL (Stateless) APIs and traffic profiles, but with packets sent only in inside-to-outside direction. Similarly to NAT44det, NAT44ed unidirectional throughput tests include a ramp-up phase to establish and verify the presence of required NAT44ed binding entries. As the sessions have finite duration, the test code keeps inserting ramp-up trials during the search, if it detects a risk of sessions timing out. Any zero loss trial visits all sessions, so it acts also as a ramp-up.

Stateful NAT44ed tests are using TRex ASTF (Advanced Stateful) APIs and traffic profiles, with packets sent in both directions. Tests are run with both UDP and TCP sessions. As NAT44ed CPS (connections-per-second) stateful tests measure (also) session opening performance, they use state reset instead of ramp-up trial. NAT44ed TPUT (bidirectional throughput) tests prepend ramp-up trials as in the unidirectional tests, so the test results describe performance without translation entry creation overhead.

Associated CSIT test cases use the following naming scheme to indicate NAT44det case tested:

- Stateless: ethip4udp-nat44ed-h{H}-p{P}-s{S}-udir-[mrr|ndrpdr|soak]
  - {H}, number of inside hosts, H = 1024, 4096, 16384, 65536, 262144.
  - {P}, number of ports per inside host, P = 63.
  - {S}, number of sessions, S = 64512, 258048, 1032192, 4128768, 16515072.
  - udir-[mrr|ndrpdr|soak], unidirectional stateless tests MRR, NDRPDR or SOAK.
- Stateful: ethip4[udp|tcp]-nat44ed-h{H}-p{P}-s{S}-[cps|tput]-[mrr|ndrpdr|soak]
  - [udp|tcp], UDP or TCP sessions
  - {H}, number of inside hosts, H = 1024, 4096, 16384, 65536, 262144.
  - {P}, number of ports per inside host, P = 63.
  - {S}, number of sessions, S = 64512, 258048, 1032192, 4128768, 16515072.
  - [cps|tput], connections-per-second session establishment rate or packets-per-second average rate, or packets-per-second rate without session establishment.
  - [mrr|ndrpdr|soak], bidirectional stateful tests MRR, NDRPDR, or SOAK.

Stateful traffic profiles

There are several important details which distinguish ASTF profiles from stateless profiles.

General considerations

Protocols

ASTF profiles are limited to either UDP or TCP protocol.

Programs

Each template in the profile defines two “programs”, one for the client side and one for the server side.

Each program specifies when that side has to wait until enough data is received (counted in packets for UDP and in bytes for TCP) and when to send additional data. Together, the two programs define a single transaction. Due to packet loss, transaction may take longer, use more packets (retransmission) or never finish in its entirety.

Instances

A client instance is created according to TPS parameter for the trial, and sends the first packet of the transaction (in some cases more packets). Each client instance uses a different source address (see sequencing below) and some source port. The destination address also comes from a range, but destination port has to be constant for a given program.

TRex uses an opaque way to chose source ports, but as session counting shows, next client with the same source address uses a different source port.

Server instance is created when the first packet arrives to the server side. Source address and port of the first packet are used as destination address and port for the server responses. This is the ability we need when outside surface is not predictable.

When a program reaches its end, the instance is deleted. This creates possible issues with server instances. If the server instance does not read all the data client has sent, late data packets can cause a second copy of server instance to be created, which breaks assumptions on how many packet a transaction should have.

The need for server instances to read all the data reduces the overall bandwidth TRex is able to create in ASTF mode.

Note that client instances are not created on packets, so it is safe to end client program without reading all server data (unless the definition of transaction success requires that).

Sequencing

ASTF profiles offer two modes for choosing source and destination IP addresses for client programs: seqential and pseudorandom. In current tests we are using sequential addressing only (if destination address varies at all).

For client destination UDP/TCP port, we use a single constant value. (TRex can support multiple program pairs in the same traffic profile, distinguished by the port number.)

Transaction overlap

If a transaction takes longer to finish, compared to period implied by TPS, TRex will have multiple client or server instances active at a time.

During calibration testing we have found this increases CPU utilization, and for high TPS it can lead to TRex's Rx or Tx buffers becoming full. This generally leads to duration stretching, and/or packet loss on TRex.

Currently used transactions were chosen to be short, so risk of bad behavior is decreased. But in MRR tests, where load is computed based on NIC ability, not TRex ability, anomalous behavior is still possible (e.g. MRR values being way lower than NDR).

Delays

TRex supports adding constant delays to ASTF programs. This can be useful, for example if we want to separate connection establishment from data transfer.

But as TRex tracks delayed instances as active, this still results in higher CPU utilization and reduced performance issues (as other overlaping transactions). So the current tests do not use any delays.

Keepalives

Both UDP and TCP protocol implementations in TRex programs support keepalive duration. That means there is a configurable period of keepalive time, and TRex sends keepalive packets automatically (outside the program) for the time the program is active (started, not ended yet) but not sending any packets.

For TCP this is generally not a big deal, as the other side usually retransmits faster. But for UDP it means a packet loss may leave the receiving program running.

In order to avoid keepalive packets, keepalive value is set to a high number. Here, "high number" means that even at maximum scale and minimum TPS, there are still no keepalive packets sent within the corresponding (computed) trial duration. This number is kept the same also for smaller scale traffic profiles, to simplify maintenance.

Transaction success

The transaction is considered successful at Layer-7 (L7) level when both program instances close. At this point, various L7 counters (unofficial name) are updated on TRex.

We found that proper close and L7 counter update can be CPU intensive, whereas lower-level counters (ipackets, opackets) called L2 counters can keep up with higher loads.

For some tests, we do not need to confirm the whole transaction was successful. CPS (connections per second) tests are a typical example. We care only for NAT44ed creating a session (needs one packet in inside-to-outside direction per session) and being able to use it (needs one packet in outside-to-inside direction).

Similarly in TPUT tests (packet throuput, counting both control and data packets), we care about NAT44ed ability to forward packets, we do not care whether aplications (TRex) can fully process them at that rate.

Therefore each type of tests has its own formula (usually just one counter already provided by TRex) to count "successful enough" transactions and attempted transactions. Currently, all tests relying on L7 counters use size-limited profiles, so they know what the count of attempted transactions should be, but due to duration stretching TRex might have been unable to send that many packets. For search purposes, unattempted transactions are treated the same as attempted but failed transactions.

Sometimes even the number of transactions as tracked by search algorithm does not match the transactions as defined by ASTF programs. See TCP TPUT profile below.

UDP CPS

This profile uses a minimalistic transaction to verify NAT44ed session has been created and it allows outside-to-inside traffic.

Client instance sends one packet and ends. Server instance sends one packet upon creation and ends.

In principle, packet size is configurable, but currently used tests apply only one value (100 bytes frame).

Transaction counts as attempted when opackets counter increases on client side. Transaction counts as successful when ipackets counter increases on client side.

TCP CPS

This profile uses a minimalistic transaction to verify NAT44ed session has been created and it allows outside-to-inside traffic.

Client initiates TCP connection. Client waits until connection is confirmed (by reading zero data bytes). Client ends. Server accepts the connection. Server waits for indirect confirmation from client (by waiting for client to initiate close). Server ends.

Without packet loss, the whole transaction takes 7 packets to finish (4 and 3 per direction). From NAT44ed point of view, only the first two are needed to verify the session got created.

Packet size is not configurable, but currently used tests report frame size as 64 bytes.

Transaction counts as attempted when tcps_connattempt counter increases on client side. Transaction counts as successful when tcps_connects counter increases on client side.

UDP TPUT

This profile uses a small transaction of "request-response" type, with several packets simulating data payload.

Client sends 5 packets and closes immediately. Server reads all 5 packets (needed to avoid late packets creating new server instances), then sends 5 packets and closes. The value 5 was chosen to mirror what TCP TPUT (see below) choses.

Packet size is configurable, currently we have tests for 100, 1518 and 9000 bytes frame (to match size of TCP TPUT data frames, see below).

As this is a packet oriented test, we do not track the whole 10 packet transaction. Similarly to stateless tests, we treat each packet as a "transaction" for search algorithm packet loss ratio purposes. Therefore a "transaction" is attempted when opacket counter on client or server side is increased. Transaction is successful if ipacket counter on client or server side is increased.

If one of 5 client packets is lost, server instance will get stuck in the reading phase. This probably decreases TRex performance, but it leads to more stable results then alternatives.

TCP TPUT

This profile uses a small transaction of "request-response" type, with some data amount to be transferred both ways.

In CSIT release 22.06, TRex behavior changed, so we needed to edit the traffic profile. Let us describe the pre-22.06 profile first.

Client connects, sends 5 data packets worth of data, receives 5 data packets worth of data and closes its side of the connection. Server accepts connection, reads 5 data packets worth of data, sends 5 data packets worth of data and closes its side of the connection. As usual in TCP, sending side waits for ACK from the receiving side before proceeding with next step of its program.

Server read is needed to avoid premature close and second server instance. Client read is not stricly needed, but ACKs allow TRex to close the server instance quickly, thus saving CPU and improving performance.

The number 5 of data packets was chosen so TRex is able to send them in a single burst, even with 9000 byte frame size (TRex has a hard limit on initial window size). That leads to 16 packets (9 of them in c2s direction) to be exchanged if no loss occurs. The size of data packets is controlled by the traffic profile setting the appropriate maximum segment size. Due to TRex restrictions, the minimal size for IPv4 data frame achievable by this method is 70 bytes, which is more than our usual minimum of 64 bytes. For that reason, the data frame sizes available for testing are 100 bytes (that allows room for eventually adding IPv6 ASTF tests), 1518 bytes and 9000 bytes. There is no control over control packet sizes.

Exactly as in UDP TPUT, ipackets and opackets counters are used for counting "transactions" (in fact packets).

If packet loss occurs, there can be large transaction overlap, even if most ASTF programs finish eventually. This can lead to big duration stretching and somehow uneven rate of packets sent. This makes it hard to interpret MRR results (frequently MRR is below NDR for this reason), but NDR and PDR results tend to be stable enough.

In 22.06, the "ACK from the receiving side" behavior changed, the receiving side started sending ACK sometimes also before receiving the full set of 5 data packets. If the previous profile is understood as a "single challenge, single response" where challenge (and also response) is sent as a burst of 5 data packets, the new profile uses "bursts" of 1 packet instead, but issues the challenge-response part 5 times sequentially (waiting for receiving the response before sending next challenge). This new profile happens to have the same overall packet count (when no re-transmissions are needed). Although it is possibly more taxing for TRex CPU, the results are comparable to the old traffic profile.

Ip4base tests

Contrary to stateless traffic profiles, we do not have a simple limit that would guarantee TRex is able to send traffic at specified load. For that reason, we have added tests where "nat44ed" is replaced by "ip4base". Instead of NAT44ed processing, the tests set minimalistic IPv4 routes, so that packets are forwarded in both inside-to-outside and outside-to-inside directions.

The packets arrive to server end of TRex with different source address&port than in NAT44ed tests (no translation to outside values is done with ip4base), but those are not specified in the stateful traffic profiles. The server end (as always) uses the received address&port as destination for outside-to-inside traffic. Therefore the same stateful traffic profile works for both NAT44ed and ip4base test (of the same scale).

The NAT44ed results are displayed together with corresponding ip4base results. If they are similar, TRex is probably the bottleneck. If NAT44ed result is visibly smaller, it describes the real VPP performance.

1.5.8 Packet Latency

TRex Traffic Generator (TG) is used for measuring one-way latency in 2-Node and 3-Node physical testbed topologies. TRex integrates High Dynamic Range Histogram (HDRH)[23] functionality and reports per packet latency distribution for latency streams sent in parallel to the main load packet streams.

Following methodology is used:

- Only NDRPDR test type measures latency and only after NDR and PDR values are determined. Other test types do not involve latency streams.
- Latency is measured at different background load packet rates:
  - No-Load: latency streams only.
  - Low-Load: at 10% PDR.
  - Mid-Load: at 50% PDR.
  - High-Load: at 90% PDR.
- Latency is measured for all tested packet sizes except IMIX due to TRex TG restriction.
- TG sends dedicated latency streams, one per direction, each at the rate of 9 kpps at the prescribed packet size; these are sent in addition to the main load streams.
- TG reports Min/Avg/Max and HDRH latency values distribution per stream direction, hence two sets of latency values are reported per test case (marked as E-W and W-E).
- +/- 1 usec is the measurement accuracy of TRex TG and the data in HDRH latency values distribution is rounded to microseconds.

[23] http://hdrhistogram.org/

- TRex TG introduces a (background) always-on Tx + Rx latency bias of 4 usec on average per direction resulting from TRex software writing and reading packet timestamps on CPU cores. Quoted values are based on TG back-to-back latency measurements.
- Latency graphs are not smoothed, each latency value has its own horizontal line across corresponding packet percentiles.
- Percentiles are shown on X-axis using a logarithmic scale, so the maximal latency value (ending at 100% percentile) would be in infinity. The graphs are cut at 99.9999% (hover information still lists 100%).

1.5.9 Packet Flow Ordering

TRex Traffic Generator (TG) supports two main ways how to cover address space (on allowed ranges) in scale tests.

In most cases only one field value (e.g. IPv4 destination address) is altered, in some cases two fields (e.g. IPv4 destination address and UDP destination port) are altered.

Incremental Ordering

This case is simpler to implement and offers greater control.

When changing two fields, they can be incremented synchronously, or one after another. In the latter case we can specify which one is incremented each iteration and which is incremented by "carrying over" only when the other "wraps around". This way also visits all combinations once before the "carry" field also wraps around.

It is possible to use increments other than 1.

Randomized Ordering

This case chooses each field value at random (from the allowed range). In case of two fields, they are treated independently. TRex allows to set random seed to get deterministic numbers. We use a different seed for each field and traffic direction. The seed has to be a non-zero number, we use 1, 2, 3, and so on.

The seeded random mode in TRex requires a "limit" value, which acts as a cycle length limit (after this many iterations, the seed resets to its initial value). We use the maximal allowed limit value (computed as 2^24 - 1).

Randomized profiles do not avoid duplicated values, and do not guarantee each possible value is visited, so it is not very useful for stateful tests.

1.5.10 Tunnel Encapsulations

Tunnel encapsulations testing is grouped based on the type of outer header: IPv4 or IPv6.

IPv4 Tunnels

VPP is tested in the following IPv4 tunnel baseline configurations:

- *ip4vxlan-l2bdbase*: VXLAN over IPv4 tunnels with L2 bridge-domain MAC switching.
- *ip4vxlan-l2xcbase*: VXLAN over IPv4 tunnels with L2 cross-connect.
- *ip4lispip4-ip4base*: LISP over IPv4 tunnels with IPv4 routing.
- *ip4lispip6-ip6base*: LISP over IPv4 tunnels with IPv6 routing.
- *ip4gtpusw-ip4base*: GTPU over IPv4 tunnels with IPv4 routing.

In all cases listed above low number of MAC, IPv4, IPv6 flows (253 or 254 per direction) is switched or routed by VPP.

In addition selected IPv4 tunnels are tested at scale:

- *dot1q-ip4vxlanscale-l2bd*: VXLAN over IPv4 tunnels with L2 bridge- domain MAC switching, with scaled up dot1q VLANs (10, 100, 1k), mapped to scaled up L2 bridge-domains (10, 100, 1k), that are in turn mapped to (10, 100, 1k) VXLAN tunnels. 64.5k flows are transmitted per direction.

IPv6 Tunnels

VPP is tested in the following IPv6 tunnel baseline configurations:

- *ip6lispip4-ip4base*: LISP over IPv4 tunnels with IPv4 routing.
- *ip6lispip6-ip6base*: LISP over IPv4 tunnels with IPv6 routing.

In all cases listed above low number of IPv4, IPv6 flows (253 or 254 per direction) is routed by VPP.

1.5.11 Internet Protocol Security (IPsec)

VPP IPsec performance tests are executed for the following crypto plugins:

- *crypto_native*, used for software based crypto leveraging CPU platform optimizations e.g. Intel's AES-NI instruction set.
- *crypto_ipsecmb*, used for hardware based crypto with Intel QAT PCIe cards.

IPsec with VPP Native SW Crypto

Currently CSIT-2210 implements following IPsec test cases relying on VPP native crypto (*crypto_native* plugin):

VPP Crypto Engine	ESP Encryption	ESP Integrity	Scale Tested
crypto_native	AES[128|256]-GCM	GCM	1 to 60k tunnels
crypto_native	AES128-CBC	SHA[256|512]	1 to 60k tunnels

VPP IPsec with SW crypto are executed in both tunnel and policy modes, with tests running on 3-node testbeds: 3n-icx, 3n-tsh.

IPsec with Intel QAT HW

Currently CSIT-2210 implements following IPsec test cases relying on ipsecmb library (*crypto_ipsecmb* plugin) and Intel QAT 8950 (50G HW crypto card):

dpdk_cryptodev

VPP Crypto Engine	VPP Crypto Workers	ESP Encryption	ESP Integrity	Scale Tested
crypto_ipsecmb	sync/all workers	AES[128|256]-GCM	GCM	1, 1k tunnels
crypto_ipsecmb	sync/all workers	AES[128]-CBC	SHA[256|512]	1, 1k tunnels
crypto_ipsecmb	async/crypto worker	AES[128|256]-GCM	GCM	1, 4, 1k tunnels
crypto_ipsecmb	async/crypto worker	AES[128]-CBC	SHA[256|512]	1, 4, 1k tunnels

IPsec with Async Crypto Feature Workers

TODO Description to be added

IPsec Uni-Directional Tests with VPP Native SW Crypto

Currently CSIT-2210 implements following IPsec uni-directional test cases relying on VPP native crypto (*crypto_native* plugin) in tunnel mode:

VPP Crypto Engine	ESP Encryption	ESP Integrity	Scale Tested
crypto_native	AES[128|256]-GCM	GCM	4, 1k, 10k tunnels
crypto_native	AES128-CBC	SHA[512]	4, 1k, 10k tunnels

In policy mode:

VPP Crypto Engine	ESP Encryption	ESP Integrity	Scale Tested
crypto_native	AES[256]-GCM	GCM	1, 40, 1k tunnels

The tests are running on 2-node testbeds: 2n-tx2. The uni-directional tests are partially addressing a weakness in 2-node testbed setups with T-Rex as the traffic generator. With just one DUT node, we can either encrypt or decrypt traffic in each direction.

The testcases are only doing encryption - packets are encrypted on the DUT and then arrive at TG where no additional packet processing is needed (just counting packets).

Decryption would require that the traffic generator generated encrypted packets which the DUT then would decrypt. However, T-Rex does not have the capability to encrypt packets.

1.5.12 Access Control Lists

VPP is tested in a number of data plane feature configurations across different forwarding modes. Following sections list features tested.

ACL Security-Groups

Both stateless and stateful access control lists (ACL), also known as security-groups, are supported by VPP.

Following ACL configurations are tested for MAC switching with L2 bridge-domains:

- *l2bdbasemaclrn-iacl{E}sl-{F}flows*: Input stateless ACL, with {E} entries and {F} flows.
- *l2bdbasemaclrn-oacl{E}sl-{F}flows*: Output stateless ACL, with {E} entries and {F} flows.
- *l2bdbasemaclrn-iacl{E}sf-{F}flows*: Input stateful ACL, with {E} entries and {F} flows.
- *l2bdbasemaclrn-oacl{E}sf-{F}flows*: Output stateful ACL, with {E} entries and {F} flows.

Following ACL configurations are tested with IPv4 routing:

- *ip4base-iacl{E}sl-{F}flows*: Input stateless ACL, with {E} entries and {F} flows.
- *ip4base-oacl{E}sl-{F}flows*: Output stateless ACL, with {E} entries and {F} flows.
- *ip4base-iacl{E}sf-{F}flows*: Input stateful ACL, with {E} entries and {F} flows.
- *ip4base-oacl{E}sf-{F}flows*: Output stateful ACL, with {E} entries and {F} flows.

ACL tests are executed with the following combinations of ACL entries and number of flows:

- ACL entry definitions

  - flow non-matching deny entry: (src-ip4, dst-ip4, src-port, dst-port).
  - flow matching permit ACL entry: (src-ip4, dst-ip4).
- {E} - number of non-matching deny ACL entries, {E} = [1, 10, 50].
- {F} - number of UDP flows with different tuple (src-ip4, dst-ip4, src-port, dst-port), {F} = [100, 10k, 100k].
- All {E}x{F} combinations are tested per ACL type, total of 9.

ACL MAC-IP

MAC-IP binding ACLs are tested for MAC switching with L2 bridge-domains:

- *l2bdbasemaclrn-macip-iacl{E}sl-{F}flows*: Input stateless ACL, with {E} entries and {F} flows.

MAC-IP ACL tests are executed with the following combinations of ACL entries and number of flows:

- ACL entry definitions
  - flow non-matching deny entry: (dst-ip4, dst-mac, bit-mask)
  - flow matching permit ACL entry: (dst-ip4, dst-mac, bit-mask)
- {E} - number of non-matching deny ACL entries, {E} = [1, 10, 50]
- {F} - number of UDP flows with different tuple (dst-ip4, dst-mac), {F} = [100, 10k, 100k]
- All {E}x{F} combinations are tested per ACL type, total of 9.

1.5.13 Multi-Core Speedup

All performance tests are executed with single physical core and with multiple cores scenarios.

Intel Hyper-Threading (HT)

Intel Xeon processors used in FD.io CSIT can operate either in HT Disabled mode (single logical core per each physical core) or in HT Enabled mode (two logical cores per each physical core). HT setting is applied in BIOS and requires server SUT reload for it to take effect, making it impractical for continuous changes of HT mode of operation.

CSIT-2210 performance tests are executed with server SUTs' Intel XEON processors configured with Intel Hyper-Threading Enabled for all Xeon Skylake and Xeon Cascadelake testbeds.

More information about physical testbeds is provided in *Performance Physical Testbeds* (page 4).

Multi-core Tests

CSIT-2210 multi-core tests are executed in the following VPP worker thread and physical core configurations:

1. Intel Xeon Icelake and Cascadelake testbeds (2n-icx, 3n-icx, 2n-clx) with Intel HT enabled (2 logical CPU cores per each physical core):

1. 2t1c - 2 VPP worker threads on 1 physical core.
2. 4t2c - 4 VPP worker threads on 2 physical cores.
3. 8t4c - 8 VPP worker threads on 4 physical cores.

VPP worker threads are the data plane threads running on isolated logical cores. With Intel HT enabled VPP workers are placed as sibling threads on each used physical core. VPP control threads (main, stats) are running on a separate non-isolated core together with other Linux processes.

In all CSIT tests care is taken to ensure that each VPP worker handles the same amount of received packet load and does the same amount of packet processing work. This is achieved by evenly distributing per interface type (e.g. physical, virtual) receive queues over VPP workers using default VPP round-robin mapping and by loading these queues with the same amount of packet flows.

If number of VPP workers is higher than number of physical or virtual interfaces, multiple receive queues are configured on each interface. NIC Receive Side Scaling (RSS) for physical interfaces and multi-queue for virtual interfaces are used for this purpose.

Section *Speedup Multi-Core* (page 426) includes a set of graphs illustrating packet throughout speedup when running VPP worker threads on multiple cores. Note that in quite a few test cases running VPP workers on 2 or 4 physical cores hits the I/O bandwidth or packets-per-second limit of tested NIC.

1.5.14 Hoststack Testing

TCP/IP with iperf3

iperf3 goodput measurement tool[24] is used for measuring the maximum attainable goodput of the VPP Host Stack connection across two instances of VPP running on separate DUT nodes. iperf3 is a popular open source tool for active measurements of the maximum achievable goodput on IP networks.

Because iperf3 utilizes the POSIX socket interface APIs, the current test configuration utilizes the LD_PRELOAD mechanism in the linux kernel to connect iperf3 to the VPP Host Stack using the VPP Communications Library (VCL) LD_PRELOAD library (libvcl_ldpreload.so).

In the future, a forked version of iperf3 which has been modified to directly use the VCL application APIs may be added to determine the difference in performance of 'VCL Native' applications versus utilizing LD_PRELOAD which inherently has more overhead and other limitations.

The test configuration is as follows:

```
       DUT1              Network               DUT2
[ iperf3-client -> VPP1 ]=======[ VPP2 -> iperf3-server]
```

where,

1. iperf3 server attaches to VPP2 and LISTENs on VPP2:TCP port 5201.
2. iperf3 client attaches to VPP1 and opens one or more stream connections to VPP2:TCP port 5201.
3. iperf3 client transmits a uni-directional stream as fast as the VPP Host Stack allows to the iperf3 server for the test duration.
4. At the end of the test the iperf3 client emits the goodput measurements for all streams and the sum of all streams.

Test cases include 1 and 10 Streams with a 20 second test duration with the VPP Host Stack configured to utilize the Cubic TCP congestion algorithm.

Note: iperf3 is single threaded, so it is expected that the 10 stream test shows little or no performance improvement due to multi-thread/multi-core execution.

There are also variations of these test cases which use the VPP Network Simulator (NSIM) plugin to test the VPP Hoststack goodput with 1 percent of the traffic being dropped at the output interface of VPP1 thereby simulating a lossy network. The NSIM tests are experimental and the test results are not currently representative of typical results in a lossy network.

[24] https://github.com/esnet/iperf

UDP/IP with iperf3

iperf3 goodput measurement tool[25] is used for measuring the maximum attainable goodput of the VPP Host Stack connection across two instances of VPP running on separate DUT nodes. iperf3 is a popular open source tool for active measurements of the maximum achievable goodput on IP networks.

Because iperf3 utilizes the POSIX socket interface APIs, the current test configuration utilizes the LD_PRELOAD mechanism in the linux kernel to connect iperf3 to the VPP Host Stack using the VPP Communications Library (VCL) LD_PRELOAD library (libvcl_ldpreload.so).

In the future, a forked version of iperf3 which has been modified to directly use the VCL application APIs may be added to determine the difference in performance of 'VCL Native' applications versus utilizing LD_PRELOAD which inherently has more overhead and other limitations.

The test configuration is as follows:

```
       DUT1              Network               DUT2
[ iperf3-client -> VPP1 ]=======[ VPP2 -> iperf3-server]
```

where,

1. iperf3 server attaches to VPP2 and LISTENs on VPP2:UDP port 5201.
2. iperf3 client attaches to VPP1 and transmits one or more streams of packets to VPP2:UDP port 5201.
3. iperf3 client transmits a uni-directional stream as fast as the VPP Host Stack allows to the iperf3 server for the test duration.
4. At the end of the test the iperf3 client emits the goodput measurements for all streams and the sum of all streams.

Test cases include 1 and 10 Streams with a 20 second test duration with the VPP Host Stack using the UDP transport layer..

Note: iperf3 is single threaded, so it is expected that the 10 stream test shows little or no performance improvement due to multi-thread/multi-core execution.

QUIC/UDP/IP with vpp_echo

vpp_echo performance testing tool[26] is a bespoke performance test application which utilizes the 'native HostStack APIs' to verify performance and correct handling of connection/stream events with uni-directional and bi-directional streams of data.

Because iperf3 does not support the QUIC transport protocol, vpp_echo is used for measuring the maximum attainable goodput of the VPP Host Stack connection utilizing the QUIC transport protocol across two instances of VPP running on separate DUT nodes. The QUIC transport protocol supports multiple streams per connection and test cases utilize different combinations of QUIC connections and number of streams per connection.

The test configuration is as follows:

```
       DUT1              Network               DUT2
[ vpp_echo-client -> VPP1 ]=======[ VPP2 -> vpp_echo-server]
                     N-streams/connection
```

where,

1. vpp_echo server attaches to VPP2 and LISTENs on VPP2:TCP port 1234.
2. vpp_echo client creates one or more connections to VPP1 and opens one or more stream per connection to VPP2:TCP port 1234.

[25] https://github.com/esnet/iperf
[26] https://wiki.fd.io/view/VPP/HostStack#External_Echo_Server.2FClient_.28vpp_echo.29

3. vpp_echo client transmits a uni-directional stream as fast as the VPP Host Stack allows to the vpp_echo server for the test duration.
4. At the end of the test the vpp_echo client emits the goodput measurements for all streams and the sum of all streams.

Test cases include

1. 1 QUIC Connection with 1 Stream
2. 1 QUIC connection with 10 Streams
3. 10 QUIC connetions with 1 Stream
4. 10 QUIC connections with 10 Streams

with stream sizes to provide reasonable test durations. The VPP Host Stack QUIC transport is configured to utilize the picotls encryption library. In the future, tests utilizing addtional encryption algorithms will be added.

VSAP ab with nginx

VSAP (VPP Stack Acceleration Project)[27] aims to establish an industry user space application ecosystem based on the VPP hoststack. As a pre-requisite to adapting open source applications using VPP Communications Library to accelerate performance, the VSAP team has introduced baseline tests utilizing the LD_PRELOAD mechanism to capture baseline performance data.

AB (Apache HTTP server benchmarking tool)[28] is used for measuring the maximum connections-per-second and requests-per-second.

NGINX[29] is a popular open source HTTP server application. Because NGINX utilizes the POSIX socket interface APIs, the test configuration uses the LD_PRELOAD mechanism to connect NGINX to the VPP Hoststack using the VPP Communications Library (VCL) LD_PRELOAD library (libvcl_ldpreload.so).

In the future, a version of NGINX which has been modified to directly use the VCL application APIs will be added to determine the difference in performance of 'VCL Native' applications versus utilizing LD_PRELOAD which inherently has more overhead and other limitations.

The test configuration is as follows:

```
  TG     Network         DUT
[ AB ]=============[ VPP -> nginx ]
```

where,

1. nginx attaches to VPP and listens on TCP port 80
2. ab runs CPS and RPS tests with packets flowing from the Test Generator node, across 100G NICs, through VPP hoststack to NGINX.
3. At the end of the tests, the results are reported by AB.

[27] https://wiki.fd.io/view/VSAP
[28] https://httpd.apache.org/docs/2.4/programs/ab.html
[29] https://www.nginx.com/

1.5.15 Generic Segmentation Offload Tests

Overview

Generic Segmentation Offload (GSO) reduces per-packet processing overhead by enabling applications to pass a multi-packet buffer to (v)NIC and process a smaller number of large packets (e.g. frame size of 64 KB), instead of processing higher numbers of small packets (e.g. frame size of 1500 B), thus reducing per-packet overhead.

CSIT-2210 introduced GSO tests for VPP vhostuser and tapv2 interfaces. All tests cases use iPerf3 client and server applications running TCP/IP as a traffic generator. For performance comparison the same tests are run without GSO enabled.

GSO Test Topologies

Two VPP GSO test topologies are implemented in CSIT-2210:

1. iPerfC_GSOvirtio_LinuxVM — GSOvhost_VPP_GSOvhost — iPerfS_GSOvirtio_LinuxVM
    - Tests VPP GSO on vhostuser interfaces and interaction with Linux virtio with GSO enabled.
1. iPerfC_GSOtap_LinuxNspace — GSOtapv2_VPP_GSOtapv2 — iPerfS_GSOtap_LinuxNspace
    - Tests VPP GSO on tapv2 interfaces and interaction with Linux tap with GSO enabled.

Common configuration:

- iPerfC (client) and iPerfS (server) run in TCP/IP mode without upper bandwidth limit.
- Trial duration is set to 30 sec.
- iPerfC, iPerfS and VPP run in the single SUT node.

VPP GSOtap Topology

VPP Configuration

VPP GSOtap tests in CSIT-2210 are executed without using hyperthreading. VPP worker runs on a single core. Multi-core tests are not executed. Each interface belongs to separate namespace. Following core pinning scheme is used:

- 1t1c (rxq=1, rx_qsz=4096, tx_qsz=4096)
    - system isolated: 0,28,56,84
    - vpp mt: 1
    - vpp wt: 2
    - vhost: 3-5
    - iperf-s: 6
    - iperf-c: 7

iPerf3 Server Configuration

iPerf3 version used 3.7

```
$ sudo -E -S ip netns exec tap1_namespace iperf3 \
    --server --daemon --pidfile /tmp/iperf3_server.pid --logfile /tmp/iperf3.log --port␣
↪5201 --affinity <X>
```

For the full iPerf3 reference please see: iPerf3 docs[30].

iPerf3 Client Configuration

iPerf3 version used 3.7

```
$ sudo -E -S ip netns exec tap1_namespace iperf3 \
    --client 2.2.2.2 --bind 1.1.1.1 --port 5201 --parallel <Y> --time 30.0 --affinity <X>␣
↪--zerocopy
```

For the full iPerf3 reference please see: iPerf3 docs[31].

VPP GSOvhost Topology

VPP Configuration

VPP GSOvhost tests in CSIT-2210 are executed without using hyperthreading. VPP worker runs on a single core. Multi-core tests are not executed. Following core pinning scheme is used:

- 1t1c (rxq=1, rx_qsz=1024, tx_qsz=1024) - system isolated: 0,28,56,84 - vpp mt: 1 - vpp wt: 2 - vm-iperf-s: 3,4,5,6,7 - vm-iperf-c: 8,9,10,11,12 - iperf-s: 1 - iperf-c: 1

iPerf3 Server Configuration

iPerf3 version used 3.7

```
$ sudo iperf3 \
    --server --daemon --pidfile /tmp/iperf3_server.pid --logfile /tmp/iperf3.log --port␣
↪5201 --affinity X
```

For the full iPerf3 reference please see: iPerf3 docs[32].

iPerf3 Client Configuration

iPerf3 version used 3.7

```
$ sudo iperf3 \
    --client 2.2.2.2 --bind 1.1.1.1 --port 5201 --parallel <Y> --time 30.0 --affinity X --
↪zerocopy
```

For the full iPerf3 reference please see: iPerf3 docs[33].

[30] https://github.com/esnet/iperf/blob/master/docs/invoking.rst
[31] https://github.com/esnet/iperf/blob/master/docs/invoking.rst
[32] https://github.com/esnet/iperf/blob/master/docs/invoking.rst
[33] https://github.com/esnet/iperf/blob/master/docs/invoking.rst

1.5.16 Reconfiguration Tests

Important: **DISCLAIMER**: Described reconf test methodology is experimental, and subject to change following consultation within csit-dev, vpp-dev and user communities. Current test results should be treated as indicative.

Overview

Reconf tests are designed to measure the impact of VPP re-configuration on data plane traffic. While VPP takes some measures against the traffic being entirely stopped for a prolonged time, the immediate forwarding rate varies during the re-configuration, as some configurations steps need the active dataplane worker threads to be stopped temporarily.

As the usual methods of measuring throughput need multiple trial measurements with somewhat long durations, and the re-configuration process can also be long, finding an offered load which would result in zero loss during the re-configuration process would be time-consuming.

Instead, reconf tests first find a througput value (lower bound for NDR) without re-configuration, and then maintain that ofered load during re-configuration. The measured loss count is then assumed to be caused by the re-configuration process. The result published by reconf tests is the effective blocked time, that is the loss count divided by the offered load.

Current Implementation

Each reconf suite is based on a similar MLRsearch performance suite.

MLRsearch parameters are changed to speed up the throughput discovery. For example, PDR is not searched for, and the final trial duration is shorter.

The MLRsearch suite has to contain a configuration parameter that can be scaled up, e.g. number of tunnels or number of service chains. Currently, only increasing the scale is supported as the re-configuration operation. In future, scale decrease or other operations can be implemented.

The traffic profile is not changed, so the traffic present is processed only by the smaller scale configuration. The added tunnels / chains are not targetted by the traffic.

For the re-configuration, the same Robot Framework and Python libraries are used, as were used in the initial configuration, with the exception of the final calls that do not interact with VPP (e.g. starting virtual machines) being skipped to reduce the test overall duration.

Discussion

Robot Framework introduces a certain overhead, which may affect timing of individual VPP API calls, which in turn may affect the number of packets lost.

The exact calls executed may contain unnecessary info dumps, repeated commands, or commands which change a value that do not need to be changed (e.g. MTU). Thus, implementation details are affecting the results, even if their effect on the corresponding MLRsearch suite is negligible.

The lower bound for NDR is the only value safe to be used when zero packets lost are expected without re-configuration. But different suites show different “jitter” in that value. For some suites, the lower bound is not tight, allowing full NIC buffers to drain quickly between worker pauses. For other suites, lower bound for NDR still has quite a large probability of non-zero packet loss even without re-configuration.

1.5.17 VPP Startup Settings

CSIT code manipulates a number of VPP settings in startup.conf for optimized performance. List of common settings applied to all tests and test dependent settings follows.

See VPP startup.conf[34] for a complete set and description of listed settings.

Common Settings

List of VPP startup.conf settings applied to all tests:

1. heap-size <value> - set separately for ip4, ip6, stats, main depending on scale tested.
2. no-tx-checksum-offload - disables UDP / TCP TX checksum offload in DPDK. Typically needed for use faster vector PMDs (together with no-multi-seg).
3. buffers-per-numa <value> - sets a number of memory buffers allocated to VPP per CPU socket. VPP default is 16384. Needs to be increased for scenarios with large number of interfaces and worker threads. To accommodate for scale tests, CSIT is setting it to the maximum possible value corresponding to the limit of DPDK memory mappings (currently 256). For Xeon Skylake platforms configured with 2MB hugepages and VPP data-size and buffer-size defaults (2048B and 2496B respectively), this results in value of 215040 (256 * 840 = 215040, 840 * 2496B buffers fit in 2MB hugepage).

Per Test Settings

List of vpp startup.conf settings applied dynamically per test:

1. corelist-workers <list_of_cores> - list of logical cores to run VPP worker data plane threads. Depends on HyperThreading and core per test configuration.
2. num-rx-queues <value> - depends on a number of VPP threads and NIC interfaces.
3. no-multi-seg - disables multi-segment buffers in DPDK, improves packet throughput, but disables Jumbo MTU support. Disabled for all tests apart from the ones that require Jumbo 9000B frame support.
4. UIO driver - depends on topology file definition.
5. QAT VFs - depends on NRThreads, each thread = 1QAT VFs.

1.5.18 KVM VMs vhost-user

QEMU is used for KVM VM vhost-user testing enviroment. By default, standard QEMU version is used, preinstalled from OS repositories (qemu-2.11.1 for Ubuntu 18.04). The path to the QEMU binary can be adjusted in *Constants.py*.

FD.io CSIT performance lab is testing VPP vhost-user with KVM VMs using following environment settings:

CSIT supports two types of VMs:

- **Image-VM**: used for all functional, VPP_device, and regular performance tests except NFV density tests.
- **Kernel-VM**: new VM type introduced for NFV density tests to provide greater in-VM application install flexibility and to further reduce test execution time by simpler VM lifecycle management.

[34] https://git.fd.io/vpp/tree/src/vpp/conf/startup.conf?h=stable/2210&id=07e0c05e698cf5ffd1e2d2de0296d1907519dc3d

Image-VM

CSIT can use a pre-created VM image. The path to the image can be adjusted in *Constants.py*. For convenience and full compatibility CSIT repository contains a set of scripts to prepare Built-root[35] based embedded Linux image with all the dependencies needed to run DPDK Testpmd, DPDK L3Fwd, Linux bridge or Linux IPv4 forwarding.

Built-root was chosen for a VM image to make it lightweight and with fast booting time to limit impact on tests duration.

In order to execute CSIT tests, VM image must have following software installed: qemu-guest-agent, sshd, bridge-utils, VirtIO support and DPDK Testpmd/L3fwd applications. Username/password for the VM must be `cisco/cisco` and `NOPASSWD` sudo access. The interface naming is based on the driver (management interface type is Intel E1000), all E1000 interfaces will be named `mgmt<n>` and all VirtIO interfaces will be named `virtio<n>`. In VM `/etc/init.d/qemu-guest-agent` must be set to `TRANSPORT=isa-serial:/dev/ttyS1` because ttyS0 is used by serial console and ttyS1 is dedicated for qemu-guest-agent in QEMU setup.

Kernel-VM

CSIT can use a kernel KVM image as a boot kernel, as an alternative to image VM. This option allows better configurability of what application is running in VM userspace. Using root9p filesystem allows mapping the host-OS filesystem as read only guest-OS filesystem.

Example of custom init script for the kernel-VM:

```
#!/bin/bash
mount -t sysfs -o "nodev,noexec,nosuid" sysfs /sys
mount -t proc -o "nodev,noexec,nosuid" proc /proc
mkdir /dev/pts
mkdir /dev/hugepages
mount -t devpts -o "rw,noexec,nosuid,gid=5,mode=0620" devpts /dev/pts || true
mount -t tmpfs -o "rw,noexec,nosuid,size=10%,mode=0755" tmpfs /run
mount -t tmpfs -o "rw,noexec,nosuid,size=10%,mode=0755" tmpfs /tmp
mount -t hugetlbfs -o "rw,relatime,pagesize=2M" hugetlbfs /dev/hugepages
echo 0000:00:06.0 > /sys/bus/pci/devices/0000:00:06.0/driver/unbind
echo 0000:00:07.0 > /sys/bus/pci/devices/0000:00:07.0/driver/unbind
echo vfio-pci > /sys/bus/pci/devices/0000:00:06.0/driver_override
echo vfio-pci > /sys/bus/pci/devices/0000:00:07.0/driver_override
echo 0000:00:06.0 > /sys/bus/pci/drivers/vfio-pci/bind
echo 0000:00:07.0 > /sys/bus/pci/drivers/vfio-pci/bind
$vnf_bin
poweroff -f
```

QemuUtils library during runtime replaces the `$vnf_bin` variable by the path to NF binary and its parameters. This allows CSIT to run any application installed on host OS, for example the same version of VPP as running on the host-OS.

Kernel-VM image must be available in the host filesystem as a prerequisite. The path to kernel-VM image is defined in *Constants.py*.

[35] https://buildroot.org/

1.5.19 Container Orchestration in CSIT

Overview

Linux Containers

Linux Containers is an OS-level virtualization method for running multiple isolated Linux systems (containers) on a compute host using a single Linux kernel. Containers rely on Linux kernel cgroups functionality for controlling usage of shared system resources (i.e. CPU, memory, block I/O, network) and for namespace isolation. The latter enables complete isolation of applications' view of operating environment, including process trees, networking, user IDs and mounted file systems.

LXC (Linux Containers) combine kernel's cgroups and support for isolated namespaces to provide an isolated environment for applications. Docker does use LXC as one of its execution drivers, enabling image management and providing deployment services. More information in [lxc], [lxcnamespace] and [stgraber].

Linux containers can be of two kinds: privileged containers and unprivileged containers.

Unprivileged Containers

Running unprivileged containers is the safest way to run containers in a production environment. From LXC 1.0 one can start a full system container entirely as a user, allowing to map a range of UIDs on the host into a namespace inside of which a user with UID 0 can exist again. In other words an unprivileged container does mask the userid from the host, making it impossible to gain a root access on the host even if a user gets root in a container. With unprivileged containers, non-root users can create containers and will appear in the container as the root, but will appear as userid <non-zero> on the host. Unprivileged containers are also better suited to supporting multi-tenancy operating environments. More information in [lxcsecurity] and [stgraber].

Privileged Containers

Privileged containers do not mask UIDs, and container UID 0 is mapped to the host UID 0. Security and isolation is controlled by a good configuration of cgroup access, extensive AppArmor profile preventing the known attacks as well as container capabilities and SELinux. Here a list of applicable security control mechanisms:

- Capabilities - keep (whitelist) or drop (blacklist) Linux capabilities, [capabilities].
- Control groups - cgroups, resource bean counting, resource quotas, access restrictions, [cgroup1], [cgroup2].
- AppArmor - apparmor profiles aim to prevent any of the known ways of escaping a container or cause harm to the host, [apparmor].
- SELinux - Security Enhanced Linux is a Linux kernel security module that provides similar function to AppArmor, supporting access control security policies including United States Department of Defense-style mandatory access controls. Mandatory access controls allow an administrator of a system to define how applications and users can access different resources such as files, devices, networks and inter- process communication, [selinux].
- Seccomp - secure computing mode, enables filtering of system calls, [seccomp].

More information in [lxcsecurity] and [lxcsecfeatures].

Linux Containers in CSIT

CSIT is using Privileged Containers as the `sysfs` is mounted with RW access. Sysfs is required to be mounted as RW due to VPP accessing **`/sys/bus/pci/drivers/uio_pci_generic/unbind`**. This is not the case of unprivileged containers where `sysfs` is mounted as read-only.

Orchestrating Container Lifecycle Events

Following Linux container lifecycle events need to be addressed by an orchestration system:

1. Acquire - acquiring/downloading existing container images via **`docker pull`** or **`lxc-create -t download`**.
2. Build - building a container image from scratch or another container image via **`docker build <dockerfile/composefile>`** or customizing LXC templates in GitHub[36].
3. (Re-)Create - creating a running instance of a container application from anew, or re-creating one that failed. A.k.a. (re-)deploy via **`docker run`** or **`lxc-start`**
4. Execute - execute system operations within the container by attaching to running container. THis is done by **`lxc-attach`** or **`docker exec`**
5. Distribute - distributing pre-built container images to the compute nodes. Currently not implemented in CSIT.

Container Orchestration Systems Used in CSIT

Current CSIT testing framework integrates following Linux container orchestration mechanisms:

- LXC/Docker for complete VPP container lifecycle control.

LXC

LXC is the well-known and heavily tested low-level Linux container runtime [lxcsource], that provides a userspace interface for the Linux kernel containment features. With a powerful API and simple tools, LXC enables Linux users to easily create and manage system or application containers. LXC uses following kernel features to contain processes:

- Kernel namespaces: ipc, uts, mount, pid, network and user.
- AppArmor and SELinux security profiles.
- Seccomp policies.
- Chroot.
- Cgroups.

CSIT uses LXC runtime and LXC usertools to test VPP data plane performance in a range of virtual networking topologies.

Known Issues

- Current CSIT restriction: only single instance of lxc runtime due to the cgroup policies used in CSIT. There is plan to add the capability into code to create cgroups per container instance to address this issue. This sort of functionality is better supported in LXC 2.1 but can be done is current version as well.
- CSIT code is currently using cgroup to control the range of CPU cores the LXC container runs on. VPP thread pinning is defined vpp startup.conf.

[36] https://github.com/lxc/lxc/tree/master/templates

Docker

Docker builds on top of Linux kernel containment features, and offers a high-level tool for wrapping the processes, maintaining and executing them in containers [docker]. Currently it is using *runc*, a CLI tool for spawning and running containers according to the OCI specification[37].

A Docker container image is a lightweight, stand-alone, executable package that includes everything needed to run the container: code, runtime, system tools, system libraries, settings.

CSIT uses Docker to manage the maintenance and execution of containerized applications used in CSIT performance tests.

- Data plane thread pinning to CPU cores - Docker CLI and/or Docker configuration file controls the range of CPU cores the Docker image must run on. VPP thread pinning defined vpp startup.conf.

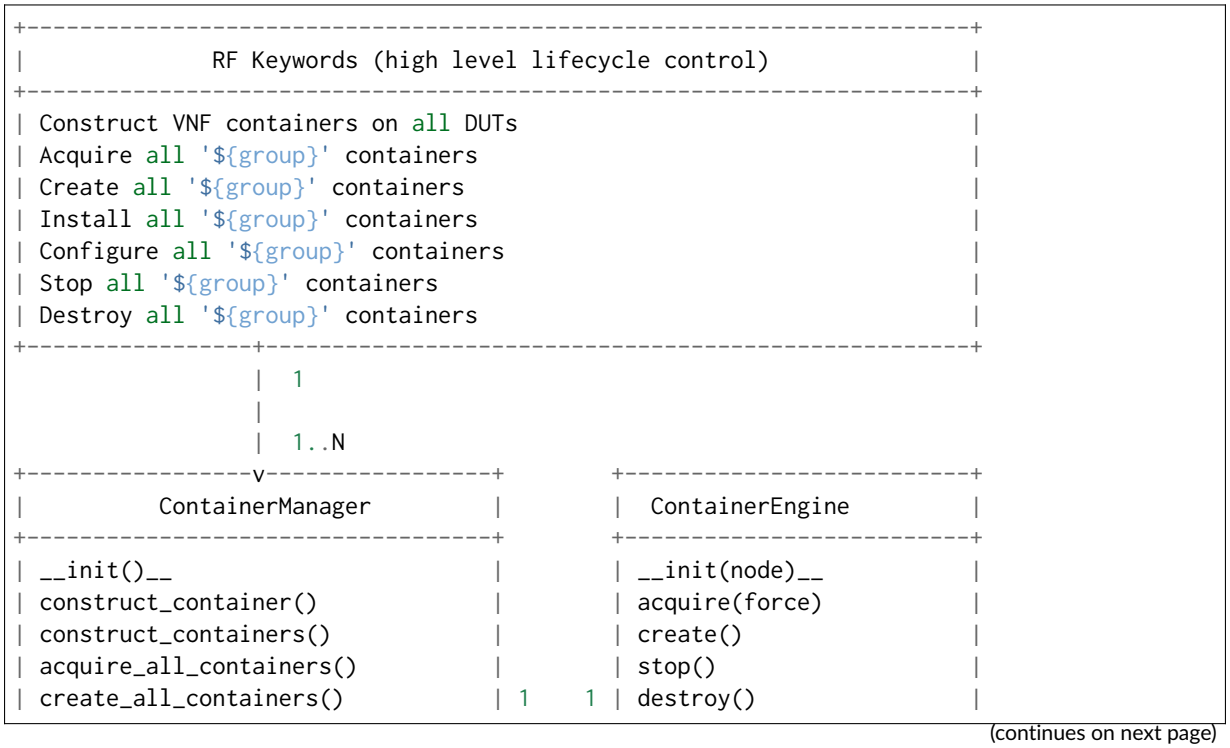
Implementation

CSIT container orchestration is implemented in CSIT Level-1 keyword Python libraries following the Builder design pattern. Builder design pattern separates the construction of a complex object from its representation, so that the same construction process can create different representations e.g. LXC, Docker, other.

CSIT Robot Framework keywords are then responsible for higher level lifecycle control of of the named container groups. One can have multiple named groups, with 1..N containers in a group performing different role/functionality e.g. NFs, Switch, Kafka bus, ETCD datastore, etc. ContainerManager class acts as a Director and uses ContainerEngine class that encapsulate container control.

Current CSIT implementation is illustrated using UML Class diagram:

1. Acquire
2. Build
3. (Re-)Create
4. Execute



(continues on next page)

[37] https://www.opencontainers.org/

(continued from previous page)

```
| execute_on_container()            <>-------| info()                   |
| execute_on_all_containers()       |        | execute(command)         |
| install_vpp_in_all_containers()   |        | system_info()            |
| configure_vpp_in_all_containers() |        | install_supervisor()     |
| stop_all_containers()             |        | install_vpp()            |
| destroy_all_containers()          |        | restart_vpp()            |
+-----------------------------------+        | create_vpp_exec_config() |
                                             | create_vpp_startup_config|
                                             | is_container_running()   |
                                             | is_container_present()   |
                                             | _configure_cgroup()      |
                                             +------------^-------------+
                                                          |
                                                          |
                                                          |
                                                +---------+---------+
                                                |                   |
                                          +-----+------+      +-----+------+
                                          |    LXC     |      |   Docker   |
                                          +------------+      +------------+
                                          | (inherited) |      | (inherited) |
                                          +-----+------+      +-----+------+
                                                |                   |
                                                +---------+---------+
                                                          |
                                                          | constructs
                                                          |
                                                +---------v---------+
                                                |     Container     |
                                                +-------------------+
                                                | __getattr__(a)    |
                                                | __setattr__(a, v) |
                                                +-------------------+
```

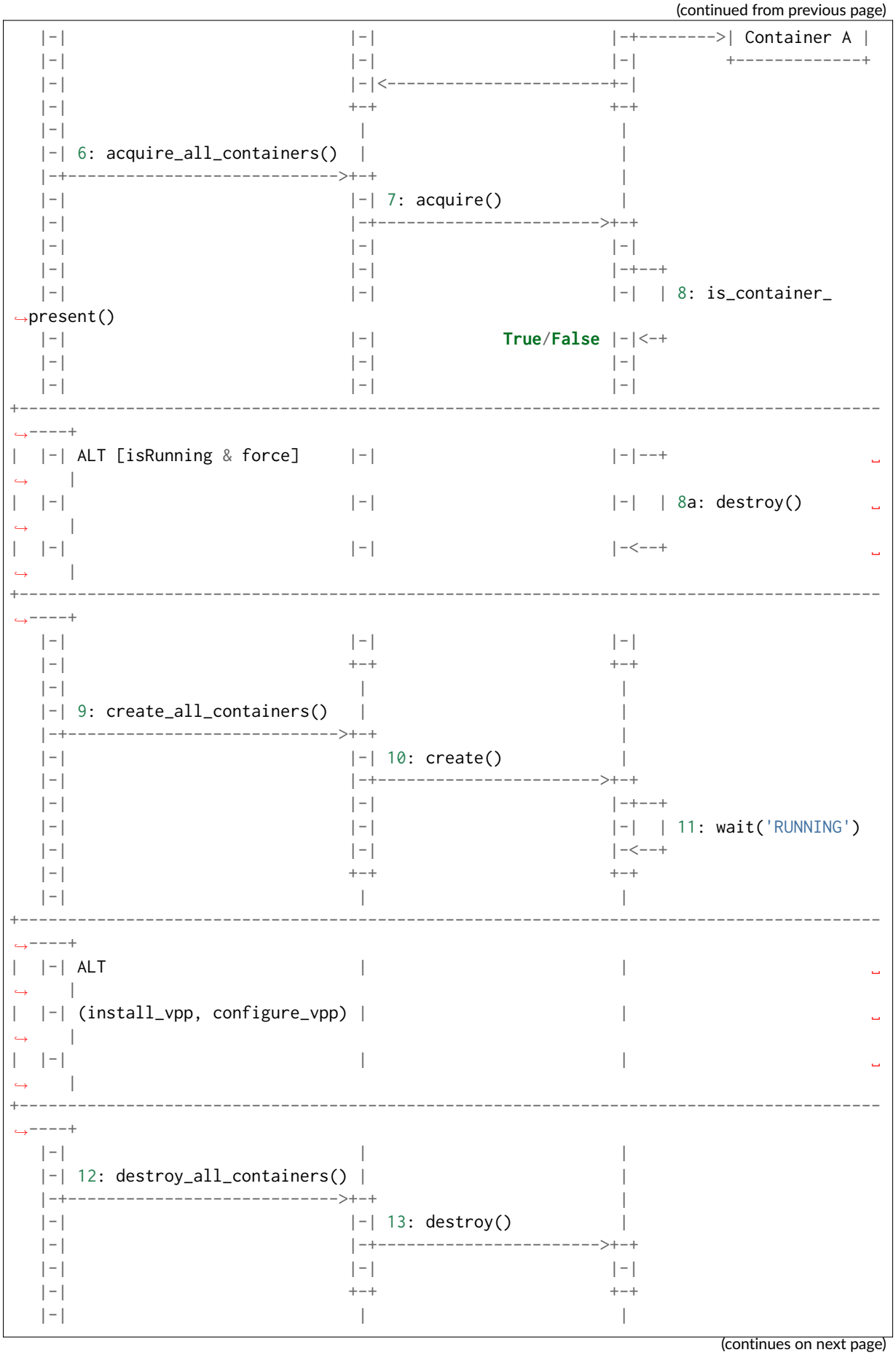
Sequentional diagram that illustrates the creation of a single container.

```
Legend:
   e  = engine [Docker|LXC]
   .. = kwargs (variable number of keyword argument)

+-------+                 +------------------+       +-----------------+
| RF KW |                 | ContainerManager |       | ContainerEngine |
+---+---+                 +--------+---------+       +--------+--------+
    |                              |                          |
    |  1: new ContainerManager(e)  |                          |
   +-+----------------------------->+-+                       |
   |-|                              |-| 2: new ContainerEngine  |
   |-|                              |-+------------------------>+-+
   |-|                              |-|                         |-|
   |-|                              +-+                         +-+
   |-|                               |                           |
   |-| 3: construct_container(..)    |                           |
   |-+----------------------------->+-+                         |
   |-|                              |-| 4: init()               |
   |-|                              |-+------------------------>+-+
   |-|                              |-|                         |-| 5: new  +-------------+
```

(continues on next page)

(continued from previous page)



(continues on next page)

(continued from previous page)

```
  +++                           |                       |
   |                            |                       |
   +                            +                       +
```

Container Data Structure

Container is represented in Python L1 library as a separate Class with instance variables and no methods except overriden `__getattr__` and `__setattr__`. Instance variables are assigned to container dynamically during the `construct_container(**kwargs)` call and are passed down from the RF keyword.

There is no parameters check functionality. Passing the correct arguments is a responsibility of the caller.

Examples

This section contains a high-level example of multiple initialization steps via ContainerManager; taken from an actual CSIT code, but with non-code lines (comments, Documentation) removed for brevity.

:

```
| Start containers for test
| | [Arguments] | ${dut}=${None} | ${nf_chains}=${1} | ${nf_nodes}=${1}
| | ... | ${auto_scale}=${True} | ${pinning}=${True}
| |
| | Set Test Variable | @{container_groups} | @{EMPTY}
| | Set Test Variable | ${container_group} | CNF
| | Set Test Variable | ${nf_nodes}
| | Import Library | resources.libraries.python.ContainerUtils.ContainerManager
| | ... | engine=${container_engine} | WITH NAME | ${container_group}
| | Construct chains of containers
| | ... | dut=${dut} | nf_chains=${nf_chains} | nf_nodes=${nf_nodes}
| | ... | auto_scale=${auto_scale} | pinning=${pinning}
| | Acquire all '${container_group}' containers
| | Create all '${container_group}' containers
| | Configure VPP in all '${container_group}' containers
| | Start VPP in all '${container_group}' containers
| | Append To List | ${container_groups} | ${container_group}
| | Save VPP PIDs
```

Kubernetes

For the future use, Kubernetes [k8sdoc] is implemented as separate library `KubernetesUtils.py`, with a class with the same name. This utility provides an API for L2 Robot Keywords to control `kubectl` installed on each of DUTs. One time initialization script, `resources/libraries/bash/k8s_setup.sh` does reset/init kubectl, and initializes the `csit` namespace. CSIT namespace is required to not to interfere with existing setups and it further simplifies apply/get/delete Pod/ConfigMap operations on SUTs.

Kubernetes utility is based on YAML templates to avoid crafting the huge YAML configuration files, what would lower the readability of code and requires complicated algorithms.

Two types of YAML templates are defined:

- Static - do not change between deployments, that is infrastructure containers like Kafka, Calico, ETCD.
- Dynamic - per test suite/case topology YAML files.

Making own python wrapper library of `kubectl` instead of using the official Python package allows to control and deploy environment over the SSH library without the need of using isolated driver running on each of DUTs.

Tested Topologies

Listed CSIT container networking test topologies are defined with DUT containerized VPP switch forwarding packets between NF containers. Each NF container runs their own instance of VPP in L2XC configuration.

Following container networking topologies are tested in CSIT-2210:

- LXC topologies:
  - eth-l2xcbase-eth-2memif-1lxc.
  - eth-l2bdbasemaclrn-eth-2memif-1lxc.
- Docker topologies:
  - eth-l2xcbase-eth-2memif-1docker.
  - eth-l2xcbase-eth-1memif-1docker

References

1.5.20 LXC/DRC Container Memif

CSIT includes tests taking advantage of VPP memif virtual interface (shared memory interface) to interconnect VPP running in Containers. VPP vswitch instance runs in bare-metal user-mode handling NIC interfaces and connecting over memif (Slave side) to VPPs running in Linux Container (LXC) or in Docker Container (DRC) configured with memif (Master side). LXCs and DRCs run in a priviliged mode with VPP data plane worker threads pinned to dedicated physical CPU cores per usual CSIT practice. All VPP instances run the same version of software. This test topology is equivalent to existing tests with vhost-user and VMs as described earlier in *Logical Topologies* (page 79).

In addition to above vswitch tests, a single memif interface test is executed. It runs in a simple topology of two VPP container instances connected over memif interface in order to verify standalone memif interface performance.

More information about CSIT LXC and DRC setup and control is available in *Container Orchestration in CSIT* (page 56).

1.5.21 NFV Service Density

Network Function Virtualization (NFV) service density tests focus on measuring total per server throughput at varied NFV service "packing" densities with vswitch providing host dataplane. The goal is to compare and contrast performance of a shared vswitch for different network topologies and virtualization technologies, and their impact on vswitch performance and efficiency in a range of NFV service configurations.

Each NFV service instance consists of a set of Network Functions (NFs), running in VMs (VNFs) or in Containers (CNFs), that are connected into a virtual network topology using VPP vswitch running in Linux user-mode. Multiple service instances share the vswitch that in turn provides per service chain forwarding context(s). In order to provide a most complete picture, each network topology and service configuration is tested in different service density setups by varying two parameters:

- Number of service instances (e.g. 1, 2, 4, 6, 8, 10).
- Number of NFs per service instance (e.g. 1, 2, 4, 6, 8, 10).

Implementation of NFV service density tests in CSIT-2210 is using two NF applications:

- VNF: VPP of the same version as vswitch running in KVM VM, configured with /8 IPv4 prefix routing.
- CNF: VPP of the same version as vswitch running in Docker Container, configured with /8 IPv4 prefix routing.

Tests are designed such that in all tested cases VPP vswitch is the most stressed application, as for each flow vswitch is processing each packet multiple times, whereas VNFs and CNFs process each packets only once. To that end, all VNFs and CNFs are allocated enough resources to not become a bottleneck.

Service Configurations

Following NFV network topologies and configurations are tested:

- VNF Service Chains (VSC) with L2 vswitch
  - *Network Topology*: Sets of VNFs dual-homed to VPP vswitch over virtio-vhost links. Each set belongs to separate service instance.
  - *Network Configuration*: VPP L2 bridge-domain contexts form logical service chains of VNF sets and connect each chain to physical interfaces.
- CNF Service Chains (CSC) with L2 vswitch
  - *Network Topology*: Sets of CNFs dual-homed to VPP vswitch over memif links. Each set belongs to separate service instance.
  - *Network Configuration*: VPP L2 bridge-domain contexts form logical service chains of CNF sets and connect each chain to physical interfaces.
- CNF Service Pipelines (CSP) with L2 vswitch
  - *Network Topology*: Sets of CNFs connected into pipelines over a series of memif links, with edge CNFs single-homed to VPP vswitch over memif links. Each set belongs to separate service instance.
  - *Network Configuration*: VPP L2 bridge-domain contexts connect each CNF pipeline to physical interfaces.

Thread-to-Core Mapping

CSIT defines specific ratios for mapping software threads of vswitch and VNFs/CNFs to physical cores, with separate ratios defined for main control threads and data-plane threads.

In CSIT-2210 NFV service density tests run on Intel Xeon testbeds with Intel Hyper-Threading enabled, so each physical core is associated with a pair of sibling logical cores corresponding to the hyper-threads.

CSIT-2210 executes tests with the following software thread to physical core mapping ratios:

- vSwitch
  - Data-plane on single core
    - (main:core) = (1:1) => 1mt1c - 1 main thread on 1 core.
    - (data:core) = (1:1) => 2dt1c - 2 Data-plane Threads on 1 Core.
  - Data-plane on two cores
    - (main:core) = (1:1) => 1mt1c - 1 Main Thread on 1 Core.
    - (data:core) = (1:2) => 4dt2c - 4 Data-plane Threads on 2 Cores.
- VNF and CNF
  - Data-plane on single core

        * (main:core) = (2:1) => 2mt1c - 2 Main Threads on 1 Core, 1 Thread per NF, core shared between two NFs.
        * (data:core) = (1:1) => 2dt1c - 2 Data-plane Threads on 1 Core per NF.
    - Data-plane on single logical core (Two NFs per physical core)
        * (main:core) = (2:1) => 2mt1c - 2 Main Threads on 1 Core, 1 Thread per NF, core shared between two NFs.
        * (data:core) = (2:1) => 2dt1c - 2 Data-plane Threads on 1 Core, 1 Thread per NF, core shared between two NFs.

Maximum tested service densities are limited by a number of physical cores per NUMA. CSIT-2210 allocates cores within NUMA0. Support for multi NUMA tests is to be added in future release.

1.5.22 VPP_Device Functional

CSIT-2210 includes VPP_Device test environment for functional VPP device tests integrated into LFN CI/CD infrastructure. VPP_Device tests run on 1-Node testbeds (1n-skx, 1n-arm) and rely on Linux SRIOV Virtual Function (VF), dot1q VLAN tagging and external loopback cables to facilitate packet passing over external physical links. Initial focus is on few baseline tests. New device tests can be added by small edits to existing CSIT Performance (2-node) test. RF test definition code stays unchanged with the exception of traffic generator related L2 KWs.

1.5.23 Suite Generation

CSIT uses robot suite files to define tests. However, not all suite files available for Jenkins jobs (or manually started bootstrap scripts) are present in CSIT git repository. They are generated only when needed.

Autogen Library

There is a code generation layer implemented as Python library called “autogen”, called by various bash scripts.

It generates the full extent of CSIT suites, using the ones in git as templates.

Sources

The generated suites (and their contents) are affected by multiple information sources, listed below.

Git Suites

The suites present in git repository act as templates for generating suites. One of autogen design principles is that any template suite should also act as a full suite (no placeholders).

In practice, autogen always re-creates the template suite with exactly the same content, it is one of checks that autogen works correctly.

Regenerate Script

Not all suites present in CSIT git repository act as template for autogen. The distinction is on per-directory level. Directories with regenerate_testcases.py script usually consider all suites as templates (unless possibly not included by the glob patten in the script).

The script also specifies minimal frame size, indirectly, by specifying protocol (protocol "ip4" is the default, leading to 64B frame size).

Constants

Values in Constants.py are taken into consideration when generating suites. The values are mostly related to different NIC models and NIC drivers.

Python Code

Python code in resources/libraries/python/autogen contains several other information sources.

Testcase Templates

The test case part of template suite is ignored, test case lines are created according to text templates in Testcase.py file.

Testcase Argument Lists

Each testcase template has different number of "arguments", e.g. values to put into various placeholders. Different test types need different lists of the argument values, the lists are in regenerate_glob method in Regenerator.py file.

Iteration Over Values

Python code detects the test type (usually by substrings of suite file name), then iterates over different quantities based on type. For example, only ndrpdr suite templates generate other types (mrr and soak).

Hardcoded Exclusions

Some combinations of values are known not to work, so they are excluded. Examples: Density tests for too much CPUs; IMIX for ASTF.

Non-Sources

Some information sources are available in CSIT repository, but do not affect the suites generated by autogen.

Testbeds

Overall, no information visible in topology yaml files is taken into account by autogen.

Testbed Architecture

Historically, suite files are agnostic to testbed architecture, e.g. ICX or ALT.

Testbed Size

Historically, 2-node and 3-node suites have diferent names, and while most of the code is common, the differences are not always simple enough. Autogen treat 2-node and 3-node suites as independent templates.

TRex suites are intended for a 1-node circuit of otherwise 2-node or 3-node testbeds, so they support all 3 robot tags. They are also detected and treated differently by autogen, mainly because they need different testcase arguments (no CPU count). Autogen does nothing specifically related to the fact they should run only in testbeds/NICs with TG-TG line available.

Other Topology Info

Some bonding tests need two (parallel) links between DUTs. Autogen does not care, as suites are agnostic. Robot tag marks the difference, but the link presence is not explicitly checked.

Job specs

Information in job spec files depend on generated suites (not the other way). Autogen should generate more suites, as job spec is limited by time budget. More suites should be available for manually triggered verify jobs, so autogen covers that.

Bootstrap Scripts

Historically, bootstrap scripts perform some logic, perhaps adding exclusion options to Robot invocation (e.g. skipping testbed+NIC combinations for tests that need parallel links).

Once again, the logic here relies on what autogen generates, autogen does not look into bootstrap scripts.

1.5.24 Testing in AWS EC2

AWS Performance Testbeds

CSIT implements two virtual machine topology types running in AWS EC2:

- **2-Node Topology**: Consists of one EC2 instance as a System Under Test (SUT) and one EC2 instance acting as a Traffic Generator (TG), with both instances connected into a ring topology. Used for executing tests that require frame encapsulations supported by TG.
- **3-Node Topology**: Consists of two EC2 instances acting as a Systems Under Test (SUTs) and one EC2 instance acting as a Traffic Generator (TG), with all instances connected into a ring topology. Used for executing tests that require frame encapsulations not supported by TG e.g. certain overlay tunnel encapsulations and IPsec.

AWS EC2 Instances

CSIT is using AWS EC2 C5n instances as System Under Test and TG virtual machines. C5n instances got selected to take advantage of high network throughput and packet rate performance. C5n instances offer up to 100 Gbps network bandwidth and increased memory over comparable C5 instances. For more information, see Instance types[52].

C5n features:

- 3.0 GHz Intel Xeon Platinum (Skylake) processors with Intel AVX-512 instructions.
- Sustained all core Turbo frequency of up to 3.4GHz, and single core turbo frequency of up to 3.5 GHz.
- Requires HVM AMIs (Amazon Machine Images) that include drivers for ENA and NVMe. See *CSIT Amazon Machine Images* (page 68) for more information.
- Network bandwidth to up to 100 Gbps.
- Powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor.

Model	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)***	EBS Bandwidth (Mbps)
c5n.large	2	5.25	EBS-Only	Up to 25	Up to 4,750
c5n.xlarge	4	10.5	EBS-Only	Up to 25	Up to 4,750
c5n.2xlarge	8	21	EBS-Only	Up to 25	Up to 4,750
c5n.4xlarge	16	42	EBS-Only	Up to 25	4,750
c5n.9xlarge	36	96	EBS-Only	50	9,500
c5n.18xlarge	72	192	EBS-Only	100	19,000
c5n.metal	72	192	EBS-Only	100	19,000

CSIT is configured by default to use *c5n.4xlarge* in *eu-central-1* AWS region due to allocation stability issues with *c5n.9xlarge* in *eu-central-1* region.

AWS EC2 Networking

CSIT EC2 instances are equipped with AWS Elastic Network Adapter (ENA) supporting AWS enhanced networking. Enhanced networking uses single root I/O virtualization (SR-IOV) to provide high-performance networking capabilities. For more information, see Elastic Network Adapter[53].

For more information about the current advertised AWS ENA performance limits, see Computed optimized instances[54].

CSIT DUTs make use of AWS ENA DPDK driver supplied by AWS and specified in amzn drivers dpdk[55].

[52] https://aws.amazon.com/ec2/instance-types/
[53] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking-ena.html
[54] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/compute-optimized-instances.html
[55] https://github.com/amzn/amzn-drivers/tree/master/userspace/dpdk

CSIT Amazon Machine Images

An Amazon Machine Image (AMI) provides the information required to launch an instance. CSIT is using Amazon Elastic Block Store (EBS) where the root device for an instance launched from the AMI is a volume created from an Amazon EBS snapshot.

As the TG and SUT instances have slightly different software requirements, we are defining two AMIs for TG and SUT separately. AMI details examples:

- TG:
  - AMI Name: csit_c5n_ubuntu_focal_tg
  - Platform details: Linux/UNIX
  - Architecture: x86_64
  - Usage operation: RunInstances
  - Image Type: machine
  - Virtualization type: hvm
  - Description: CSIT TG image based on Ubuntu Focal
  - Root Device Name: /dev/sda1
  - Root Device Type: ebs
- SUT:
  - AMI Name: csit_c5n_ubuntu_focal_sut
  - Platform details: Linux/UNIX
  - Architecture: x86_64
  - Usage operation: RunInstances
  - Image Type: machine
  - Virtualization type: hvm
  - Description: CSIT SUT image based on Ubuntu Focal
  - Root Device Name: /dev/sda1
  - Root Device Type: ebs

Both TG and SUT AMIs are created manually before launching topology and are not part of automated scripts.

Building AMIs requires Hashicorp Packer with Amazon plugin installed.

For more information, see Amazon Machine Images[56].

AWS Deployments

CSIT performance testbed deployments in AWS rely on Infrastructure-as-a-C (IaaC) Terraform AWS providers. Terraform providers specified in CSIT interact with resources provided by AWS to orchestrate virtual environment for running CSIT performance tests.

[56] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

Compatibility

Software	OSS Version
Terraform	1.0.3 or newer
Vault	1.8.4 or newer

Requirements

- Required Modules and Providers
  - Terraform Registry aws[57].
  - Terraform Registry null[58].
  - Terraform Registry tls[59].
  - Terraform Registry vault[60].
- Required software
  - Vault[61] service available on specified ip/port.

Deployment Example

Following is an example of a Terraform deploy module[62] for a CSIT 2-Node testbed topology with AWS variables set to default values. A number of variables is also defined in a separate Terraform variable file[63].

```
module "deploy" {
  source = "./deploy"

  # Parameters starting with var. can be set using "TF_VAR_*" environment
  # variables or -var parameter when running "terraform apply", for default
  # values see ./variables.tf
  testbed_name          = var.testbed_name
  topology_name         = var.topology_name
  environment_name      = var.environment_name
  resources_name_prefix = var.resources_name_prefix

  # AWS general
  region        = var.region
  avail_zone    = var.avail_zone
  instance_type = var.instance_type
  ami_image_tg  = var.ami_image_tg
  ami_image_sut = var.ami_image_sut

  # AWS Network
  vpc_cidr_mgmt = "192.168.0.0/24"
  vpc_cidr_b    = "192.168.10.0/24"
  vpc_cidr_c    = "200.0.0.0/24"
  vpc_cidr_d    = "192.168.20.0/24"
```

(continues on next page)

[57] https://registry.terraform.io/providers/hashicorp/aws/latest
[58] https://registry.terraform.io/providers/hashicorp/null/latest
[59] https://registry.terraform.io/providers/hashicorp/tls
[60] https://registry.terraform.io/providers/hashicorp/vault
[61] https://releases.hashicorp.com/vault/
[62] https://git.fd.io/csit/tree/fdio.infra.terraform/2n_aws_c5n/main.tf
[63] https://git.fd.io/csit/tree/fdio.infra.terraform/2n_aws_c5n/variables.tf

(continued from previous page)

```
  tg_mgmt_ip   = "192.168.0.10"
  dut1_mgmt_ip = "192.168.0.11"

  tg_if1_ip   = "192.168.10.254"
  tg_if2_ip   = "192.168.20.254"
  dut1_if1_ip = "192.168.10.11"
  dut1_if2_ip = "192.168.20.11"

  trex_dummy_cidr_port_0 = "10.0.0.0/24"
  trex_dummy_cidr_port_1 = "20.0.0.0/24"

  # Ansible
  ansible_python_executable = "/usr/bin/python3"
  ansible_file_path         = "../../fdio.infra.ansible/site.yaml"
  ansible_topology_path     = "../../fdio.infra.ansible/cloud_topology.yaml"
  ansible_provision_pwd     = "Csit1234"

  # First run
  first_run_commands = [
    "sudo sed -i 's/^PasswordAuthentication/#PasswordAuthentication/' /etc/ssh/sshd_config
↪",
    "sudo systemctl restart sshd",
    "sudo useradd --create-home -s /bin/bash provisionuser",
    "echo 'provisionuser:Csit1234' | sudo chpasswd",
    "echo 'provisionuser ALL = (ALL) NOPASSWD: ALL' | sudo tee -a /etc/sudoers",
    "sudo useradd --create-home -s /bin/bash testuser",
    "echo 'testuser:Csit1234' | sudo chpasswd",
    "echo 'testuser ALL = (ALL) NOPASSWD: ALL' | sudo tee -a /etc/sudoers"
  ]
}
```

Secrets & Credentials

Set credentials manually

To set the credentials manually you first need to tell the module to not fetch credentials from Vault. To do that, set *provider "aws" access_key* and *secret_key* to custom value or use credentials file as a source.

```
provider "aws" {
  region     = var.region
  access_key = data.vault_aws_access_credentials.creds.access_key
  secret_key = data.vault_aws_access_credentials.creds.secret_key
}
```

1.5.25 Root Cause Analysis

Per-patch performance tests

Updated for CSIT git commit id: 72b45cfe662107c8e1bb549df71ba51352a898ee.

A methodology similar to trending analysis is used for comparing performance before a DUT code change is merged. This can act as a verify job to disallow changes which would decrease performance without a good reason.

Existing jobs

VPP is the only project currently using such jobs. They are not started automatically, must be triggered on demand. They allow full tag expressions, but some tags are enforced (such as MRR).

There are jobs available for multiple types of testbeds, based on various processors. Their Gerrit triggers words are of the form "perftest-{node_arch}" where the node_arch combinations currently supported are: 2n-clx, 2n-dnv, 2n-tx2, 2n-zn2, 3n-dnv, 3n-tsh.

Test selection

Gerrit trigger line without any additional arguments selects a small set of test cases to run. If additional arguments are added to the Gerrit trigger, they are treated as Robot tag expressions to select tests to run. While very flexible, this method of test selection also allows the user to accidentally select too high number of tests, blocking the testbed for days.

What follows is a list of explanations and recommendations to help users to select the minimal set of tests cases.

Verify cycles

When Gerrit schedules multiple jobs to run for the same patch set, it waits until all runs are complete. While it is waiting, it is possible to trigger more jobs (adding runs to the set Gerrit is waiting for), but it is not possible to trigger more runs for the same job, until Gerrit is done waiting. After Gerrit is done waiting, it becames possible to trigger the same job again.

Example. User triggers one set of tests on 2n-icx and immediately also triggers other set of tests on 3n-icx. Then the user notices 2n-icx run end early because of a typo in tag expression. When the user tries to re-trigger 2n-icx (with fixed tag expression), that comment gets ignored by Jenkins. Only when 3n-icx job finishes, the user can trigger 2n-icx.

One comment many jobs

In the past, the CSIT code which parses for perftest trigger comments was buggy, which lead to bad behavior (as in selection all performance test, because "perftest" is also a robot tag) when user included multiple perftest trigger words in the same comment.

The worst bugs were fixed since then, but it is still recommended to use just one trigger word per Gerrit comment, just to be safe.

Multiple test cases in run

While Robot supports OR operator, it does not support parentheses, so the OR operator is not very useful. It is recommended to use space instead of OR operator.

Example template: perftest-2n-icx {tag_expression_1} {tag_expression_2}

See below for more concrete examples.

Suite tags

Traditionally, CSIT maintains broad Robot tags that can be used to select tests, for details on existing tags, see CSIT Tags[64].

But it is not recommended to use them for test selection, as it is not that easy to determine how many test cases are selected.

The recommended way is to look into CSIT repository first, and locate a specific suite the user is interested in, and use its suite tag. For example, "ethip4-ip4base" is a suite tag selecting just one suite in CSIT git repository, avoiding all scale, container, and other simialr variants.

Note that CSIT uses "autogen" code generator, so the robot running in Jenkins has access to more suites than visible just by looking into CSIT git repository, so suite tag is not enough to select even the intended suite, and user still probably wants to narrow down to a single test case within a suite.

Fully specified tag expressions

Here is one template to select a single test case: {test_type}AND{nic_model}AND{nic_driver}AND{cores}AND{frame_size}A
where the variables are all lower case (so AND operator stands out).

Currently only one test type is supported by the performance comparison jobs: "mrr". The nic_driver options depend on nic_model. For Intel cards "drv_avf" (AVF plugin) and "drv_vfio_pci" (DPDK plugin) are popular, for Mellanox "drv_rdma_core". Currently, the performance using "drv_af_xdp" is not reliable enough, so do not use it unless you are specifically testing for AF_XDP.

The most popular nic_model is "nic_intel-xxv710", but that is not available on all testbed types. It is safe to use "1c" for cores (unless you are suspection multi-core performance is affected differently) and "64b" for frame size ("78b" for ip6 and more for dot1q and other encapsulated traffic; "1518b" is popular for ipsec and other payload-bound tests).

As there are more test cases than CSIT can periodically test, it is possible to encounter an old test case that currently fails. To avoid that, you can look at "job spec" files we use for periodic testing, for example this one[65].

Shortening triggers

Advanced users may use the following tricks to avoid writing long trigger comments.

Robot supports glob matching, which can be used to select multiple suite tags at once.

Not specifying one of 6 parts of the recommended expression pattern will select all available options. For example not specifying nic_driver for nic_intel-xxv710 will select all 3 applicable drivers. You can use NOT operator to reject some options (e.g. NOTdrv_af_xdp), but beware, with NOT the order matters: tag1ANDtag2NOTtag3 is not the same as tag1NOTtag3ANDtag2, the latter is evaluated as tag1AND(NOT(tag3ANDtag2)).

[64] https://github.com/FDio/csit/blob/master/docs/tag_documentation.rst
[65] https://github.com/FDio/csit/blob/master/docs/job_specs/report_iterative/2n-icx/vpp-mrr-00.md

Beware when not specifying nic_model. As a precaution, CSIT code will insert the defailt NIC model for the tetsbed used. Example: Specifying drv_rdma_core without specifying nic_model will fail, as the default nic_model is nic_intel-xxv710 which does not support RDMA core driver.

Complete example

A user wants to test a VPP change which may affect load balance whith bonding. Searching tag documentation for "bonding" finds LBOND tag and its variants. Searching CSIT git repository (directory tests/) finds 8 suite files, all suited only for 3-node testbeds. All suites are using vhost, but differ by the forwarding app inside VM (DPDK or VPP), by the forwarding mode of VPP acting as host level vswitch (MAC learning or cross connect), and by the number of DUT1-DUT2 links available (1 or 2).

As not all NICs and testbeds offer enogh ports for 2 parallel DUT-DUT links, the user looks at testbed specifications[66] and finds that only xxv710 NIC on 3n-icx testbed matches the requirements. Quick look into the suites confirm the smallest frame size is 64 bytes (despite DOT1Q robot tag, as the encapsulation does not happen on TG-DUT links). It is ok to use just 1 physical core, as 3n-icx has hyperthreading enabled, so VPP vswitch will use 2 worker threads.

The user decides the vswitch forwarding mode is not important (so choses cross connect as that has less CPU overhead), but wants to test both NIC drivers (not AF_XDP), both apps in VM, and both 1 and 2 parallel links.

After shortening, this is the trigger comment fianlly used: perftest-3n-icx mrrANDnic_intel-x710AND1cAND64bAND?lbvpplacp-dot1q-l2xcbase-eth-2vhostvr1024-1vm*NOTdrv_af_xdp

Basic operation

The job builds VPP .deb packages for both the patch under test (called "current") and its parent patch (called "parent").

For each test (from a set defined by tag expression), both builds are subjected to several trial measurements (BMRR). Measured samples are grouped to "parent" sequence, followed by "current" sequence. The same Minimal Description Length algorithm as in trending is used to decide whether it is one big group, or two smaller gropus. If it is one group, a "normal" result is declared for the test. If it is two groups, and current average is less then parent average, the test is declared a regression. If it is two groups and current average is larger or equal, the test is declared a progression.

The whole job fails (giving -1) if some trial measurement failed, or if any test was declared a regression.

Temporary specifics

The Minimal Description Length analysis is performed by CSIT code equivalent to jumpavg-0.1.3 library available on PyPI.

In hopes of strengthening of signal (code performance) compared to noise (all other factors influencing the measured values), several workarounds are applied.

In contrast to trending, trial duration is set to 10 seconds, and only 5 samples are measured for each build. Both parameters are set in ci-management.

This decreases sensitivity to regressions, but also decreases probability of false positives.

[66] https://github.com/FDio/csit/tree/master/topologies/available

Console output

The following information as visible towards the end of Jenkins console output, repeated for each analyzed test.

The original 5 values are visible in order they were measured. The 5 values after processing are also visible in output, this time sorted by value (so people can see minimum and maximum).

The next output is difference of averages. It is the current average minus the parent average, expressed as percentage of the parent average.

The next three outputs contain the jumpavg representation of the two groups and a combined group. Here, "bits" is the description length; for "current" sequence it includes effect from "parent" average value (jumpavg-0.1.3 penalizes sequences with too close averages).

Next, a sentence describing which grouping description is shorter, and by how much bits. Finally, the test result classification is visible.

The algorithm does not track test case names, so test cases are indexed (from 0).

1.5.26 Trending Methodology

Overview

This document describes a high-level design of a system for continuous performance measuring, trending and change detection for FD.io VPP SW data plane (and other performance tests run within CSIT subproject).

Trend Analysis

All measured performance trend data is treated as time-series data that is modeled as a concatenation of groups, within each group the samples come (independently) from the same normal distribution (with some center and standard deviation).

Center of the normal distribution for the group (equal to population average) is called a trend for the group. All the analysis is based on finding the right partition into groups and comparing their trends.

Anomalies in graphs

In graphs, the start of the following group is marked as a regression (red circle) or progression (green circle), if the new trend is lower (or higher respectively) then the previous group's.

Implementation details

Partitioning into groups

While sometimes the samples within a group are far from being distributed normally, currently we do not have a better tractable model.

Here, "sample" should be the result of single trial measurement, with group boundaries set only at test run granularity. But in order to avoid detecting causes unrelated to VPP performance, the current presentation takes average of all trials within the run as the sample. Effectively, this acts as a single trial with aggregate duration.

Performance graphs show the run average as a dot (not all individual trial results).

The group boundaries are selected based on Minimum Description Length[67].

[67] https://en.wikipedia.org/wiki/Minimum_description_length

Minimum Description Length

Minimum Description Length[68] (MDL) is a particular formalization of Occam's razor[69] principle.

The general formulation mandates to evaluate a large set of models, but for anomaly detection purposes, it is useful to consider a smaller set of models, so that scoring and comparing them is easier.

For each candidate model, the data should be compressed losslessly, which includes model definitions, encoded model parameters, and the raw data encoded based on probabilities computed by the model. The model resulting in shortest compressed message is the "the" correct model.

For our model set (groups of normally distributed samples), we need to encode group length (which penalizes too many groups), group average (more on that later), group stdev and then all the samples.

Luckily, the "all the samples" part turns out to be quite easy to compute. If sample values are considered as coordinates in (multi-dimensional) Euclidean space, fixing stdev means the point with allowed coordinates lays on a sphere. Fixing average intersects the sphere with a (hyper)-plane, and Gaussian probability density on the resulting sphere is constant. So the only contribution is the "area" of the sphere, which only depends on the number of samples and stdev.

A somehow ambiguous part is in choosing which encoding is used for group size, average and stdev. Different encodings cause different biases to large or small values. In our implementation we have chosen probability density corresponding to uniform distribution (from zero to maximal sample value) for stdev and average of the first group, but for averages of subsequent groups we have chosen a distribution which discourages delimiting groups with averages close together.

Our implementation assumes that measurement precision is 1.0 pps. Thus it is slightly wrong for trial durations other than 1.0 seconds. Also, all the calculations assume 1.0 pps is totally negligible, compared to stdev value.

The group selection algorithm currently has no parameters, all the aforementioned encodings and handling of precision is hard-coded. In principle, every group selection is examined, and the one encodable with least amount of bits is selected. As the bit amount for a selection is just sum of bits for every group, finding the best selection takes number of comparisons quadratically increasing with the size of data, the overall time complexity being probably cubic.

The resulting group distribution looks good if samples are distributed normally enough within a group. But for obviously different distributions (for example bimodal distribution[70]) the groups tend to focus on less relevant factors (such as "outlier" density).

[68] https://en.wikipedia.org/wiki/Minimum_description_length
[69] https://en.wikipedia.org/wiki/Occam%27s_razor
[70] https://en.wikipedia.org/wiki/Bimodal_distribution

Common Patterns

When an anomaly is detected, it frequently falls into few known patterns, each having its typical behavior over time.

We are going to describe the behaviors, as they motivate our choice of trend compliance metrics.

Sample time and analysis time

But first we need to distinguish two roles time plays in analysis, so it is more clear which role we are referring to.

Sample time is the more obvious one. It is the time the sample is generated. It is the start time or the end time of the Jenkins job run, does not really matter which (parallel runs are disabled, and length of gap between samples does not affect metrics).

Analysis time is the time the current analysis is computed. Again, the exact time does not usually matter, what matters is how many later (and how fewer earlier) samples were considered in the computation.

For some patterns, it is usual for a previously reported anomaly to "vanish", or previously unseen anomaly to "appear late", as later samples change which partition into groups is more probable.

Dashboard and graphs are always showing the latest analysis time, the compliance metrics are using earlier sample time with the same latest analysis time.

Alerting e-mails use the latest analysis time at the time of sending, so the values reported there are likely to be different from the later analysis time results shown in dashboard and graphs.

Ordinary regression

The real performance changes from previously stable value into a new stable value.

For medium to high magnitude of the change, one run is enough for anomaly detection to mark this regression.

Ordinary progressions are detected in the same way.

Small regression

The real performance changes from previously stable value into a new stable value, but the difference is small.

For the anomaly detection algorithm, this change is harder to detect, depending on the standard deviation of the previous group.

If the new performance value stays stable, eventually the detection algorithm is able to detect this anomaly when there are enough samples around the new value.

If the difference is too small, it may remain undetected (as new performance change happens, or full history of samples is still not enough for the detection).

Small progressions have the same behavior.

Reverted regression

This pattern can have two different causes. We would like to distinguish them, but that is usually not possible to do just by looking at the measured values (and not telemetry).

In one cause, the real DUT performance has changed, but got restored immediately. In the other cause, no real performance change happened, just some temporary infrastructure issue has caused a wrong low value to be measured.

For small measured changes, this pattern may remain undetected. For medium and big measured changes, this is detected when the regression happens on just the last sample.

For big changes, the revert is also immediately detected as a subsequent progression. The trend is usually different from the previously stable trend (as the two population averages are not likely to be exactly equal), but the difference between the two trends is relatively small.

For medium changes, the detection algorithm may need several new samples to detect a progression (as it dislikes single sample groups), in the meantime reporting regressions (difference decreasing with analysis time), until it stabilizes the same way as for big changes (regression followed by progression, small difference between the old stable trend and last trend).

As it is very hard for a fault code or an infrastructure issue to increase performance, the opposite (temporary progression) almost never happens.

Summary

There is a trade-off between detecting small regressions and not reporting the same old regressions for a long time.

For people reading e-mails, a sudden regression with a big number of samples in the last group means this regression was hard for the algorithm to detect.

If there is a big regression with just one run in the last group, we are not sure if it is real, or just a temporary issue. It is useful to wait some time before starting an investigation.

With decreasing (absolute value of) difference, the number of expected runs increases. If there is not enough runs, we still cannot distinguish real regression from temporary regression just from the current metrics (although humans frequently can tell by looking at the graph).

When there is a regression or progression with just a small difference, it is probably an artifact of a temporary regression. Not worth examining, unless temporary regressions happen somewhat frequently.

It is not easy for the metrics to locate the previous stable value, especially if multiple anomalies happened in the last few weeks. It is good to compare last trend with long term trend maximum, as it highlights the difference between “now” and “what could be”. It is good to exclude last week from the trend maximum, as including the last week would hide all real progressions.

Trend Presentation

Failed tests

The Failed tests tables list the tests which failed during the last test run. Separate tables are generated for each testbed.

Regressions and progressions

These tables list tests which encountered a regression or progression during the specified time period, which is currently set to the last 21 days.

Trendline Graphs

Trendline graphs show measured per run averages of MRR values, NDR or PDR values, group average values, and detected anomalies. The graphs are constructed as follows:

- X-axis represents the date in the format MMDD.
- Y-axis represents run-average MRR value, NDR or PDR values in Mpps. For PDR tests also a graph with average latency at 50% PDR [us] is generated.
- Markers to indicate anomaly classification:
  - Regression - red circle.
  - Progression - green circle.
- The line shows average MRR value of each group.

In addition the graphs show dynamic labels while hovering over graph data points, presenting the CSIT build date, measured value, VPP reference, trend job build ID and the LF testbed ID.

1.6 Test Code Documentation

CSIT Performance Tests Documentation[71] contains detailed functional description and input parameters for each test case.

[71] https://s3-docs.fd.io/csit/rls2210/docs/index.html

CHAPTER

TWO

VPP PERFORMANCE

2.1 Overview

VPP performance test results are reported for a range of processors. For description of physical testbeds used for VPP performance tests please refer to *Performance Physical Testbeds* (page 4).

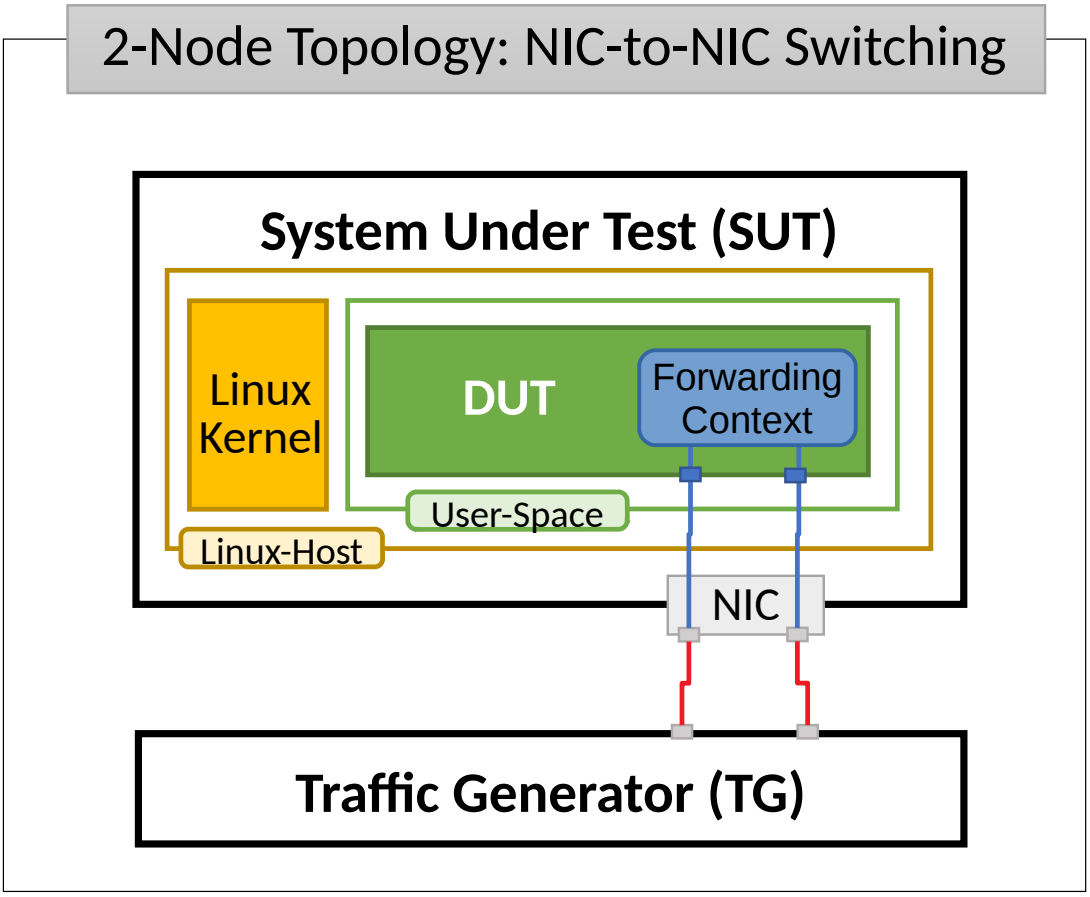
2.1.1 Logical Topologies

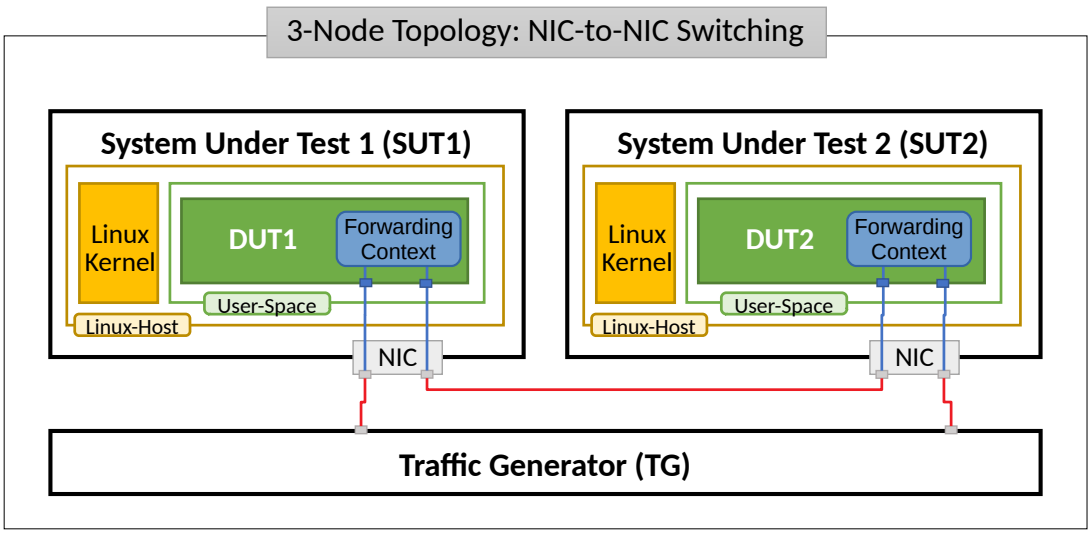
CSIT VPP performance tests are executed on physical testbeds described in *Performance Physical Testbeds* (page 4). Based on the packet path thru server SUTs, three distinct logical topology types are used for VPP DUT data plane testing:

1. NIC-to-NIC switching topologies.
2. VM service switching topologies.
3. Container service switching topologies.

NIC-to-NIC Switching

The simplest logical topology for software data plane application like VPP is NIC-to-NIC switching. Tested topologies for 2-Node and 3-Node testbeds are shown in figures below.





Server Systems Under Test (SUT) run VPP application in Linux user-mode as a Device Under Test (DUT). Server Traffic Generator (TG) runs T-Rex application. Physical connectivity between SUTs and TG is provided using different drivers and NIC models that need to be tested for performance (packet/bandwidth throughput and latency).

From SUT and DUT perspectives, all performance tests involve forwarding packets between two (or more) physical Ethernet ports (10GE, 25GE, 40GE, 100GE). In most cases both physical ports on SUT are located on the same NIC. The only exceptions are link bonding and 100GE tests. In the latter case only one port per NIC can be driven at linerate due to PCIe Gen3 x16 slot bandwidth limiations. 100GE NICs are not supported in PCIe Gen3 x8 slots.

Note that reported VPP DUT performance results are specific to the SUTs tested. SUTs with other processors than the ones used in FD.io lab are likely to yield different results. A good rule of thumb, that can be applied to estimate VPP packet thoughput for NIC-to-NIC switching topology, is to expect the forwarding performance to be proportional to processor core frequency for the same processor architecture, assuming processor is the only limiting factor and all other SUT parameters are equivalent to FD.io CSIT environment.

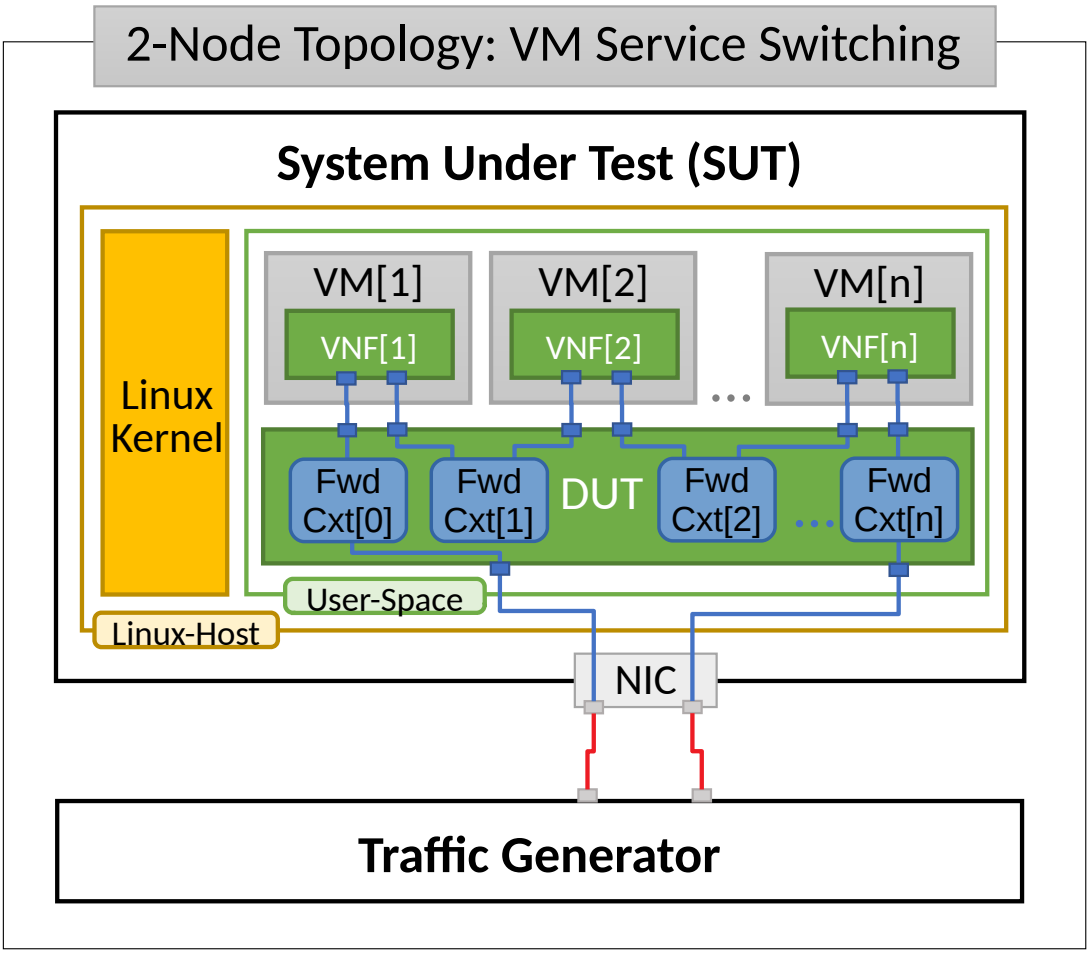
VM Service Switching

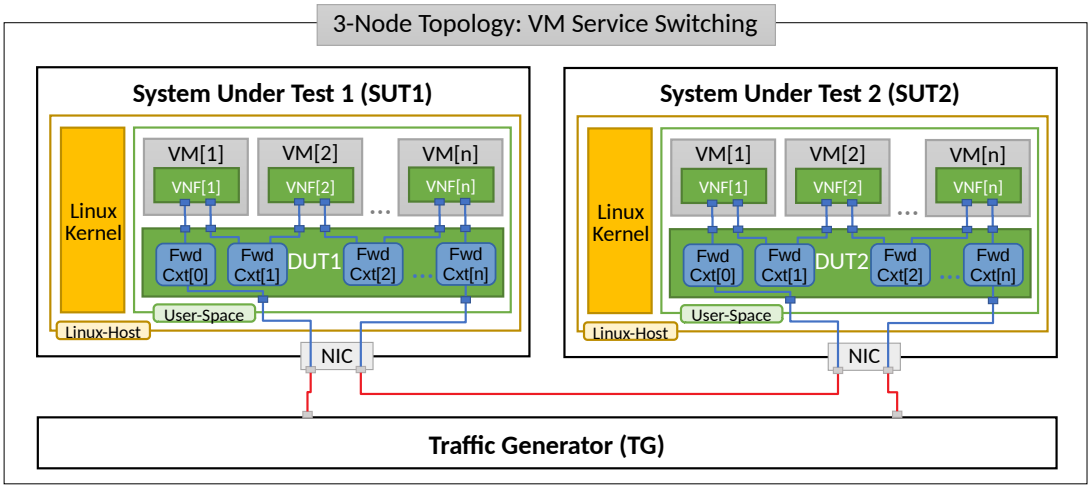
VM service switching topology test cases require VPP DUT to communicate with Virtual Machines (VMs) over vhost-user virtual interfaces.

Two types of VM service topologies are tested in CSIT-2210:

1. "Parallel" topology with packets flowing within SUT from NIC(s) via VPP DUT to VM, back to VPP DUT, then out thru NIC(s).
2. "Chained" topology (a.k.a. "Snake") with packets flowing within SUT from NIC(s) via VPP DUT to VM, back to VPP DUT, then to the next VM, back to VPP DUT and so on and so forth until the last VM in a chain, then back to VPP DUT and out thru NIC(s).

For each of the above topologies, VPP DUT is tested in a range of L2 or IPv4/IPv6 configurations depending on the test suite. Sample VPP DUT "Chained" VM service topologies for 2-Node and 3-Node testbeds with each SUT running N of VM instances is shown in the figures below.





In “Chained” VM topologies, packets are switched by VPP DUT multiple times: twice for a single VM, three times for two VMs, N+1 times for N VMs. Hence the external throughput rates measured by TG and listed in this report must be multiplied by N+1 to represent the actual VPP DUT aggregate packet forwarding rate.

For “Parallel” service topology packets are always switched twice by VPP DUT per service chain.

Note that reported VPP DUT performance results are specific to the SUTs tested. SUTs with other processor than the ones used in FD.io lab are likely to yield different results. Similarly to NIC-to-NIC switching topology, here one can also expect the forwarding performance to be proportional to processor core frequency for the same processor architecture, assuming processor is the only limiting factor. However due to much higher dependency on intensive memory operations in VM service chained topologies and sensitivity to Linux scheduler settings and behaviour, this estimation may not always yield good enough accuracy.

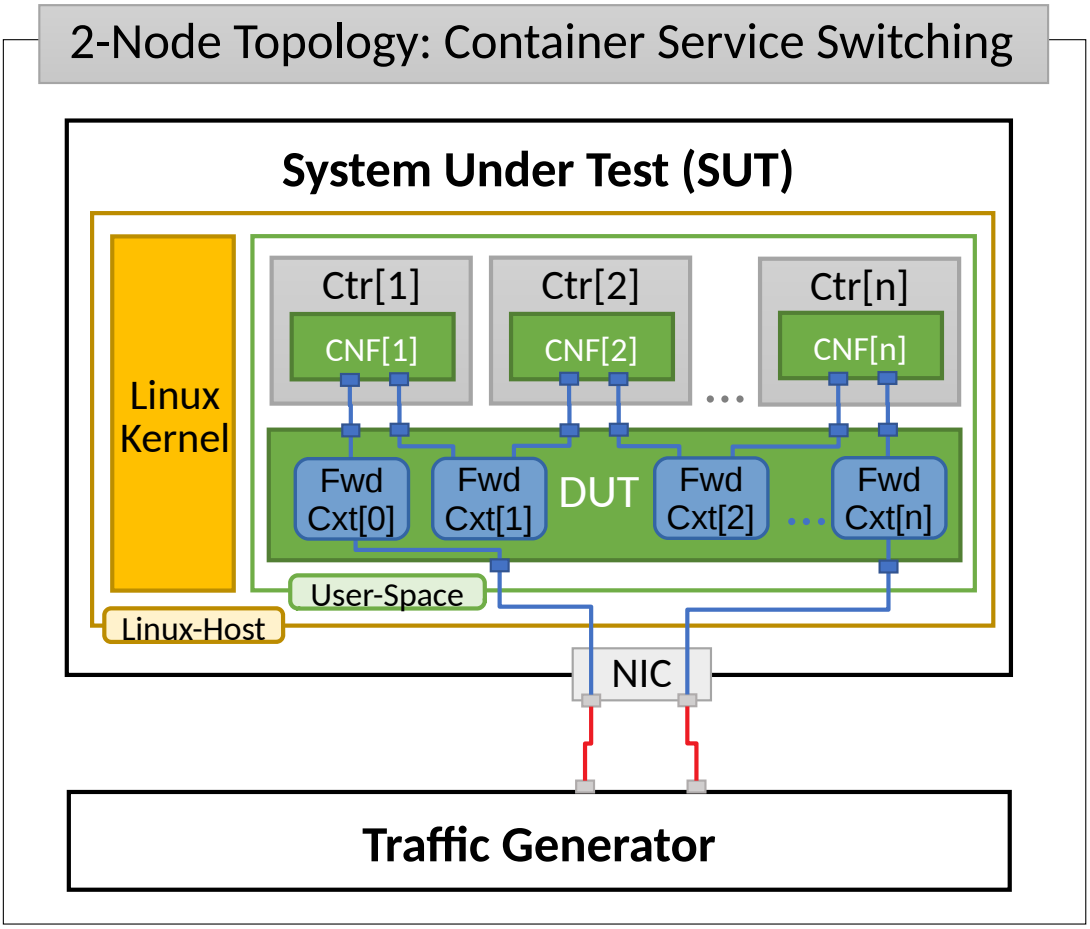
Container Service Switching

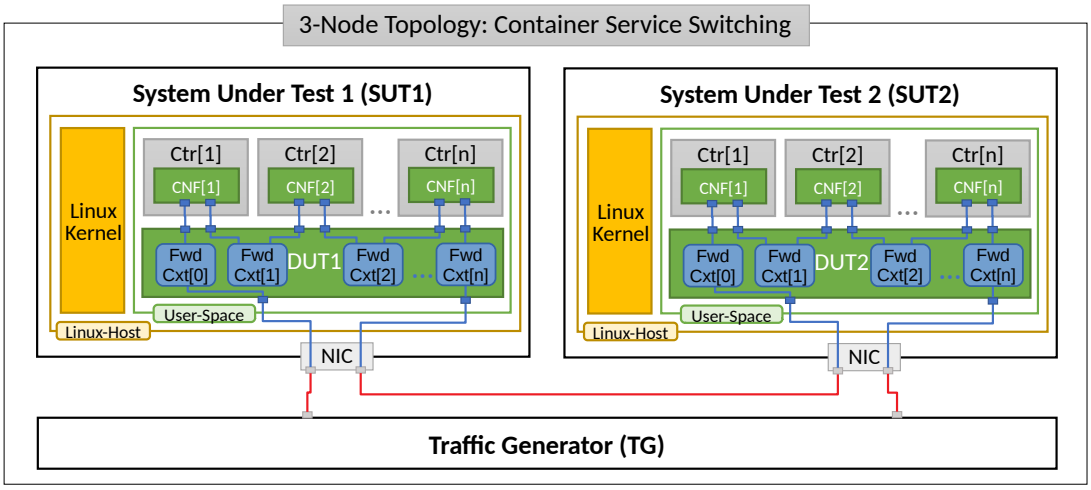
Container service switching topology test cases require VPP DUT to communicate with Containers (Ctrs) over memif virtual interfaces.

Three types of VM service topologies are tested in CSIT-2210:

1. “Parallel” topology with packets flowing within SUT from NIC(s) via VPP DUT to Container, back to VPP DUT, then out thru NIC(s).
2. “Chained” topology (a.k.a. “Snake”) with packets flowing within SUT from NIC(s) via VPP DUT to Container, back to VPP DUT, then to the next Container, back to VPP DUT and so on and so forth until the last Container in a chain, then back to VPP DUT and out thru NIC(s).
3. “Horizontal” topology with packets flowing within SUT from NIC(s) via VPP DUT to Container, then via “horizontal” memif to the next Container, and so on and so forth until the last Container, then back to VPP DUT and out thru NIC(s).

For each of the above topologies, VPP DUT is tested in a range of L2 or IPv4/IPv6 configurations depending on the test suite. Sample VPP DUT “Chained” Container service topologies for 2-Node and 3-Node testbeds with each SUT running N of Container instances is shown in the figures below.





In “Chained” Container topologies, packets are switched by VPP DUT multiple times: twice for a single Container, three times for two Containers, N+1 times for N Containers. Hence the external throughput rates measured by TG and listed in this report must be multiplied by N+1 to represent the actual VPP DUT aggregate packet forwarding rate.

For a “Parallel” and “Horizontal” service topologies packets are always switched by VPP DUT twice per service chain.

Note that reported VPP DUT performance results are specific to the SUTs tested. SUTs with other processor than the ones used in FD.io lab are likely to yield different results. Similarly to NIC-to-NIC switching topology, here one can also expect the forwarding performance to be proportional to processor core frequency for the same processor architecture, assuming processor is the only limiting factor. However due

to much higher dependency on intensive memory operations in Container service chained topologies and sensitivity to Linux scheduler settings and behaviour, this estimation may not always yield good enough accuracy.

2.1.2 Performance Tests Coverage

Performance tests measure following metrics for tested VPP DUT topologies and configurations:

- Packet Throughput: measured in accordance with **RFC 2544**[72], using FD.io CSIT Multiple Loss Ratio search (MLRsearch), an optimized binary search algorithm, producing throughput at different Packet Loss Ratio (PLR) values:
  - Non Drop Rate (NDR): packet throughput at PLR=0%.
  - Partial Drop Rate (PDR): packet throughput at PLR=0.5%.
- One-Way Packet Latency: measured at different offered packet loads:
  - 90% of discovered PDR throughput.
  - 50% of discovered PDR throughput.
  - 10% of discovered PDR throughput.
  - Minimal offered load.
- Maximum Receive Rate (MRR): measure packet forwarding rate under the maximum load offered by traffic generator over a set trial duration, regardless of packet loss. Maximum load for specified Ethernet frame size is set to the bi-directional link rate, unless there is a known limitation preventing Traffic Generator from achieving the line rate.

CSIT-2210 includes following VPP data plane functionality performance tested across a range of NIC drivers and NIC models:

[72] https://datatracker.ietf.org/doc/html/rfc2544.html

Functionality	Description
ACL	L2 Bridge-Domain switching and IPv4and IPv6 routing with iACL and oACL IP address, MAC address and L4 port security.
ADL	IPv4 and IPv6 routing with ADL address security.
GENEVE	GENEVE tunnels for IPv4 routing.
IPv4	IPv4 routing.
IPv6	IPv6 routing.
IPv4 Scale	IPv4 routing with 20k, 200k and 2M FIB entries.
IPv6 Scale	IPv6 routing with 20k, 200k and 2M FIB entries.
IPSecAsyncHW	IPSec encryption with AES-GCM, CBC-SHA-256 ciphers in async mode, in combination with IPv4 routing. Intel QAT HW acceleration.
IPSecHW	IPSec encryption with AES-GCM, CBC-SHA-256 ciphers, in combination with IPv4 routing. Intel QAT HW acceleration.
IPSec+LISP	IPSec encryption with CBC-SHA1 ciphers, in combination with LISP-GPE overlay tunneling for IPv4-over-IPv4.
IPSecSW	IPSec encryption with AES-GCM, CBC-SHA-256 ciphers, in combination with IPv4 routing.
KVM VMs vhost-user	Virtual topologies with service chains of 1 VM using vhost-user interfaces, with different VPP forwarding modes incl. L2XC, L2BD, VXLAN with L2BD, IPv4 routing.
L2BD	L2 Bridge-Domain switching of untagged Ethernet frames with MAC learning; disabled MAC learning i.e. static MAC tests to be added.
L2BD Scale	L2 Bridge-Domain switching of untagged Ethernet frames with MAC learning; disabled MAC learning i.e. static MAC tests to be added with 20k, 200k and 2M FIB entries.
L2XC	L2 Cross-Connect switching of untagged, dot1q, dot1ad VLAN tagged Ethernet frames.
LISP	LISP overlay tunneling for IPv4-over-IPv4, IPv6-over-IPv4, IPv6-over-IPv6, IPv4-over-IPv6 in IPv4 and IPv6 routing modes.
LXC/DRC Containers Memif	Container VPP memif virtual interface tests with different VPP forwarding modes incl. L2XC, L2BD.
NAT44	(Source) Network Address Translation deterministic mode and endpoint-dependent mode tests with varying number of users and ports per user for IPv4.
QoS Policer	Ingress packet rate measuring, marking and limiting (IPv4).
SRv6 Routing	Segment Routing IPv6 tests.
VPP TCP/IP stack	Tests of VPP TCP/IP stack used with VPP built-in HTTP server.
VTS	Virtual Topology System use case tests combining VXLAN overlay tunneling with L2BD, ACL and KVM VM vhost-user features.
VXLAN	VXLAN overlay tunnelling integration with L2XC and L2BD.

Execution of performance tests takes time, especially the throughput tests. Due to limited HW testbed resources available within FD.io labs hosted by LF, the number of tests for some NIC models has been limited to few baseline tests.

2.1.3 Performance Tests Naming

FD.io CSIT-2210 follows a common structured naming convention for all performance and system functional tests.

The naming should be intuitive for majority of the tests. Complete description of FD.io CSIT test naming convention is provided on *Test Naming* (page 1430).

2.2 Release Notes

2.2.1 Changes in CSIT-2210

1. VPP PERFORMANCE TESTS
   - **Added new performance testbed 3n-snr** (3 Node SnowRidge, with Intel Atom processors), to later replace 3n-dnv and 2n-dnv (3 and 2 Node Denverton) testbeds.
   - **Added GTPU HW offload tests** using VPP GTPU hardware offload with Intel e810 4p25ge NICs (3n-icx testbeds only). These tests were already there in CSIT-2206, but were yielding invalid results due to using TRex v2.97 that was incompatible with e810 NICs used for those tests.
   - **Added Wireguard tests** using VPP software crypto (3n-icx, 3n-snr testbeds) and using built-in hardware crypto QAT device (3n-snr testbed only).
   - **Reduction of tests**: Removed certain test variations executed iteratively for the report (as well as in daily and weekly trending) due to physical testbeds overload.
2. TEST FRAMEWORK
   - CSIT-2210 executes all VPP v22.10 performance tests using vpp ubuntu2204 images, due to CSIT execution environment change as noted below. This applies to all performance testbeds except Denverton. Consequently, VPP v22.06 has not been re-tested in CSIT-2210 environment, as no ubuntu204 images are available for that VPP version. Performance comparison between VPP v22.10 (current version) vs VPP v22.06 (previous version) may be impacted by VPP build environment change (ubuntu2004 to ubuntu 2204) change and CSIT environment change. See *Root Cause Analysis for Performance Changes* (page 89) for details.
   - **CSIT test environment** version has been updated to ver. 11, see *Environment Versioning* (page 1206).
   - **TCP TPUT profiles** had to be changed, as newer TRex versions are not deterministic enough when deciding when to send an ACK.
   - **CSIT PAPI support**: Due to issues with PAPI performance, and deprecation of VAT, VPP CLI is used in CSIT for many VPP scale tests. See *Known Issues* (page 87).
   - **General Code Housekeeping**: Ongoing code optimizations and bug fixes.
3. PRESENTATION AND ANALYTICS LAYER
   - **C-Dash** performance dashboard[73] got updated UI and updated backend increasing its performance and robustness.

[73] http://csit.fd.io/

2.2.2 Known Issues

New

#	JiraID	Issue Description
1	CSIT-1850[74]	2n-dnv: sporadic 1518B tput tests failing to establish required sessions.
2	CSIT-1864[75]	2n-clx: half of the packets lost on PDR tests.
3	CSIT-1868[76]	2n-clx: ALL ldpreload-nginx tests fails when trying to start nginx.
4	CSIT-1871[77]	3n-snr: 25GE interface between SUT and TG/TRex goes down randomly.
5	CSIT-1877[78]	3n-alt, 3n-tsh: VM tests failing to boot VM.
6	CSIT-1883[79]	3n-snr: All hwasync wireguard tests failing when trying to verify device.
7	CSIT-1884[80]	2n-clx, 2n-icx: All NAT44DET NDR PDR IMIX over 1M sessions BIDIR tests failing to create enough sessions.
8	CSIT-1885[81]	3n-icx: 9000b ip4 ip6 l2 NDRPDR AVF tests are failing to forward traffic.
9	CSIT-1886[82]	3n-icx: Wireguard tests with 100 and more tunnels are failing PDR criteria.

Previous

Issues reported in previous releases which still affect the current results.

[74] https://jira.fd.io/browse/CSIT-1850
[75] https://jira.fd.io/browse/CSIT-1864
[76] https://jira.fd.io/browse/CSIT-1868
[77] https://jira.fd.io/browse/CSIT-1871
[78] https://jira.fd.io/browse/CSIT-1877
[79] https://jira.fd.io/browse/CSIT-1883
[80] https://jira.fd.io/browse/CSIT-1884
[81] https://jira.fd.io/browse/CSIT-1885
[82] https://jira.fd.io/browse/CSIT-1886

#	JiraID	Issue Description
1	CSIT-1671[83], VPP-1763[84]	All CSIT scale tests can not use PAPI due to much slower performance compared to VAT/CLI (it takes much longer to program VPP). This needs to be addressed on the PAPI side. Currently, the time critical code uses VAT running large files with exec statements and CLI commands. Still, we needed to reduce the number of scale tests run to keep overall duration reasonable. More improvements needed to achieve sufficient configuration speed.
2	CSIT-1782[85]	Multicore AVF tests are failing when trying to create interface. Frequency is reduced by CSIT workaround, but occasional failures do still happen.
3	CSIT-1785[86], VPP-1972[87]	NAT44ED tests failing to establish all TCP sessions. At least for max scale, in allotted time (limited by session 500s timeout) due to worse slow path performance than previously measured and calibrated for. CSIT removed the max scale NAT tests to avoid this issue.
4	CSIT-1799[88]	All NAT44-ED 16M sessions CPS scale tests fail while setting NAT44 address range.
5	CSIT-1800[89]	All Geneve L3 mode scale tests (1024 tunnels) are failing.
6	CSIT-1801[90]	9000B payload frames not forwarded over tunnels due to violating supported Max Frame Size (VxLAN, LISP, SRv6).
7	CSIT-1802[91]	AF-XDP - NDR tests failing from time to time.
8	CSIT-1804[92]	All testbeds: NDR tests failing from time to time.
9	CSIT-1808[93]	All tests with 9000B payload frames not forwarded over memif interfaces.
10	CSIT-1827[94]	3n-icx, 3n-skx: all AVF crypto tests sporadically fail. 1518B with no traffic, IMIX with excessive packet loss.
11	CSIT-1835[95]	3n-icx: QUIC vppecho BPS tests failing on timeout when checking hoststack finished.
12	CSIT-1849[96]	2n-skx, 2n-clx, 2n-icx: UDP 16m TPUT tests fail to create all sessions.

[83] https://jira.fd.io/browse/CSIT-1671
[84] https://jira.fd.io/browse/VPP-1763
[85] https://jira.fd.io/browse/CSIT-1782
[86] https://jira.fd.io/browse/CSIT-1785
[87] https://jira.fd.io/browse/VPP-1972
[88] https://jira.fd.io/browse/CSIT-1799
[89] https://jira.fd.io/browse/CSIT-1800
[90] https://jira.fd.io/browse/CSIT-1801
[91] https://jira.fd.io/browse/CSIT-1802
[92] https://jira.fd.io/browse/CSIT-1804
[93] https://jira.fd.io/browse/CSIT-1808
[94] https://jira.fd.io/browse/CSIT-1827
[95] https://jira.fd.io/browse/CSIT-1835
[96] https://jira.fd.io/browse/CSIT-1849

Fixed

Issues reported in previous releases which were fixed in this release:

#	JiraID	Issue Description
1	CSIT-1834[97]	2n-icx, 2n-skx: sporadic AVF soak tests failing to find critical load with PLRsearch.
2	CSIT-1846[98]	2n-skx, 2n-clx, 2n-icx: ALL 1518B TCP tput tests failing with big packet loss.
3	CSIT-1851[99]	trending regression: various icelake tests around 2202-04-15 Somewhat expected consequence of a VPP usability fix, the previous VPP compiler version was too new for the OS used.

2.2.3 Root Cause Analysis for Performance Changes

List of RCAs in CSIT-2210 for VPP performance changes:

#	JiraID	Issue Description
1	CSIT-1887[100]	rls2210 RCA: ASTF tests TRex upgrade decreased TRex performance. NAT results not affected, except on Denverton due to interference from VPP-2010.
2	CSIT-1888[101]	rls2210 RCA: testbed differences, especially for ipsec Not caused by VPP code nor CSIT code. Most probable cause is clang-14 behavior.
3	CSIT-1889[102]	rls2210 RCA: policy-outbound-nocrypto When VPP added spd fast path matching (Gerrit 36097), it decreased MRR of the corresponding tests, at least on 3-alt.

[97] https://jira.fd.io/browse/CSIT-1834
[98] https://jira.fd.io/browse/CSIT-1846
[99] https://jira.fd.io/browse/CSIT-1851
[100] https://jira.fd.io/browse/CSIT-1887
[101] https://jira.fd.io/browse/CSIT-1888
[102] https://jira.fd.io/browse/CSIT-1889

2.3 Packet Throughput

Throughput graphs are generated based on the results data obtained from the CSIT-2210 test jobs. In order to verify benchmark results repeatibility selected, CSIT performance tests are executed multiple times (target: 10 times) on each physical testbed type. Box-and-Whisker plots are used to display variations in measured throughput values.

Lists of tests selected for multiple execution and graphing are captured per testbed type in test_select_list_{testbed_type}.md[103] files.

Graphs are split into sections as follows:

1. **Header 1**: VPP packet path and lookup types
    - **L2 Ethernet Switching**: L2 bridge-doman, L2 cross-connect and L2 patch
    - **IPv4 Routing**: IPv4 routing with /32 prefixes
    - **IPv6 Routing**: IPv6 routing with /128 prefixes
    - **SRv6 Routing**: SRv6 with IPv6 routing
    - **IPv4 Tunnels**: IPv4 overlay tunnels
    - **KVM VMs vhost-user**: KVM VMs connected over virtio and vhost-user interfaces
    - **LXC/DRC Container Memif**: Linux containers and Docker containers connected over Memif interfaces
    - **IPsec IPv4 Routing**: IPsec encryption/decryption with IPv4 routing
2. **Header 2**: testbeds and NIC models
    - section name format:
        - {**testbed_type**}-{**nic_model**}
    - **testbed_type**:
        - 2n-icx: 2-node Xeon Icelake
        - 3n-icx: 3-node Xeon Icelake
        - 2n-aws: 2-node AWS
        - 2n-clx: 2-node Xeon Cascade Lake
        - 2n-zn2: 2-node AMD Zen2
        - 3n-alt: 2-node Arm Altra
        - 3n-tsh: 3-node Arm TaiShan
        - 2n-tx2: 2-node Arm ThunderX2
        - 2n-dnv: 2-node Atom Denverton
        - 3n-dnv: 3-node Atom Denverton
        - 3n-snr: 3-node Atom Snowridge
    - **nic_model**:
        - xxv710: xxv710 2p25GE Intel (Fortville)
        - x710: x710 4p10GE Intel (Fortville)
        - xl710: xl710 2p40GE Intel (Fortville)
        - x520: x520 2p10GE Intel (Niantic)
        - x553: x553 2p10GE Intel (Niantic)

[103] https://git.fd.io/csit/tree/docs/job_specs

  - cx556a: cx556a-edat 2p100GE Mellanox ConnectX5
  - e810cq: E810-2CQDA2 2p100GE Intel Columbiaville

3. **Header 3**: test group names
   - section name format:
     - {**frame_size**}-{**worker_thread_core_cfg**}-{**vpp_functionality**}-{**vpp_lookup_type**}-{**baseline_scale**}-{**nic_driver**}
   - **frame_size**:
     - 64b: 64 byte frames, smallest frame size for untagged IPv4 packets
     - 78b: 78 byte frames, smallest frame size for untagged IPv6 packets
     - 114b: VXLAN encapsulated L2 frames
     - imix: a sequence of (7x64B, 4x570, 1x1518) byte frames
   - **worker_thread_core_cfg**:
     - 1t1c: 1 worker thread on 1 core, hyper-threading not used
     - 2t1c: 2 worker threads on 1 core, hyper-threading used
   - **vpp_functionality** (optional):
     - features: including input-acl, output-acl, macip-iacl, nat44
     - srv6: srv6 encap/decap, proxy
     - link-bonding: L2 link aggregation with 1 or 2 bonded links
     - ipsec: IPsec encryption/decryption with different ciphers
     - vts: Virtual Topology System specific tests
   - **vpp_lookup_type**:
     - l2switching, ip4routing, ip6routing, ip4tunnel, vhost, memif
   - **baseline_scale**:
     - base: baseline tests with less than 10 forwarding entries
     - scale: scale tests with up to 2 million forwarding entries
     - base-scale: both baseline and scale tests grouped together
   - **nic_driver**:
     - avf: VPP native avf driver for Intel Fortville NICs
     - i40e: dpdk poll mode driver for Intel Fortville NICs
     - ixgbe: dpdk poll mode driver for Intel Niantic NICs

For each test case, Box-and-Whisker plots show the quartiles (Min, 1st quartile / 25th percentile, 2nd quartile / 50th percentile / mean, 3rd quartile / 75th percentile, Max) across collected data set. Outliers are plotted as individual points.

Additional information about graph data:

1. **Graph Title**: describes tested packet path, testbed topology, processor model, NIC model, packet size, number of cores and threads used by data plane workers and indication of VPP DUT configuration.
2. **X-axis Labels**: indices of individual test suites as listed in Graph Legend.
3. **Y-axis Labels**: measured Packets Per Second [pps] throughput values.
4. **Graph Legend**: lists X-axis indices with associated CSIT test suites executed to generate graphed test results.

5. **Hover Information**: lists minimum, first quartile, median, third quartile, and maximum. If either type of outlier is present the whisker on the appropriate side is taken to 1.5×IQR from the quartile (the "inner fence") rather than the max or min, and individual outlying data points are displayed as unfilled circles (for suspected outliers) or filled circles (for outliers). (The "outer fence" is 3×IQR from the quartile.)

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[104], build logs from FD.io vpp performance job 3n-icx[105], build logs from FD.io vpp performance job 2n-aws[106], build logs from FD.io vpp performance job 2n-clx[107], build logs from FD.io vpp performance job 2n-zn2[108], build logs from FD.io vpp performance job 3n-alt[109], build logs from FD.io vpp performance job 3n-tsh[110], build logs from FD.io vpp performance job 2n-tx2[111], **`build logs from FD.io vpp performance job 3n-snr`_**, build logs from FD.io vpp performance job 2n-dnv[112] and build logs from FD.io vpp performance job 3n-dnv[113] with RF result files csit-vpp-perf-2210-*.zip archived here. Required per test case data set size is **10**, but for VPP tests the actual size varies per test case and is <=10.

---

[104] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[105] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-icx
[106] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-aws
[107] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx
[108] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-zn2
[109] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-alt
[110] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-tsh
[111] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-tx2
[112] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-dnv
[113] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-dnv

2.3.1 L2 Ethernet Switching

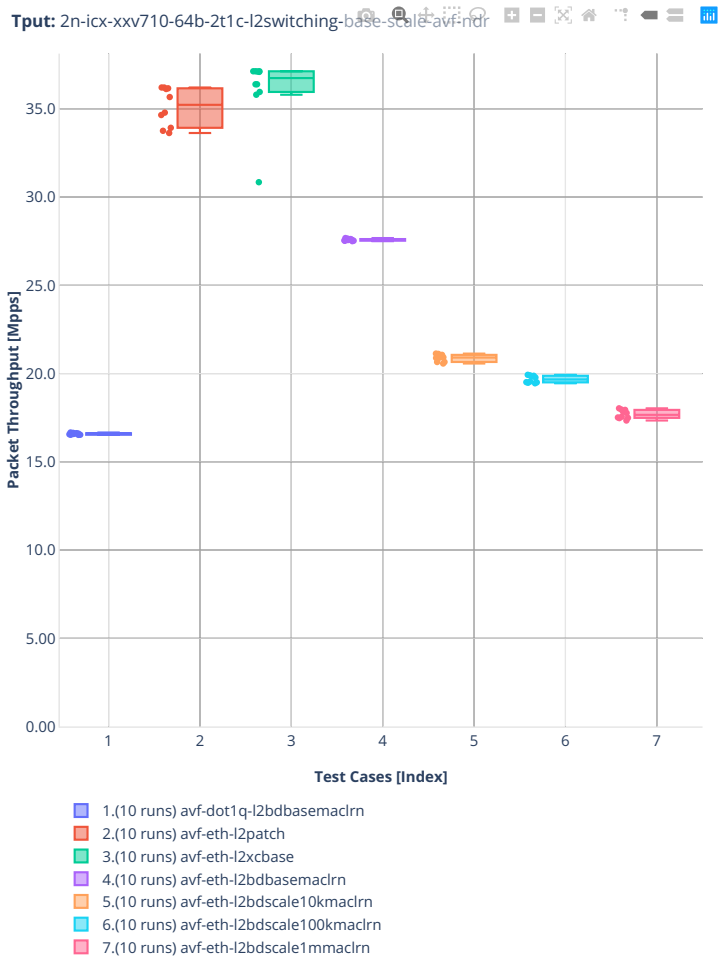
Following sections include summary graphs of VPP Phy-to-Phy performance with L2 Ethernet switching, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

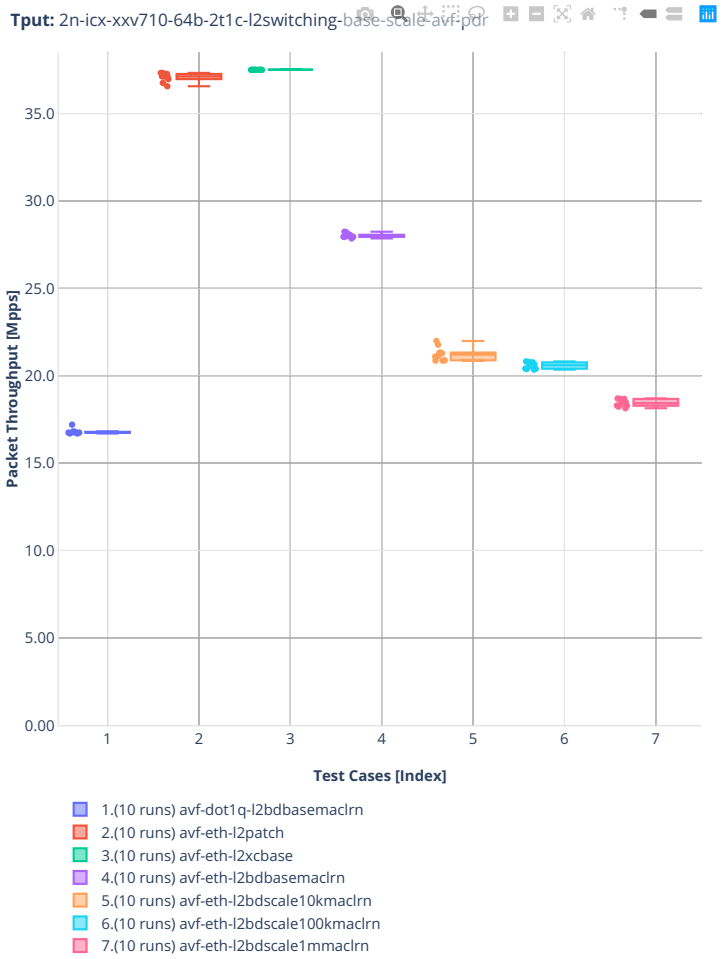
CSIT source code for the test cases used for plots can be found in CSIT git repository[114].

[114] https://git.fd.io/csit/tree/tests/vpp/perf/l2?h=rls2210

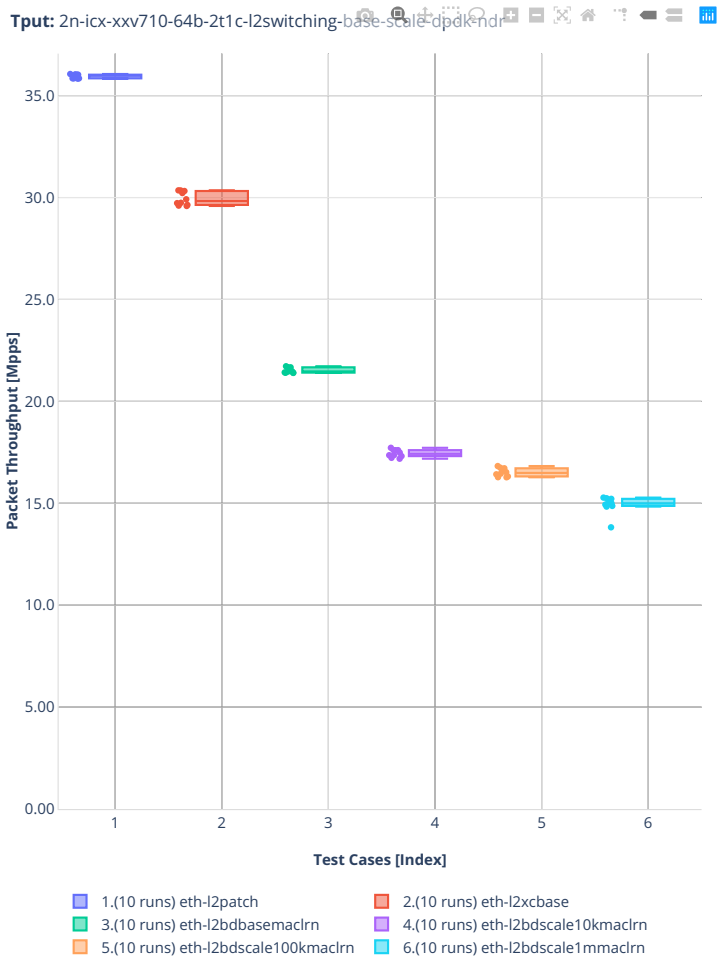
2n-icx-xxv710

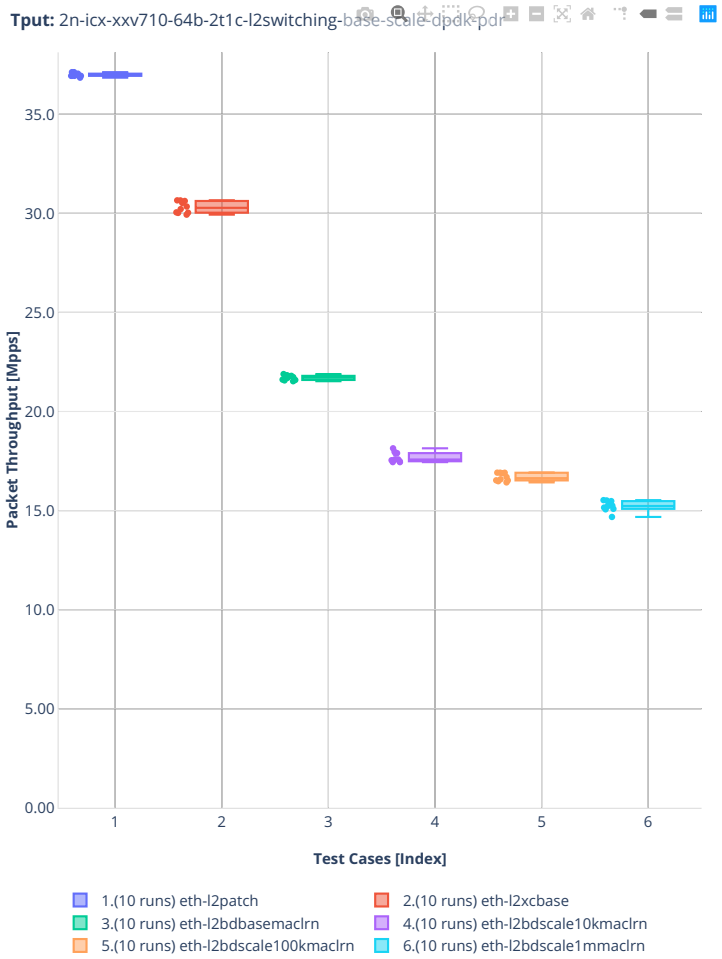
64b-2t1c-l2switching-base-scale-avf




Tput: 2n-icx-xxv710-64b-2t1c-l2switching-base-scale-avf-pdr
Packet Throughput [Mpps]
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
7
Test Cases [Index]
1.(10 runs) avf-dot1q-l2bdbasemaclrn
2.(10 runs) avf-eth-l2patch
3.(10 runs) avf-eth-l2xcbase
4.(10 runs) avf-eth-l2bdbasemaclrn
5.(10 runs) avf-eth-l2bdscale10kmaclrn
6.(10 runs) avf-eth-l2bdscale100kmaclrn
7.(10 runs) avf-eth-l2bdscale1mmaclrn

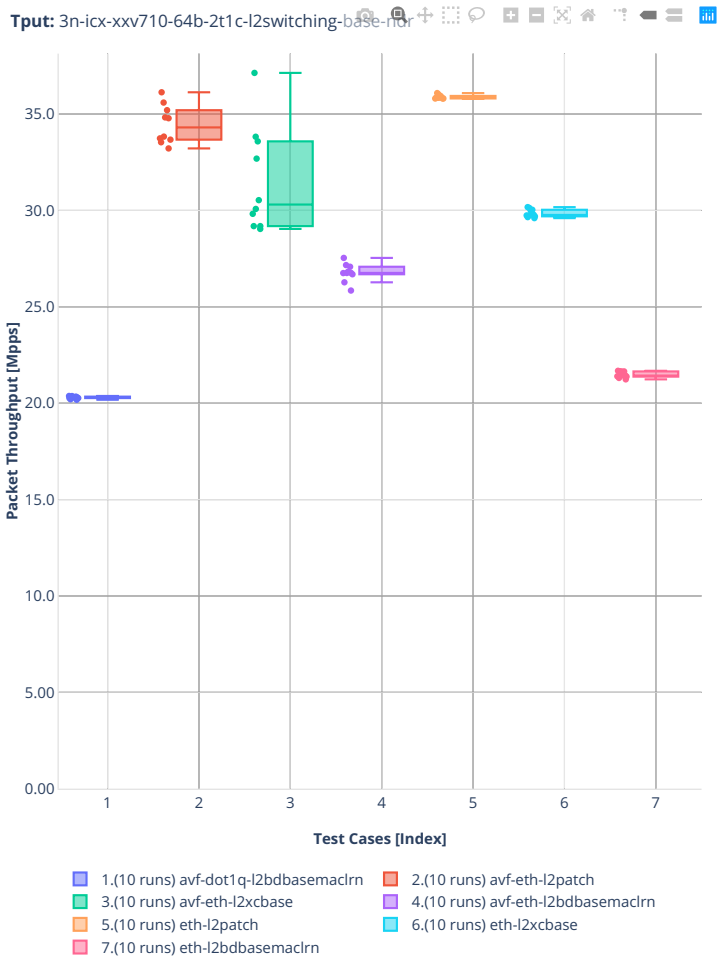
64b-2t1c-l2switching-base-scale-dpdk

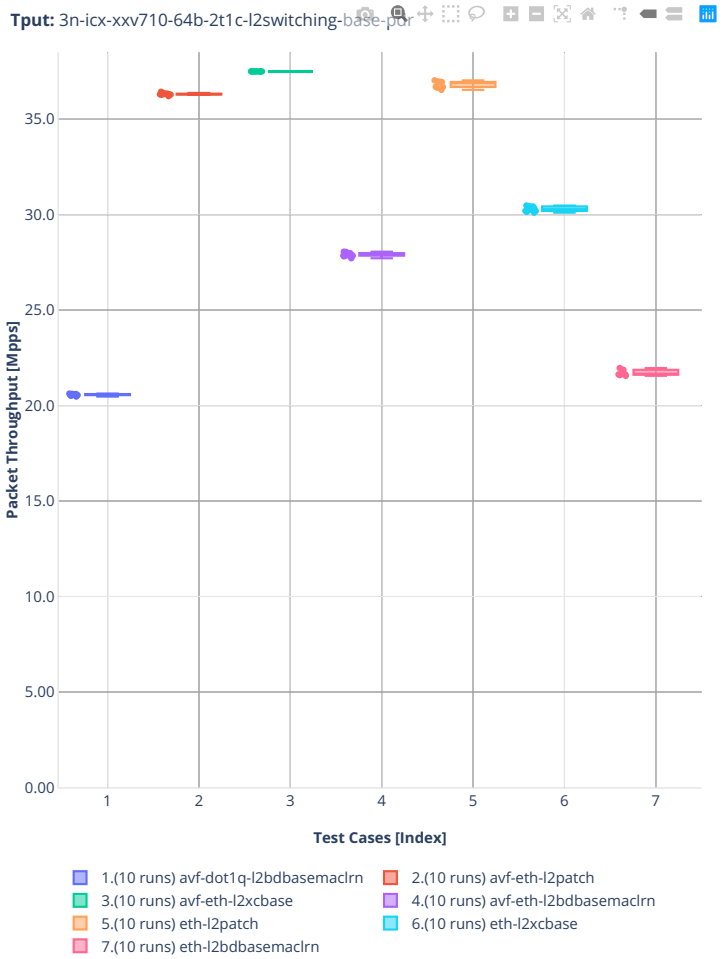



Tput: 2n-icx-xxv710-64b-2t1c-l2switching-base-scale-dpdk-pdr
Packet Throughput [Mpps]
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) eth-l2patch
2.(10 runs) eth-l2xcbase
3.(10 runs) eth-l2bdbasemaclrn
4.(10 runs) eth-l2bdscale10kmaclrn
5.(10 runs) eth-l2bdscale100kmaclrn
6.(10 runs) eth-l2bdscale1mmaclrn

3n-icx-xxv710

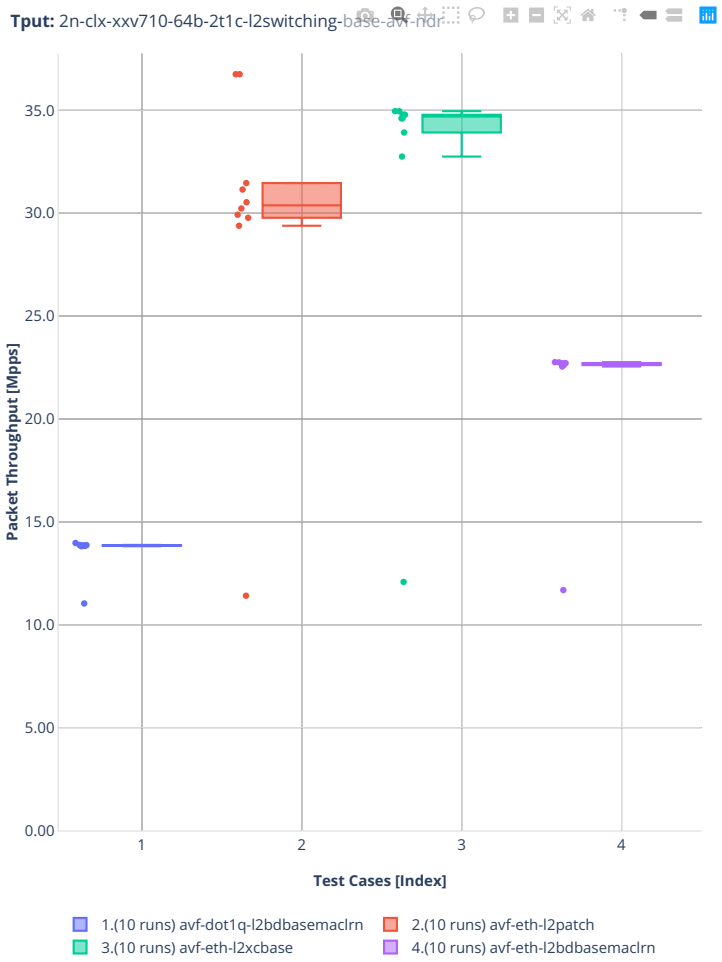
64b-2t1c-l2switching-base

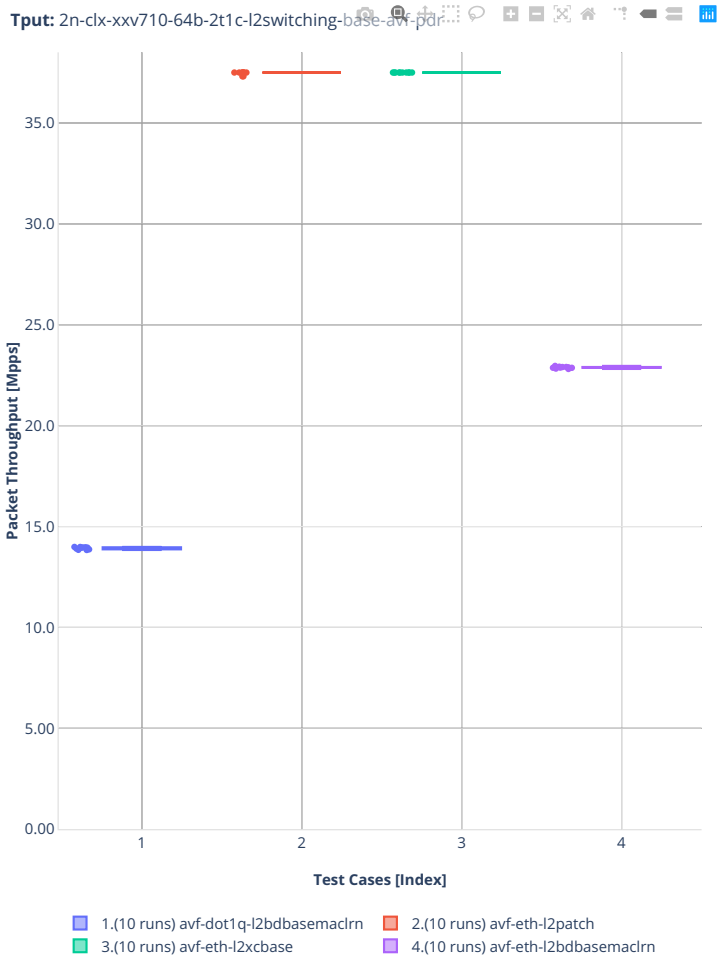



Tput: 3n-icx-xxv710-64b-2t1c-l2switching-base-pdr
Packet Throughput [Mpps]
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
7
Test Cases [Index]
1.(10 runs) avf-dot1q-l2bdbasemaclrn
2.(10 runs) avf-eth-l2patch
3.(10 runs) avf-eth-l2xcbase
4.(10 runs) avf-eth-l2bdbasemaclrn
5.(10 runs) eth-l2patch
6.(10 runs) eth-l2xcbase
7.(10 runs) eth-l2bdbasemaclrn

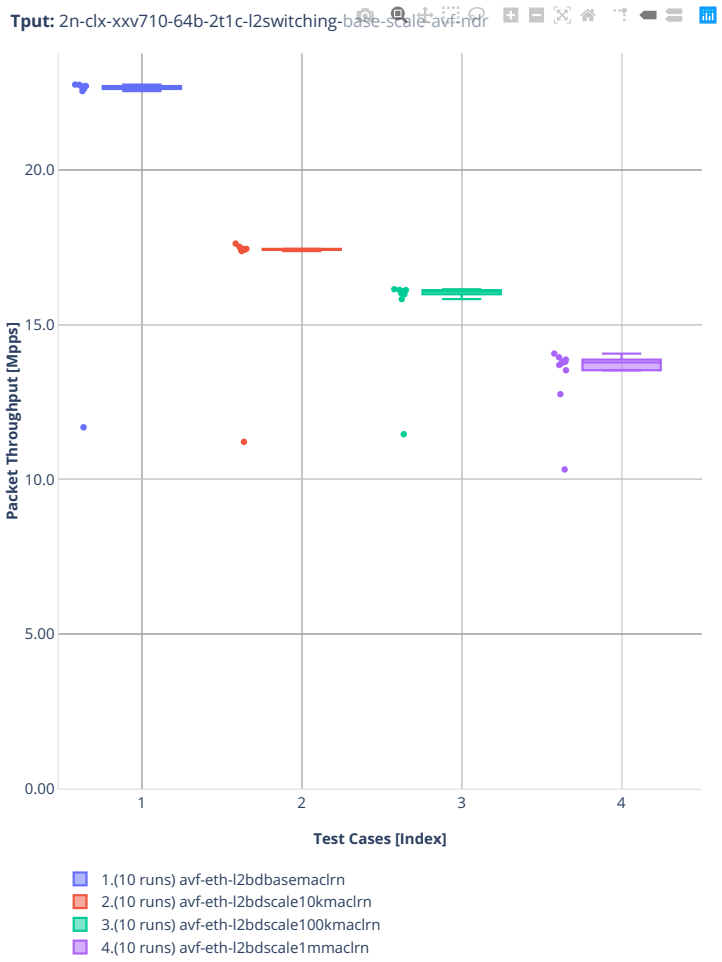
2n-clx-xxv710

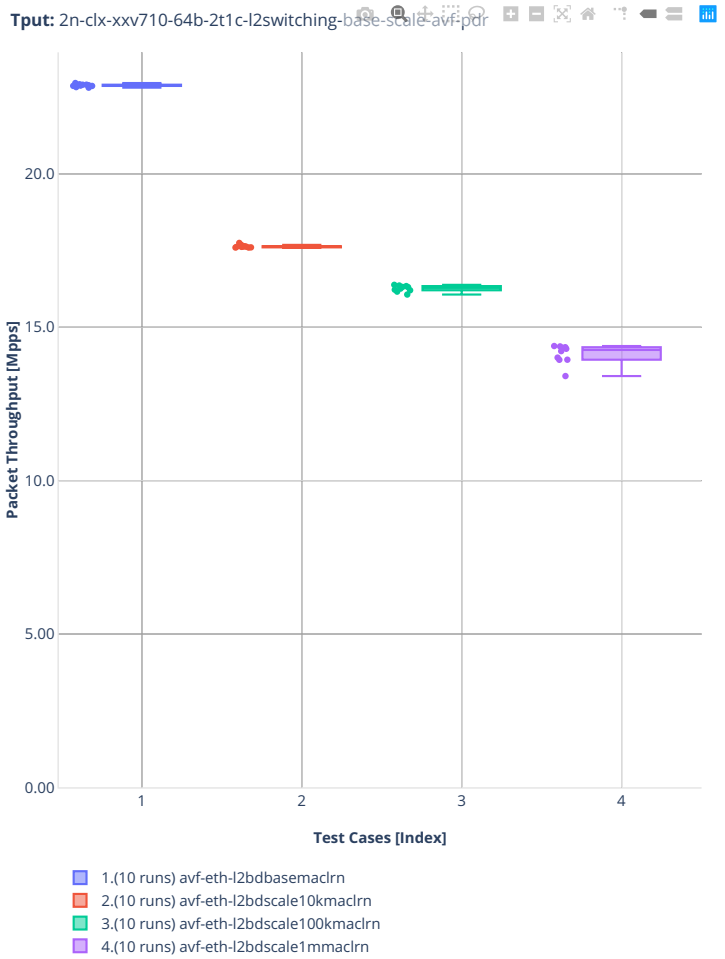
64b-2t1c-l2switching-base-avf



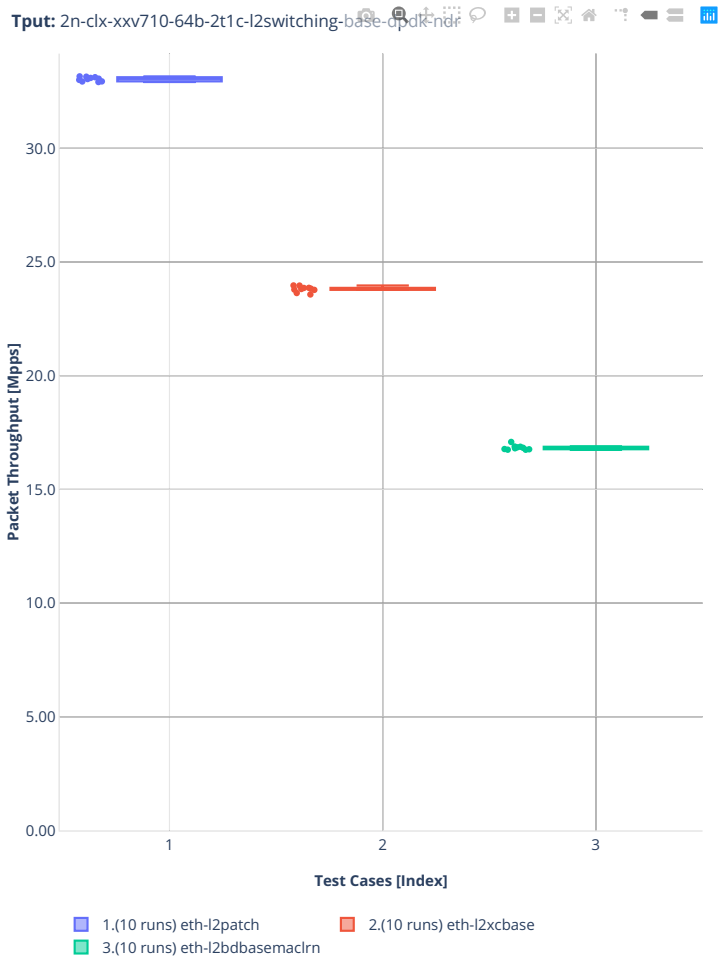

Tput: 2n-clx-xxv710-64b-2t1c-l2switching-base-avf-pdr
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-dot1q-l2bdbasemaclrn
2.(10 runs) avf-eth-l2patch
3.(10 runs) avf-eth-l2xcbase
4.(10 runs) avf-eth-l2bdbasemaclrn

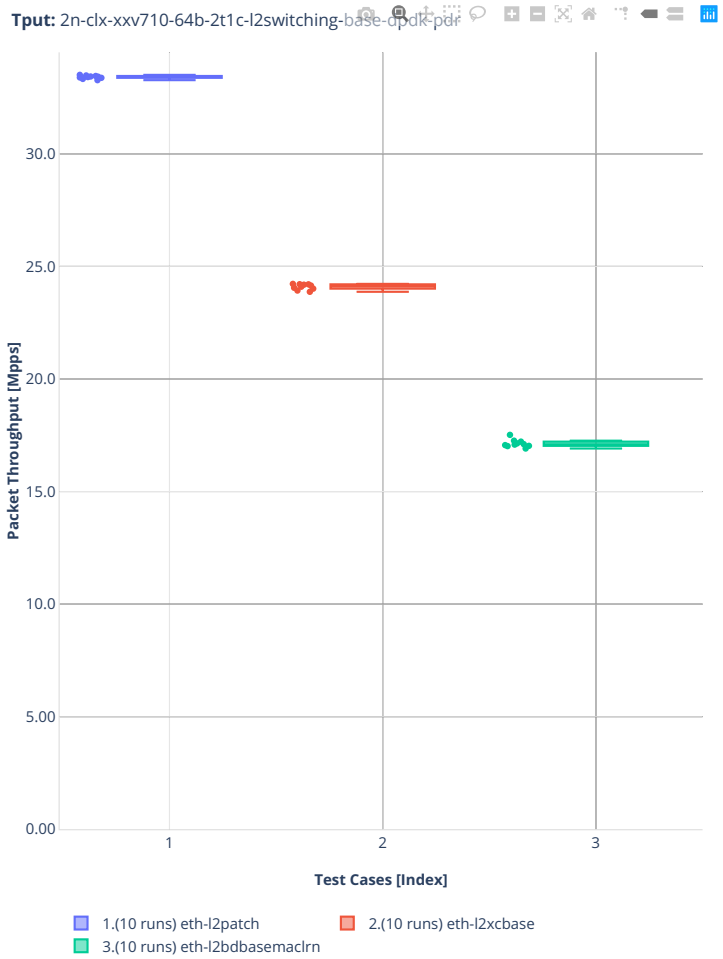
64b-2t1c-l2switching-base-scale-avf



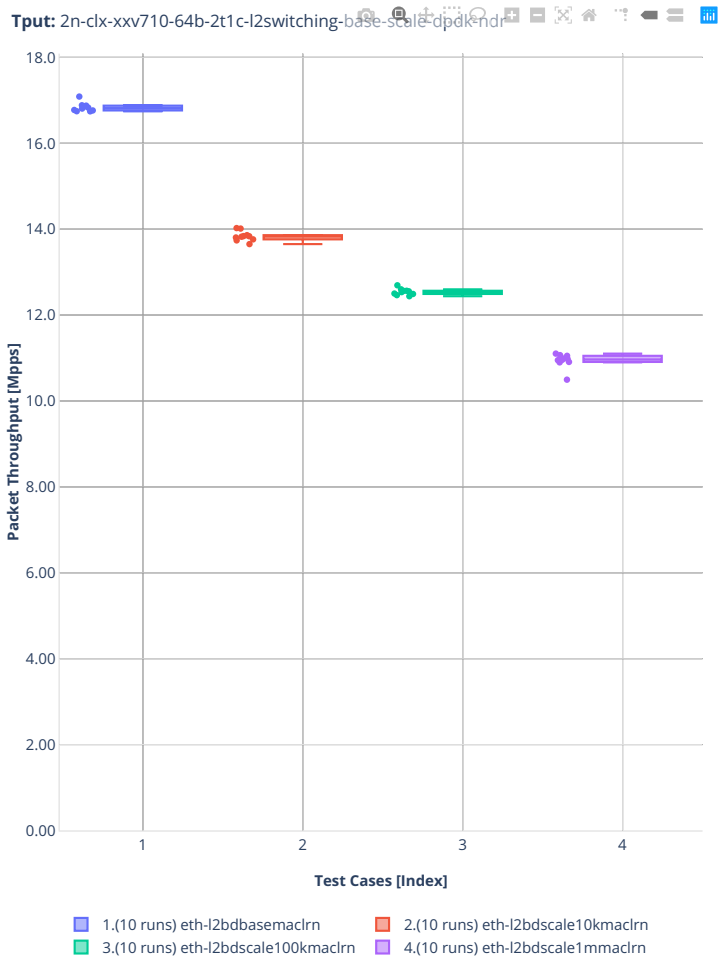

Tput: 2n-clx-xxv710-64b-2t1c-l2switching-base-scale-avf-pdr
Packet Throughput [Mpps]
20.0
15.0
10.0
5.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-eth-l2bdbasemaclrn
2.(10 runs) avf-eth-l2bdscale10kmaclrn
3.(10 runs) avf-eth-l2bdscale100kmaclrn
4.(10 runs) avf-eth-l2bdscale1mmaclrn

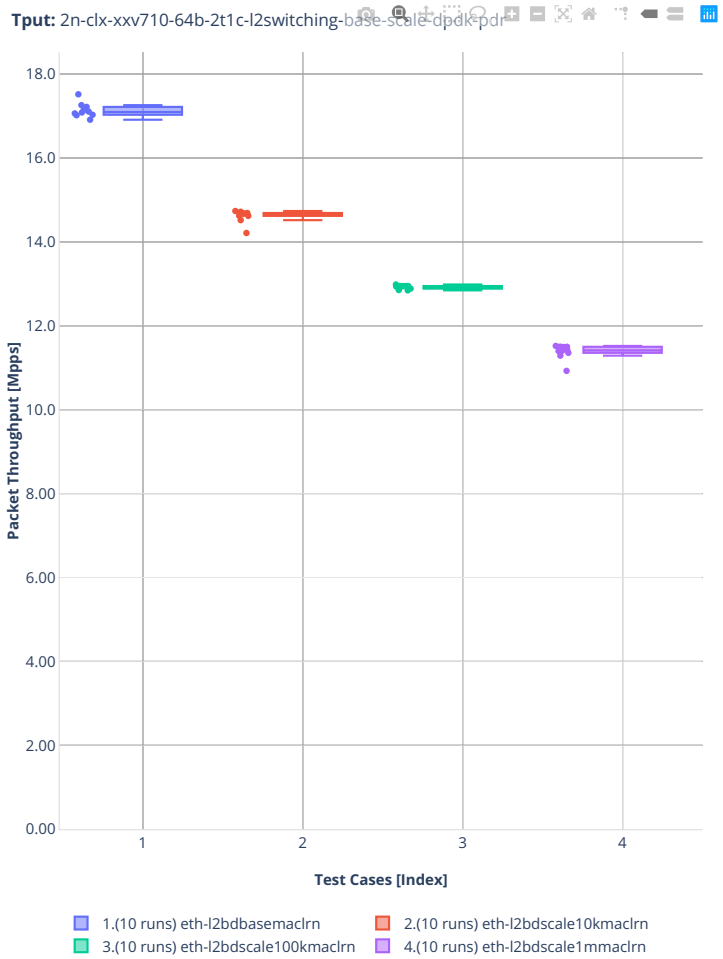
64b-2t1c-l2switching-base-dpdk




Tput: 2n-clx-xxv710-64b-2t1c-l2switching-base-dpdk-pdr
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) eth-l2patch
2.(10 runs) eth-l2xcbase
3.(10 runs) eth-l2bdbasemaclrn

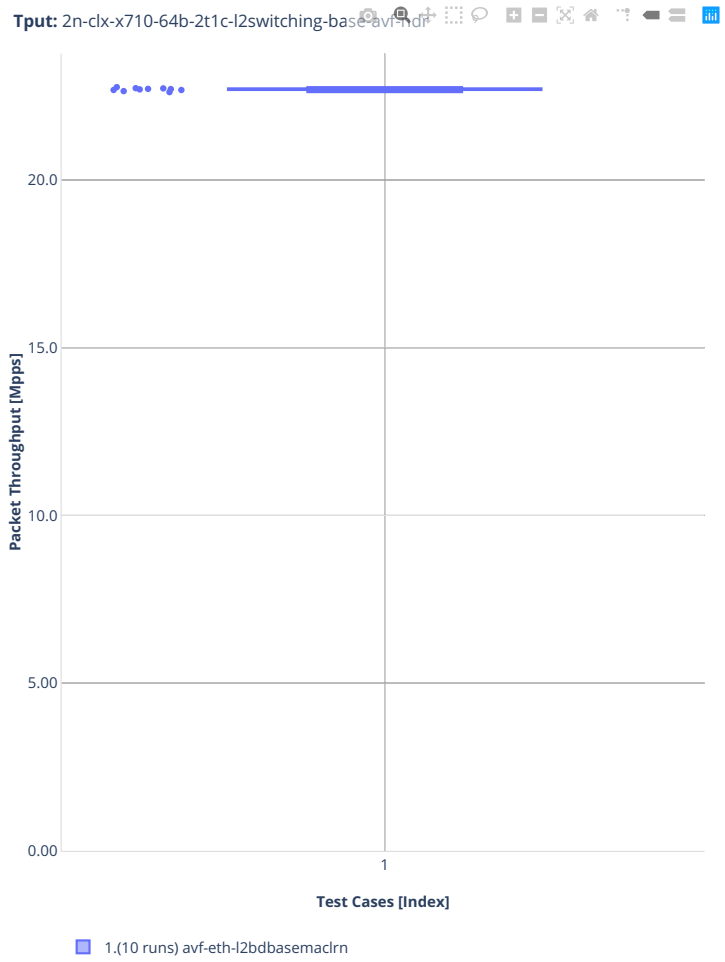
64b-2t1c-l2switching-base-scale-dpdk

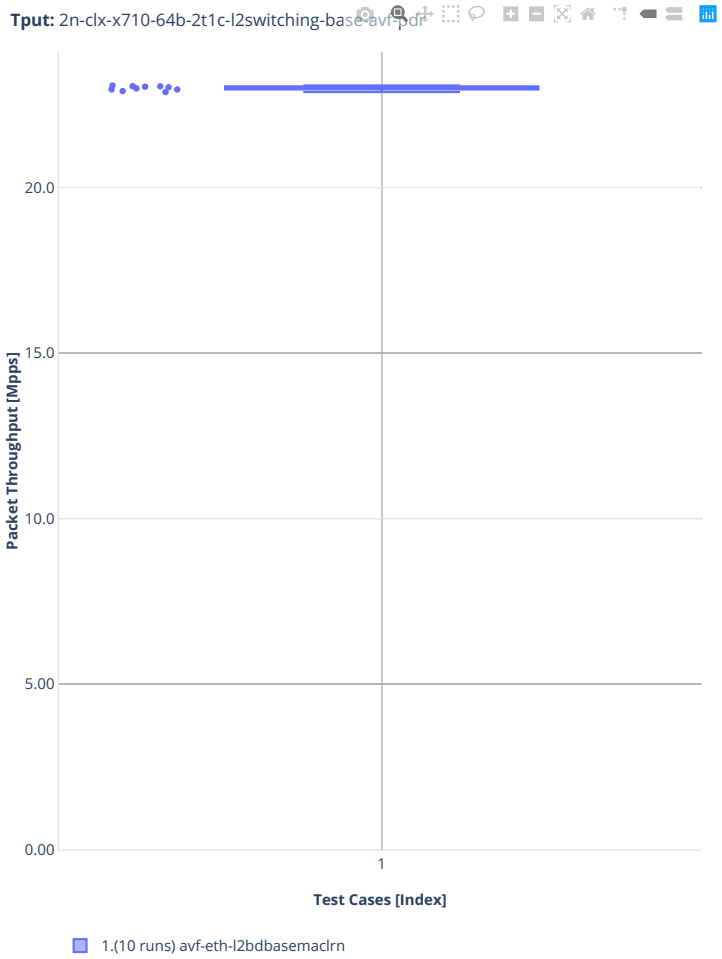



Tput: 2n-clx-xxv710-64b-2t1c-l2switching-base-scale-dpdk-pdr
18.0
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) eth-l2bdbasemaclrn
2.(10 runs) eth-l2bdscale10kmaclrn
3.(10 runs) eth-l2bdscale100kmaclrn
4.(10 runs) eth-l2bdscale1mmaclrn

2n-clx-x710

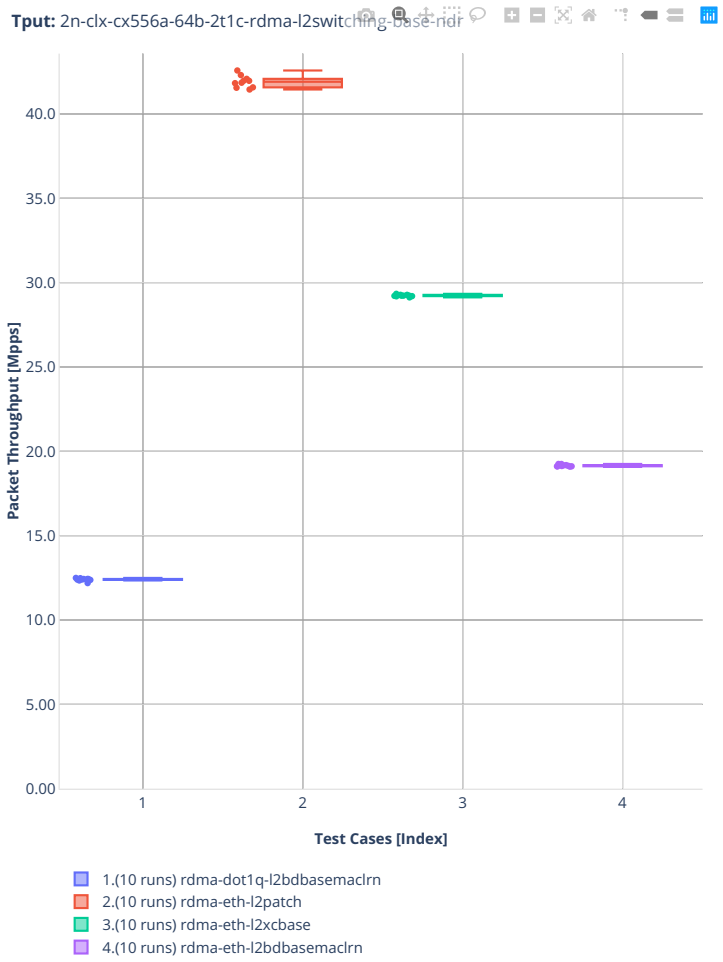
64b-2t1c-l2switching-base-avf

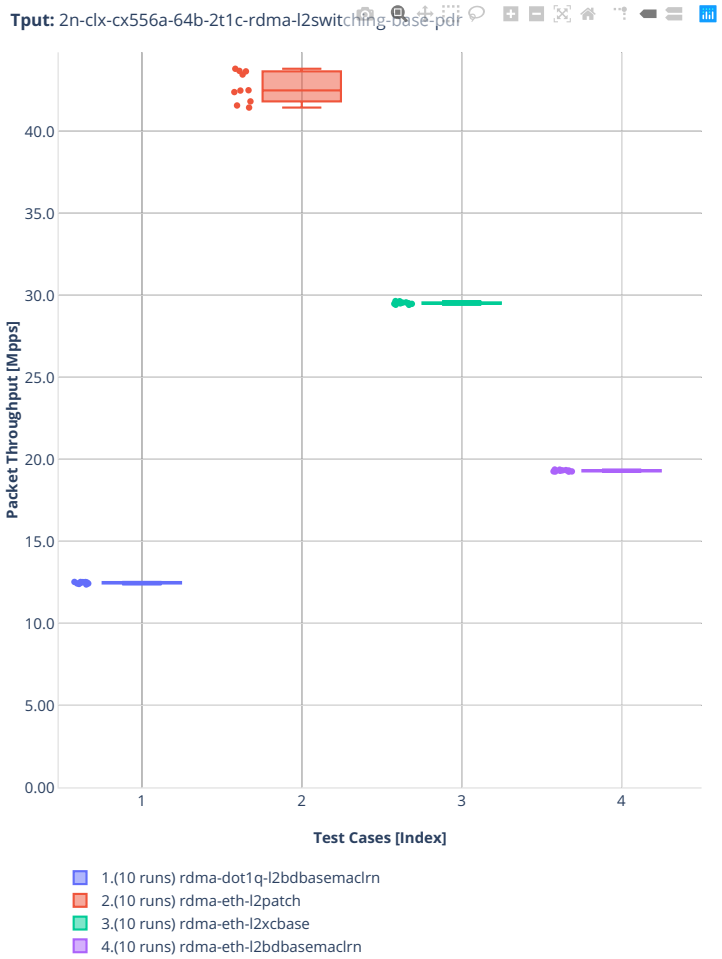



Tput: 2n-clx-x710-64b-2t1c-l2switching-base-avf-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
1.(10 runs) avf-eth-l2bdbasemaclrn

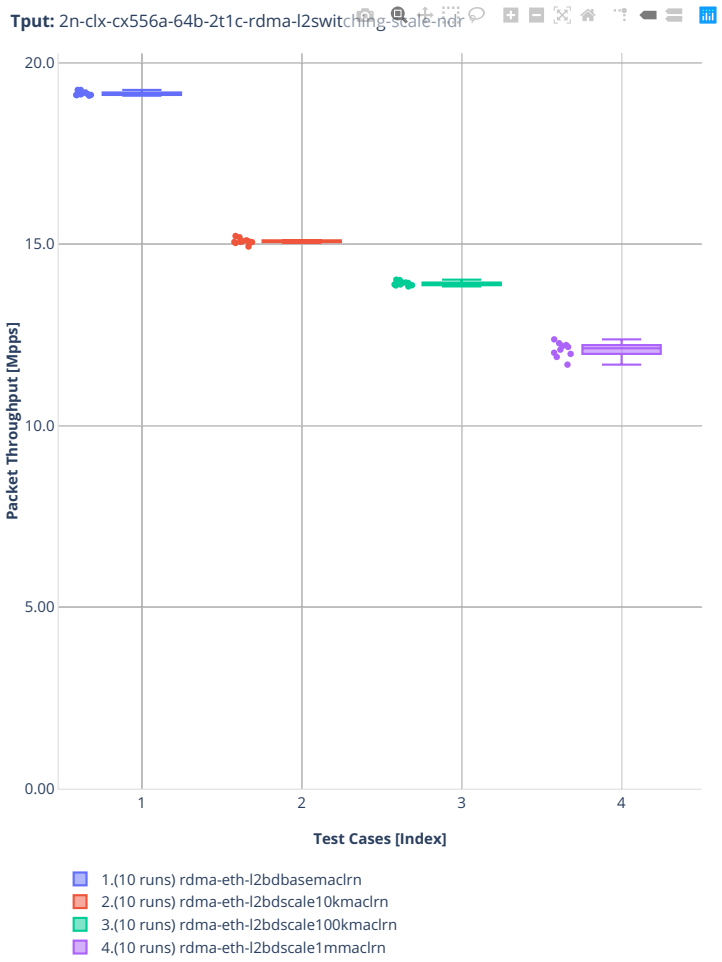
2n-clx-cx556a

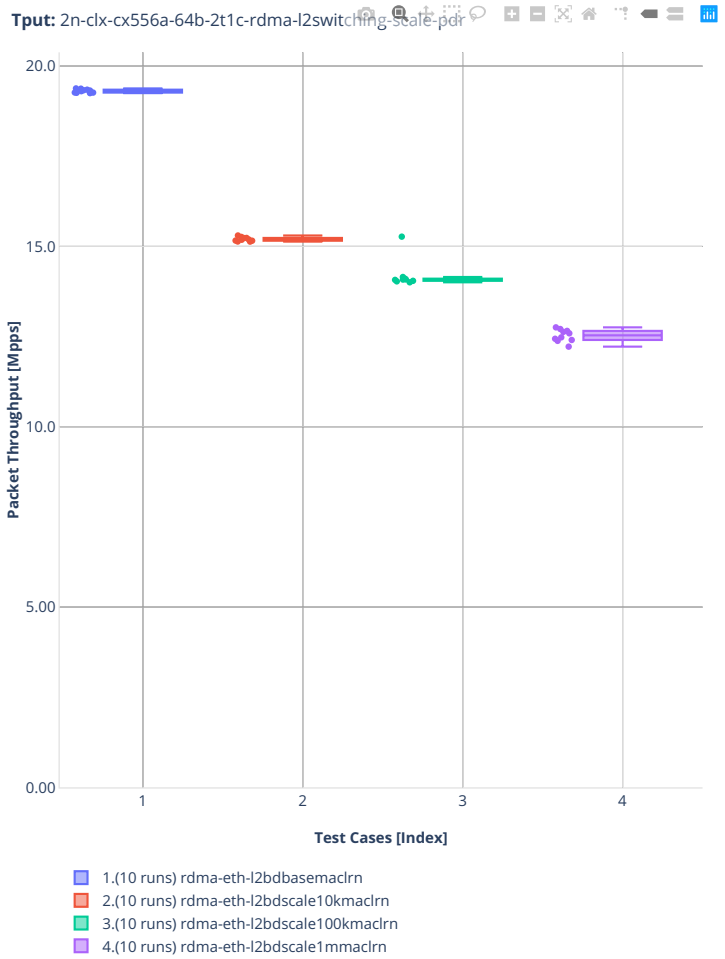
64b-2t1c-l2switching-base-rdma-core




Tput: 2n-clx-cx556a-64b-2t1c-rdma-l2switching-base-pdr
Packet Throughput [Mpps]
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) rdma-dot1q-l2bdbasemaclrn
2.(10 runs) rdma-eth-l2patch
3.(10 runs) rdma-eth-l2xcbase
4.(10 runs) rdma-eth-l2bdbasemaclrn

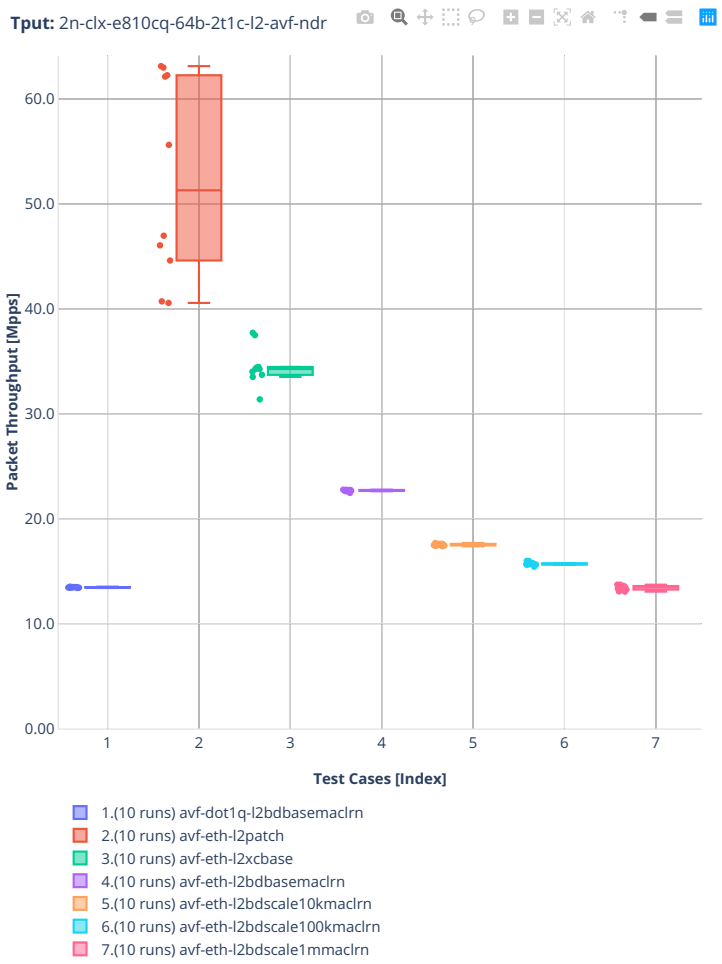
64b-2t1c-l2switching-scale-rdma-core

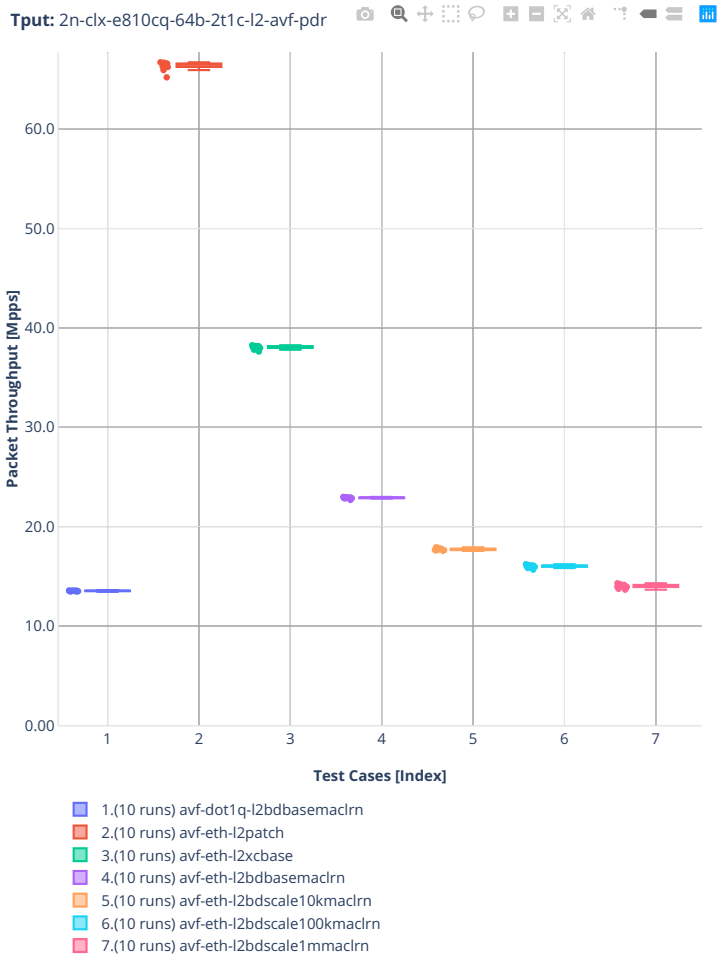



Tput: 2n-clx-cx556a-64b-2t1c-rdma-l2switching-scale-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) rdma-eth-l2bdbasemaclrn
2.(10 runs) rdma-eth-l2bdscale10kmaclrn
3.(10 runs) rdma-eth-l2bdscale100kmaclrn
4.(10 runs) rdma-eth-l2bdscale1mmaclrn

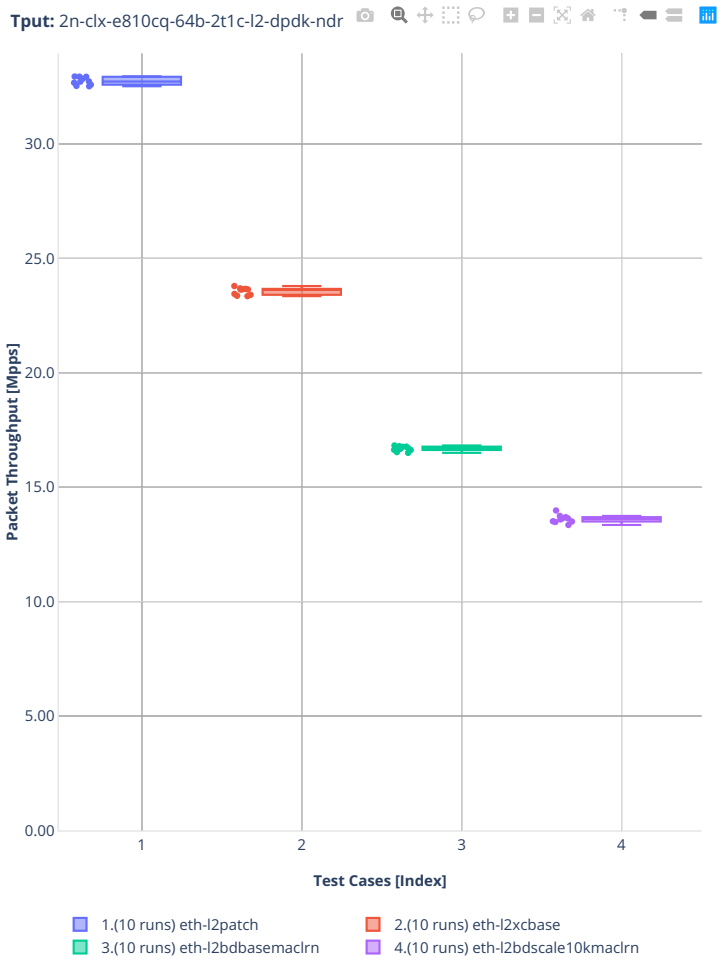
2n-clx-e810cq

64b-2t1c-l2switching-avf




Tput: 2n-clx-e810cq-64b-2t1c-l2-avf-pdr
Packet Throughput [Mpps]
60.0
50.0
40.0
30.0
20.0
10.0
0.00
1
2
3
4
5
6
7
Test Cases [Index]
1.(10 runs) avf-dot1q-l2bdbasemaclrn
2.(10 runs) avf-eth-l2patch
3.(10 runs) avf-eth-l2xcbase
4.(10 runs) avf-eth-l2bdbasemaclrn
5.(10 runs) avf-eth-l2bdscale10kmaclrn
6.(10 runs) avf-eth-l2bdscale100kmaclrn
7.(10 runs) avf-eth-l2bdscale1mmaclrn

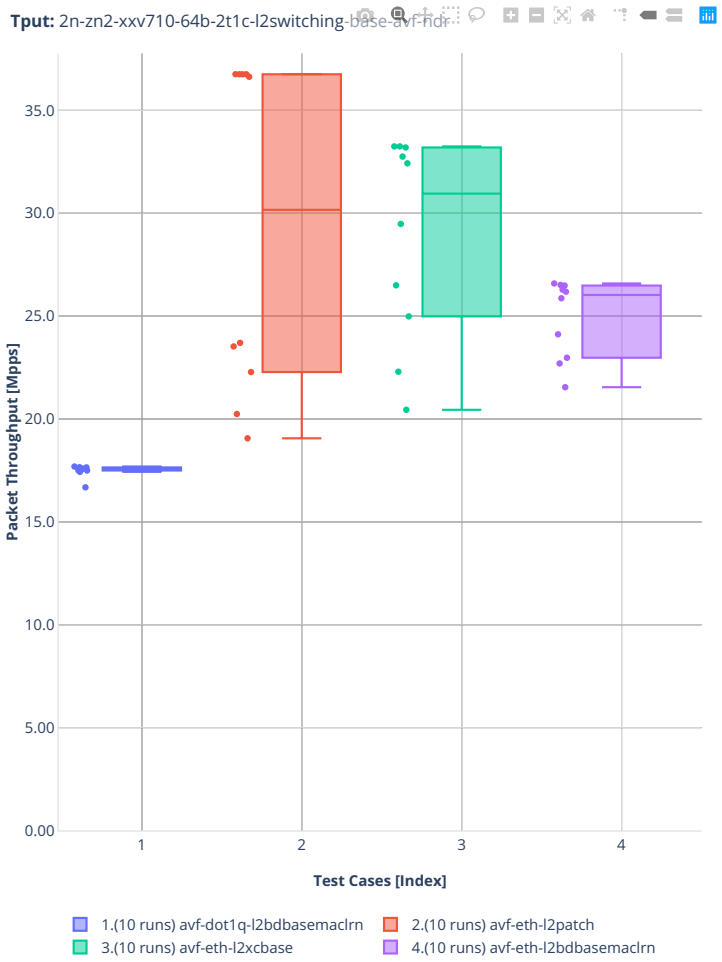
64b-2t1c-l2switching-dpdk

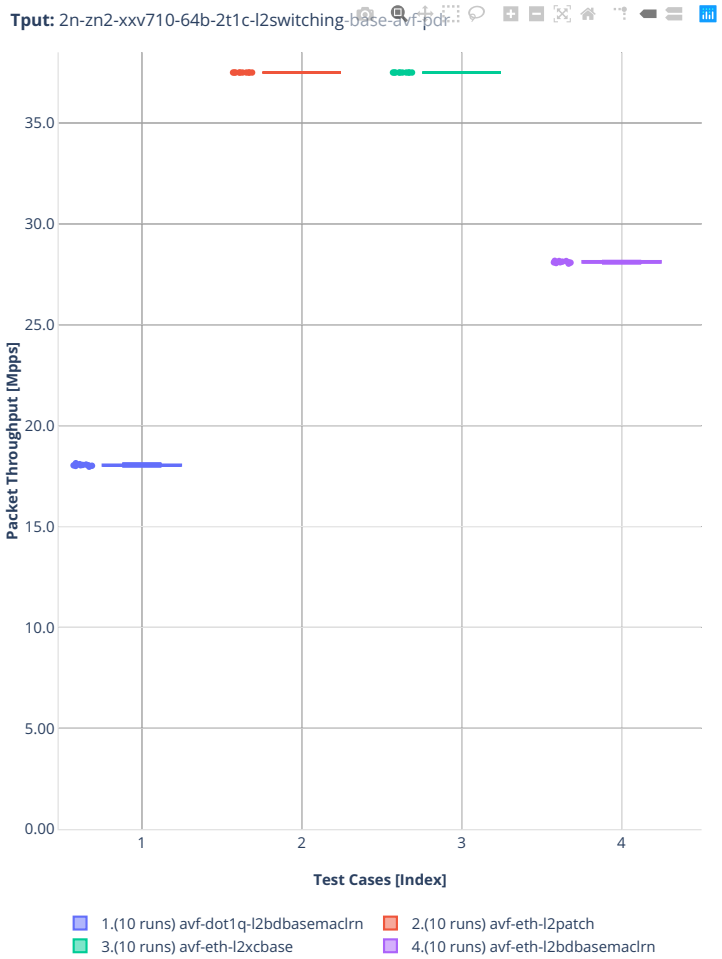



Tput: 2n-clx-e810cq-64b-2t1c-l2-dpdk-pdr
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) eth-l2patch
2.(10 runs) eth-l2xcbase
3.(10 runs) eth-l2bdbasemaclrn
4.(10 runs) eth-l2bdscale10kmaclrn

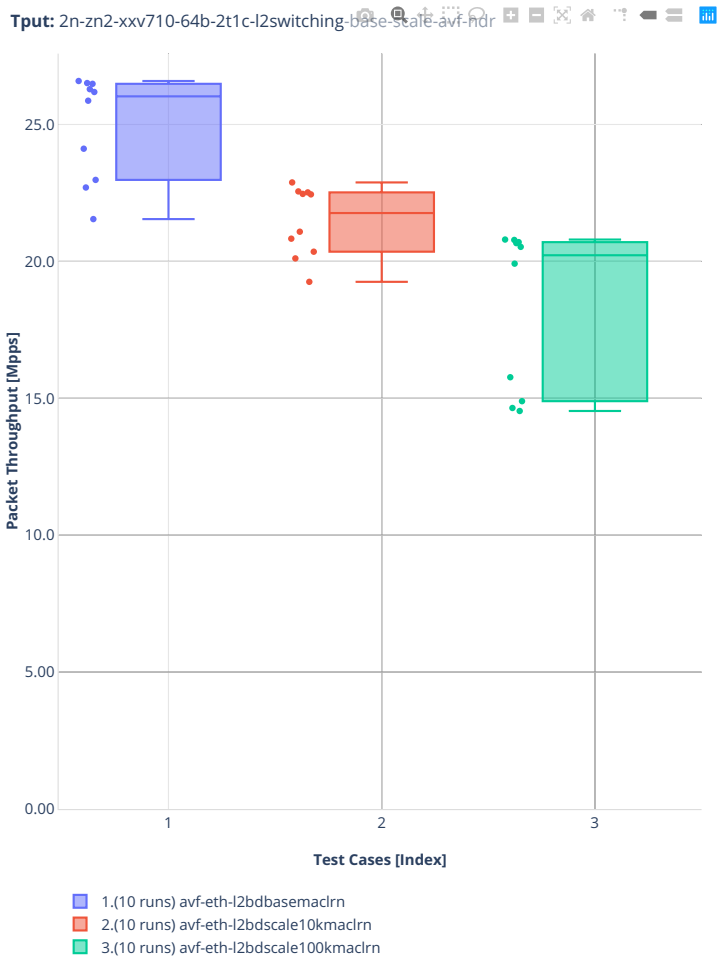
2n-zn2-xxv710

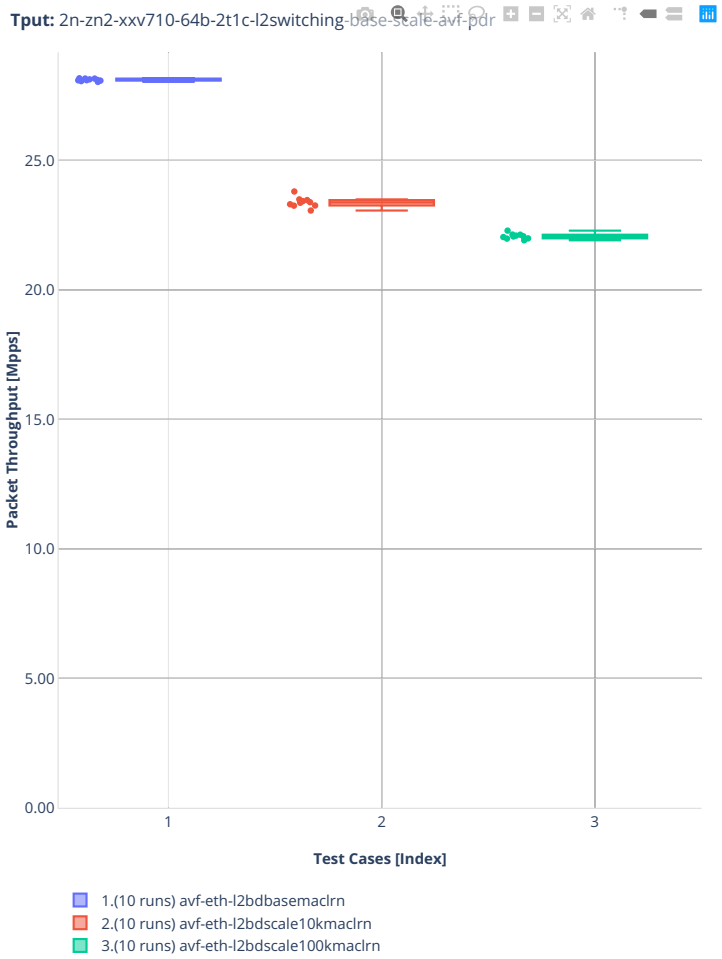
64b-2t1c-l2switching-base-avf



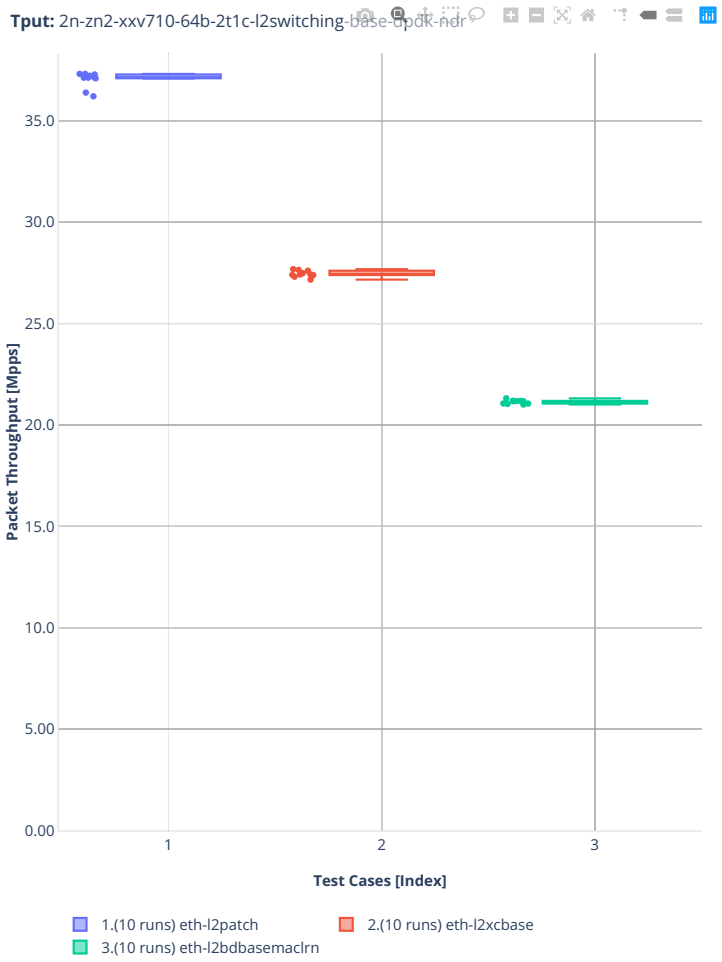

Tput: 2n-zn2-xxv710-64b-2t1c-l2switching-base-avf-pdr
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-dot1q-l2bdbasemaclrn
2.(10 runs) avf-eth-l2patch
3.(10 runs) avf-eth-l2xcbase
4.(10 runs) avf-eth-l2bdbasemaclrn

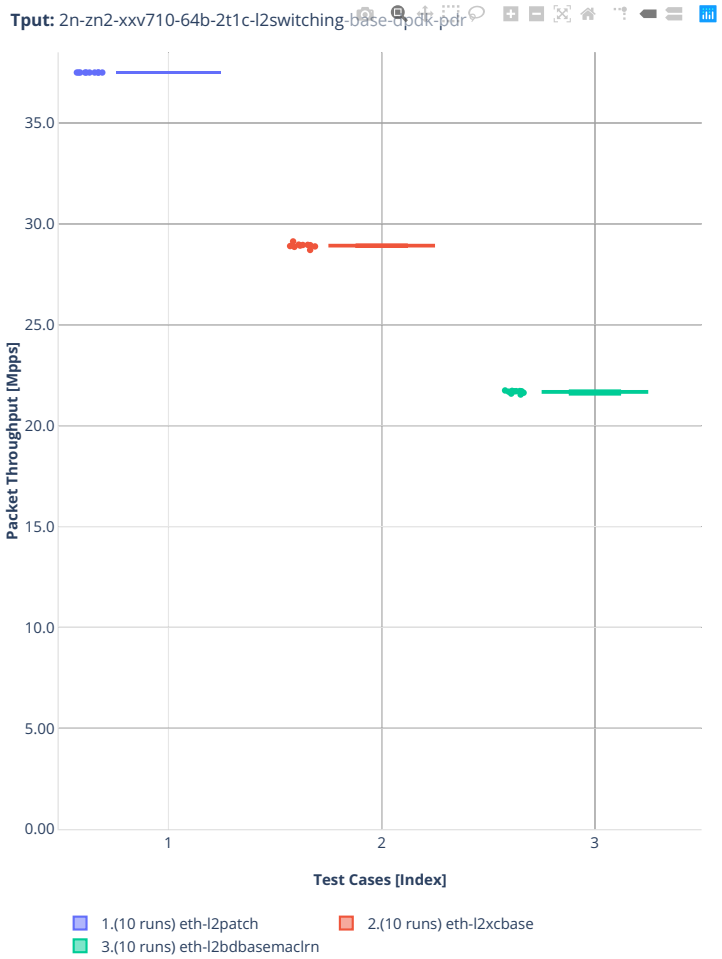
64b-2t1c-l2switching-base-scale-avf



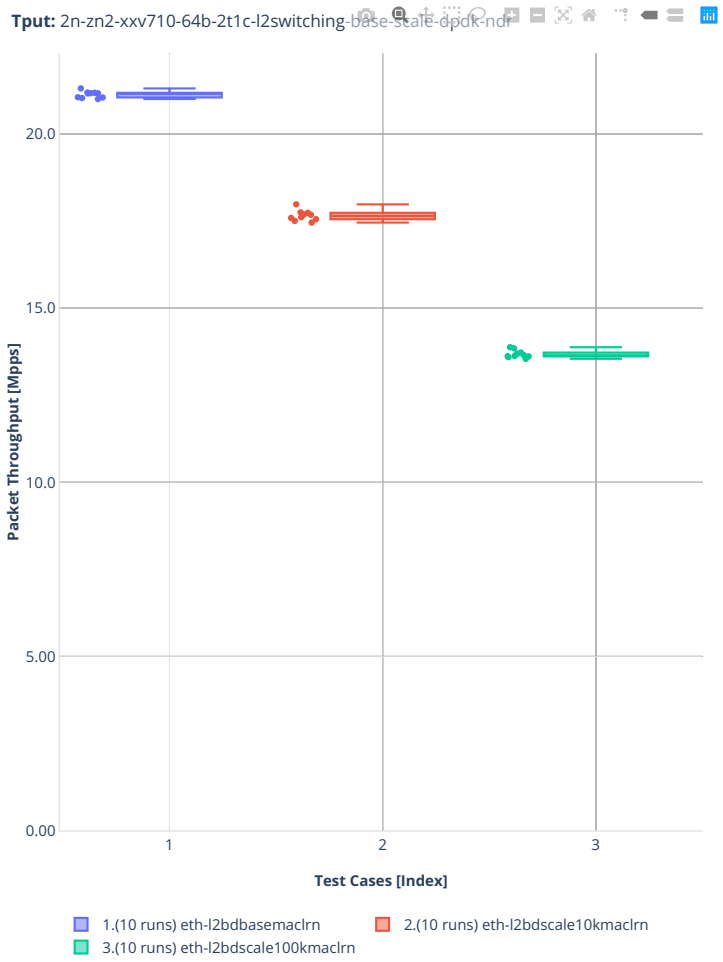

Tput: 2n-zn2-xxv710-64b-2t1c-l2switching-base-scale-avf-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-eth-l2bdbasemaclrn
2.(10 runs) avf-eth-l2bdscale10kmaclrn
3.(10 runs) avf-eth-l2bdscale100kmaclrn

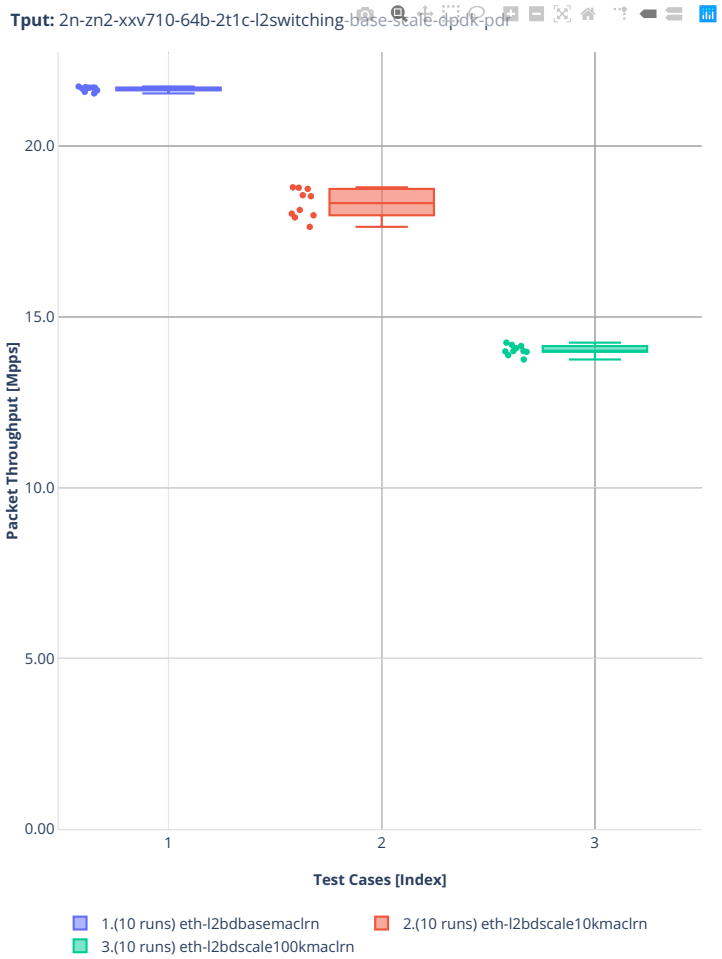
64b-2t1c-l2switching-base-dpdk




Tput: 2n-zn2-xxv710-64b-2t1c-l2switching-base-dpdk-pdr
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) eth-l2patch
2.(10 runs) eth-l2xcbase
3.(10 runs) eth-l2bdbasemaclrn

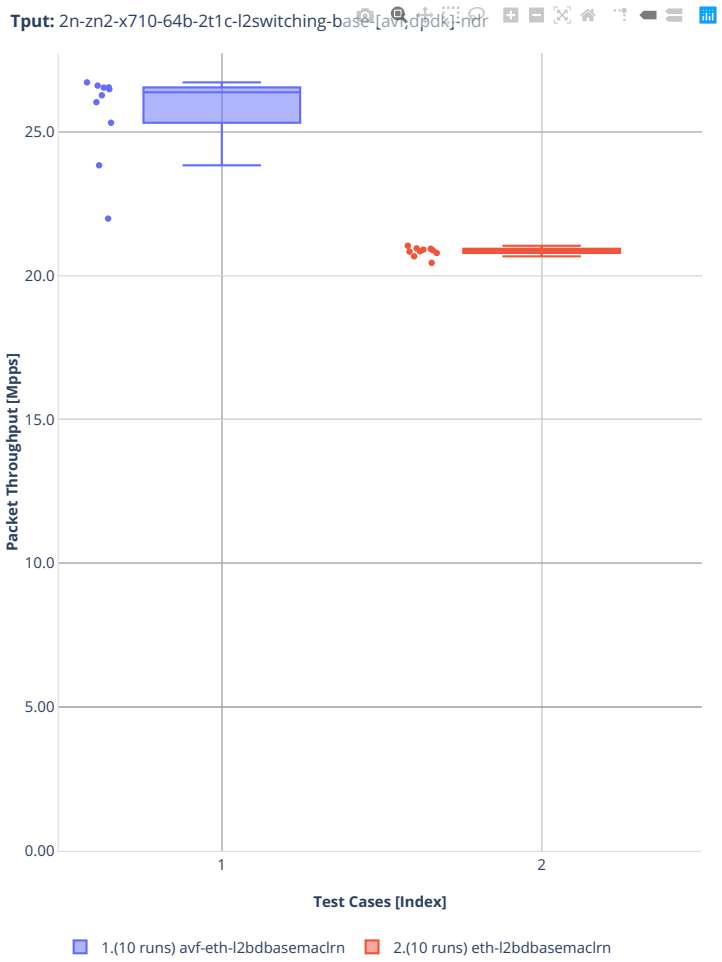
64b-2t1c-l2switching-base-scale-dpdk

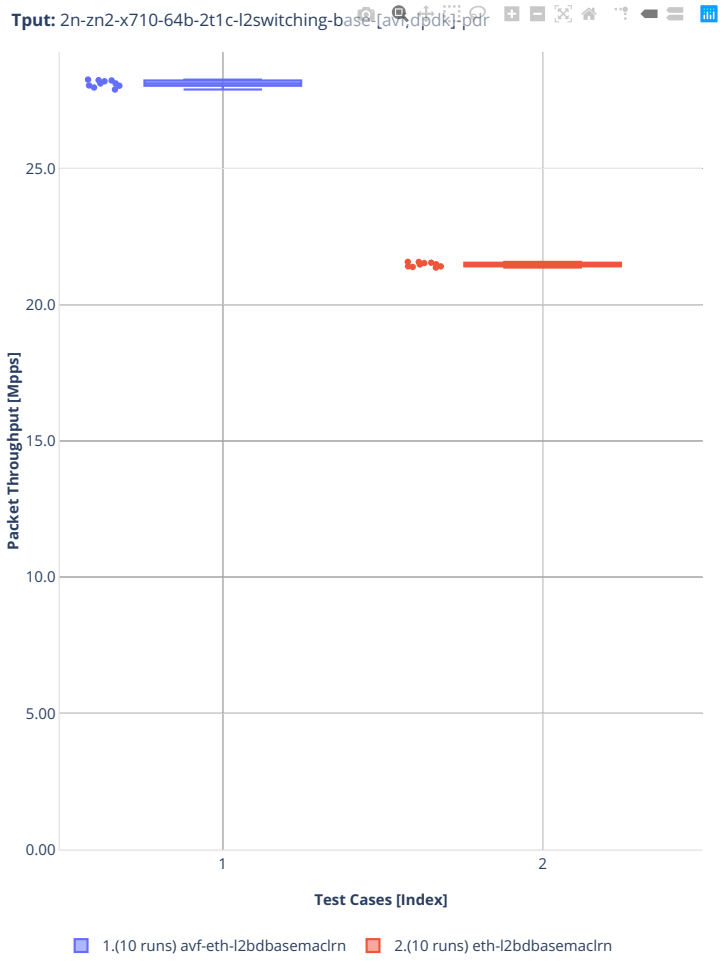



Tput: 2n-zn2-xxv710-64b-2t1c-l2switching-base-scale-dpdk-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) eth-l2bdbasemaclrn
2.(10 runs) eth-l2bdscale10kmaclrn
3.(10 runs) eth-l2bdscale100kmaclrn

2n-zn2-x710

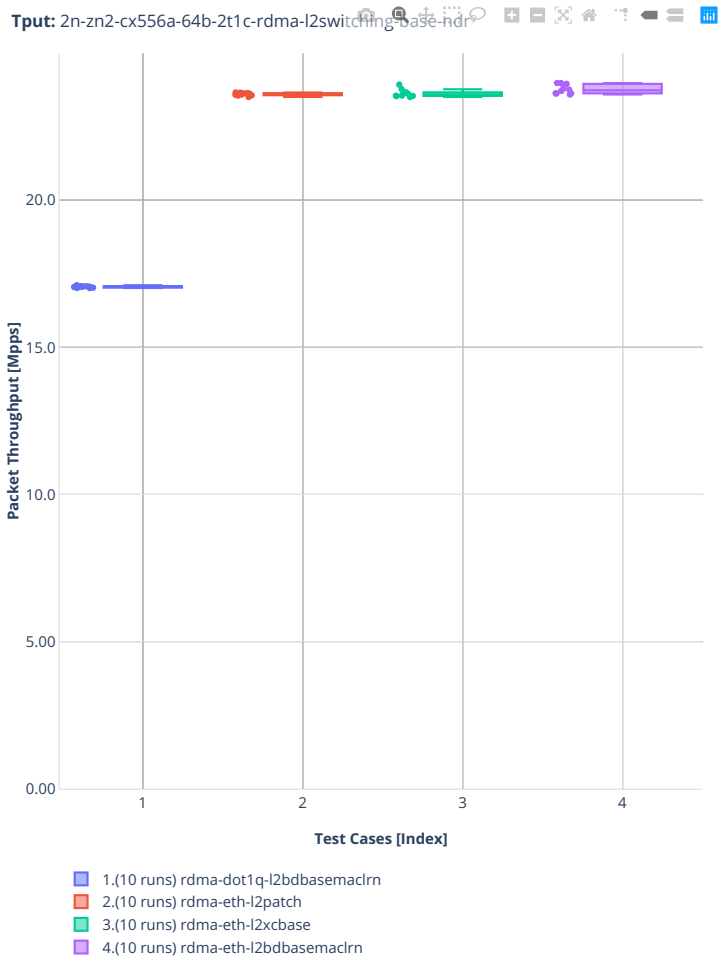
64b-2t1c-l2switching-base

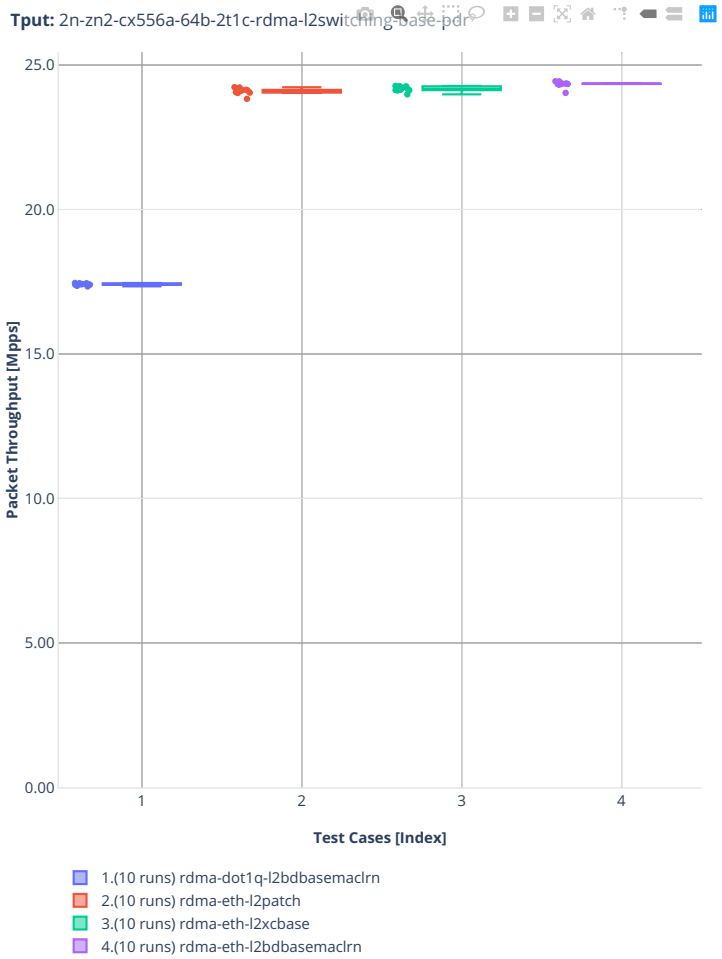



Tput: 2n-zn2-x710-64b-2t1c-l2switching-base-[avf,dpdk]-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) avf-eth-l2bdbasemaclrn
2.(10 runs) eth-l2bdbasemaclrn

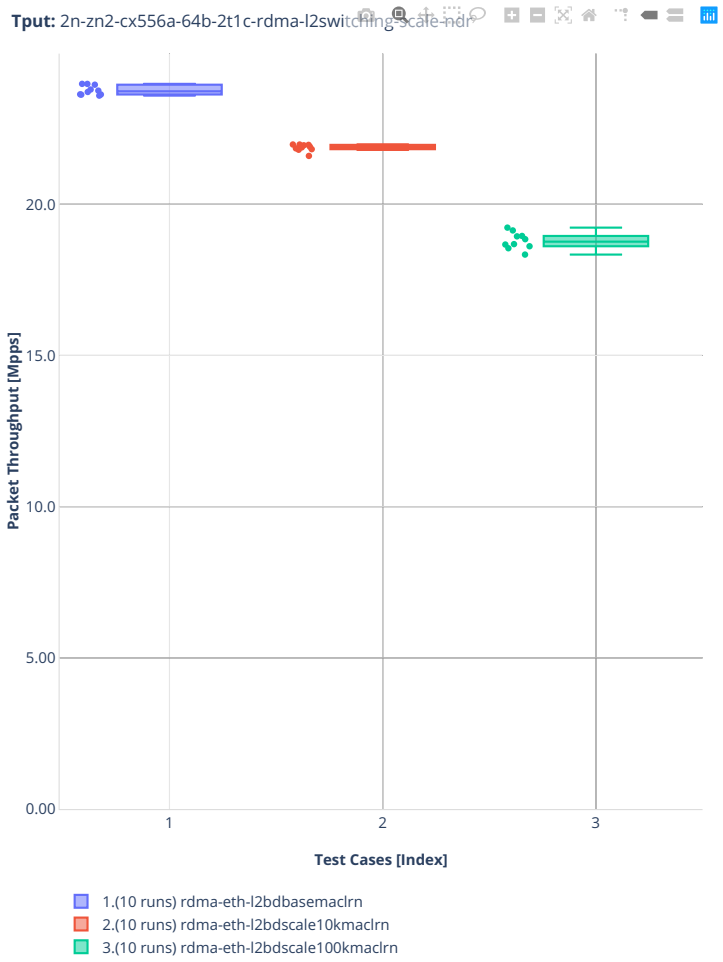
2n-zn2-cx556a

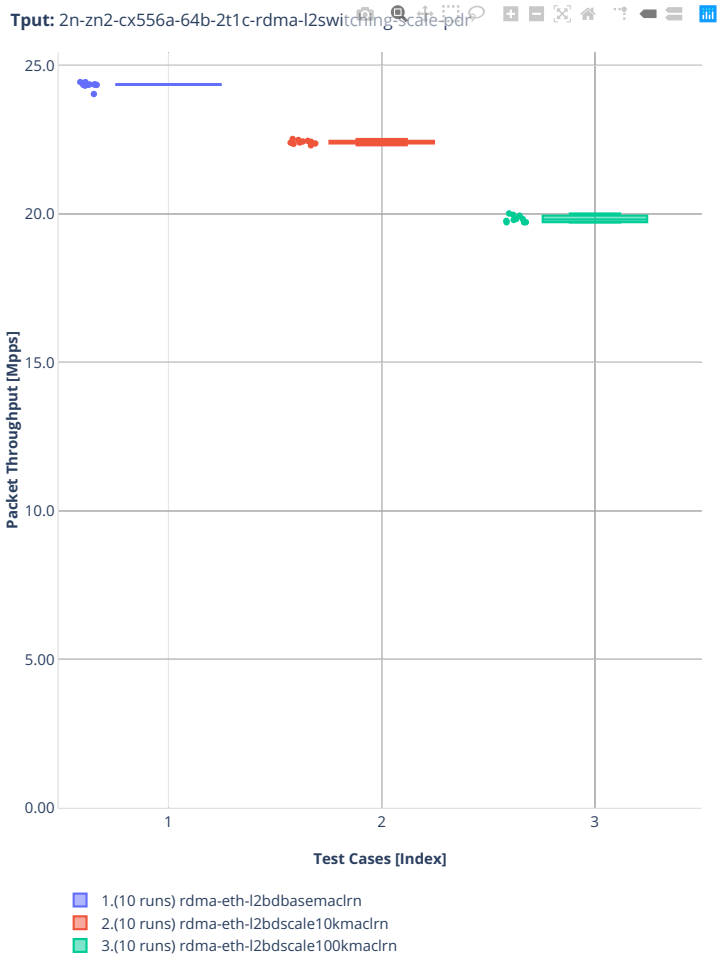
64b-2t1c-l2switching-base-rdma-core




Tput: 2n-zn2-cx556a-64b-2t1c-rdma-l2switching-base-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) rdma-dot1q-l2bdbasemaclrn
2.(10 runs) rdma-eth-l2patch
3.(10 runs) rdma-eth-l2xcbase
4.(10 runs) rdma-eth-l2bdbasemaclrn

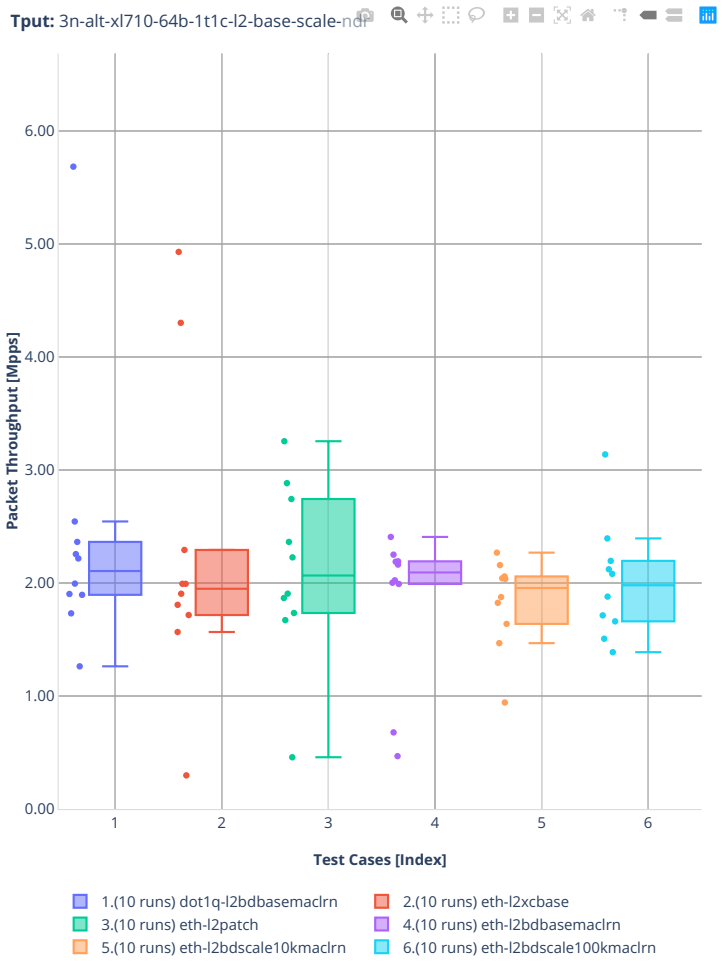
64b-2t1c-l2switching-scale-rdma-core

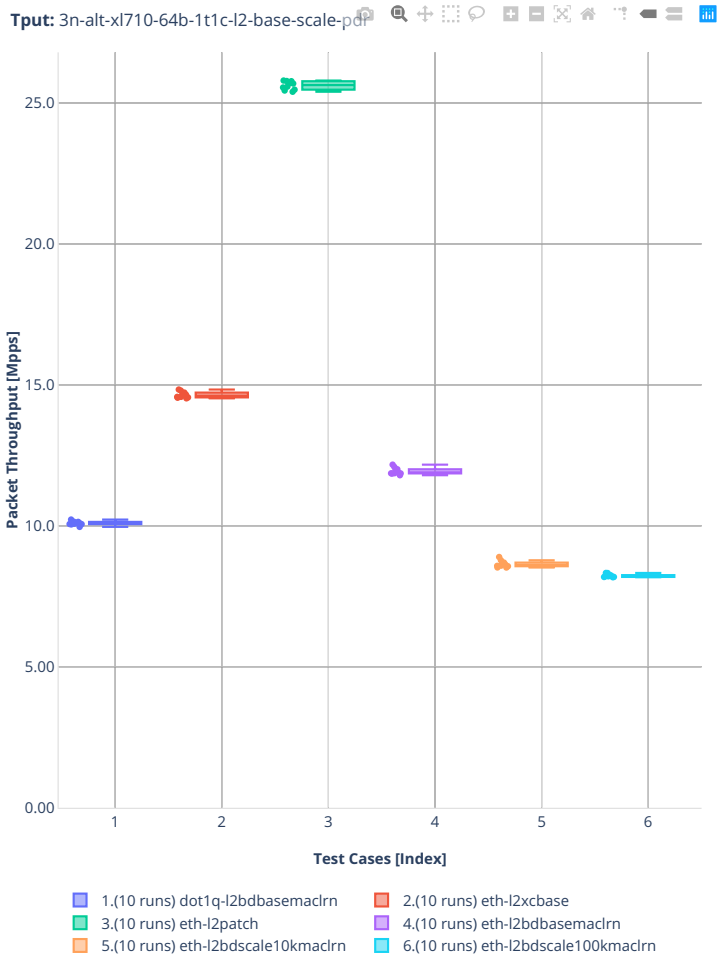



Tput: 2n-zn2-cx556a-64b-2t1c-rdma-l2switching-scale-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) rdma-eth-l2bdbasemaclrn
2.(10 runs) rdma-eth-l2bdscale10kmaclrn
3.(10 runs) rdma-eth-l2bdscale100kmaclrn

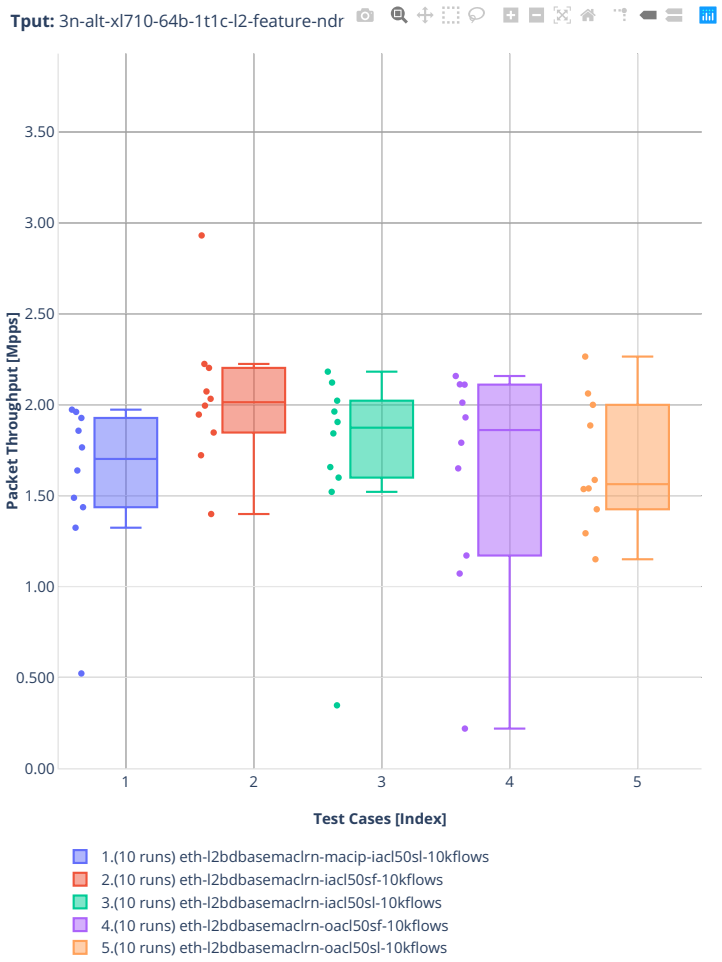
3n-alt-xl710

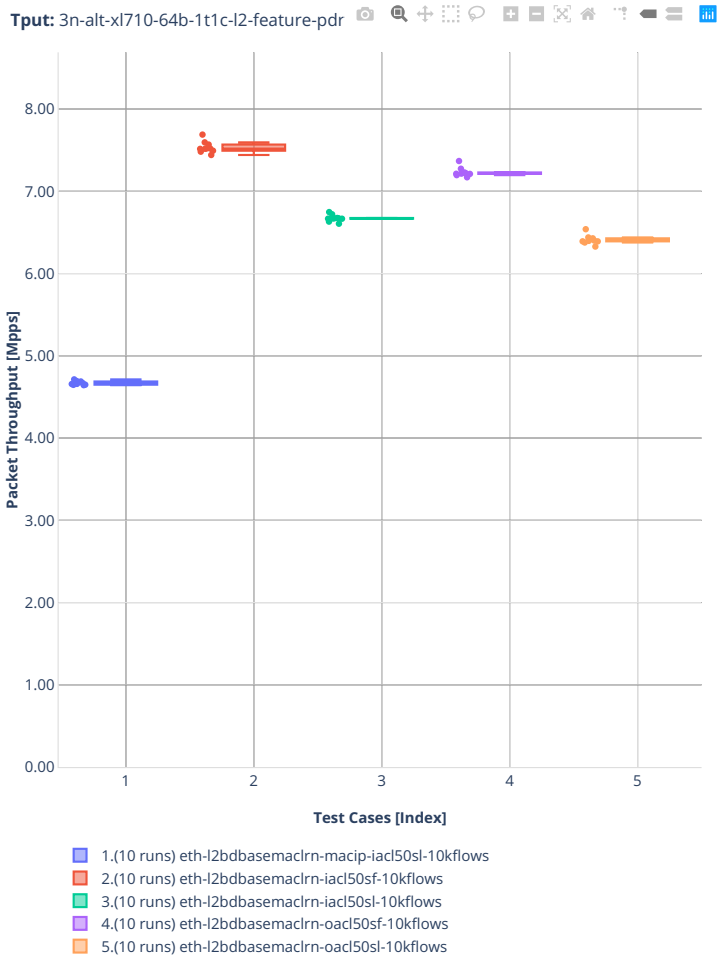
64b-1t1c-l2switching-base-scale




Tput: 3n-alt-xl710-64b-1t1c-l2-base-scale-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) dot1q-l2bdbasemaclrn
2.(10 runs) eth-l2xcbase
3.(10 runs) eth-l2patch
4.(10 runs) eth-l2bdbasemaclrn
5.(10 runs) eth-l2bdscale10kmaclrn
6.(10 runs) eth-l2bdscale100kmaclrn

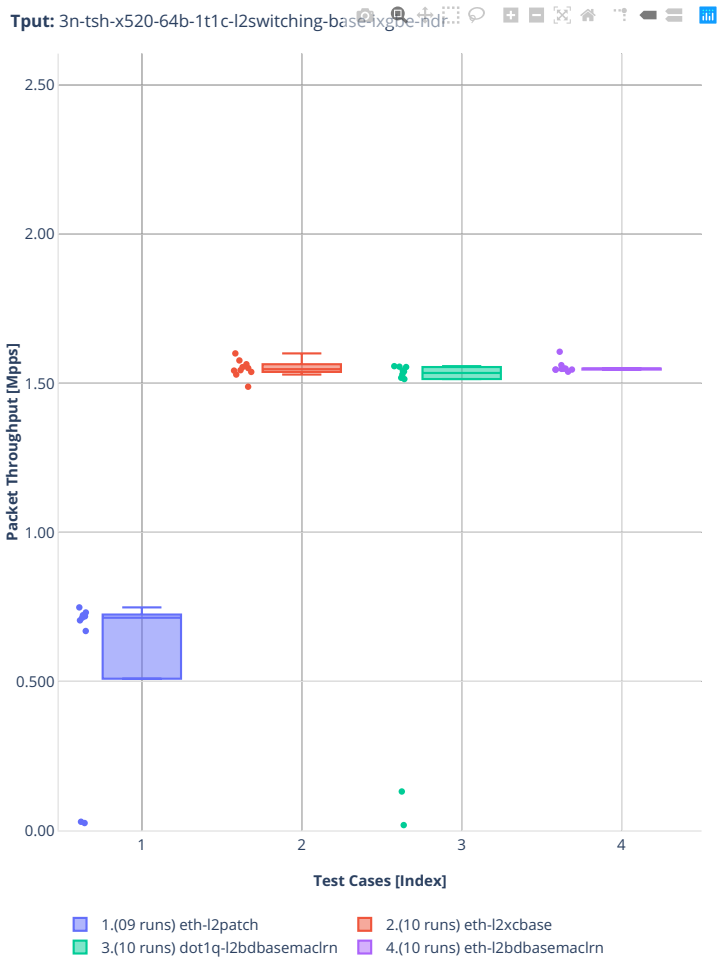
64b-1t1c-l2switching-features

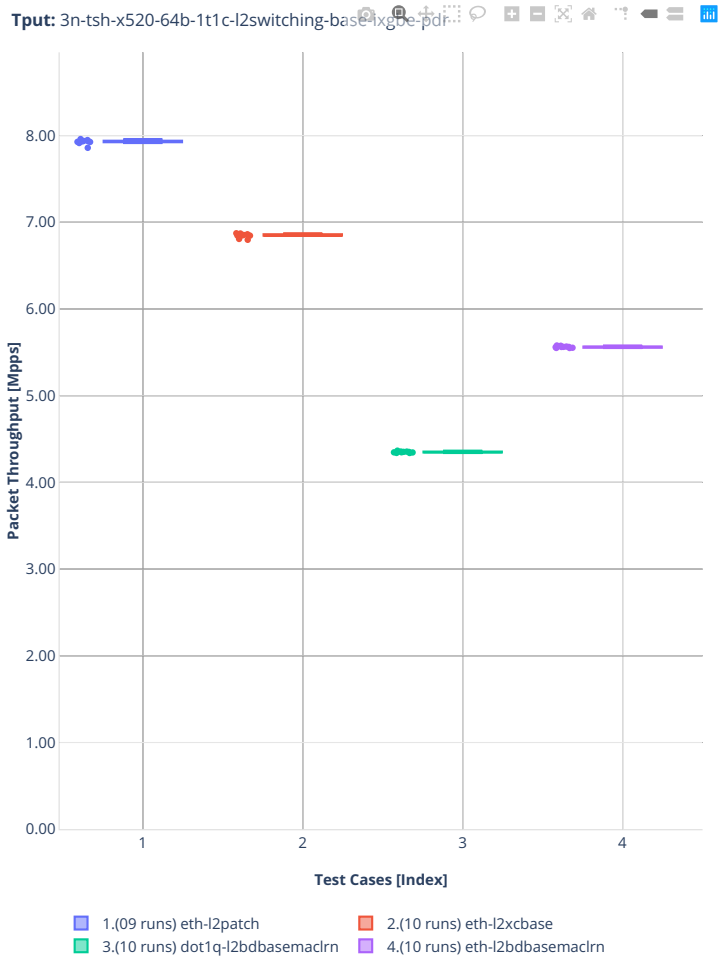



Tput: 3n-alt-xl710-64b-1t1c-l2-feature-pdr
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) eth-l2bdbasemaclrn-macip-iacl50sl-10kflows
2.(10 runs) eth-l2bdbasemaclrn-iacl50sf-10kflows
3.(10 runs) eth-l2bdbasemaclrn-iacl50sl-10kflows
4.(10 runs) eth-l2bdbasemaclrn-oacl50sf-10kflows
5.(10 runs) eth-l2bdbasemaclrn-oacl50sl-10kflows

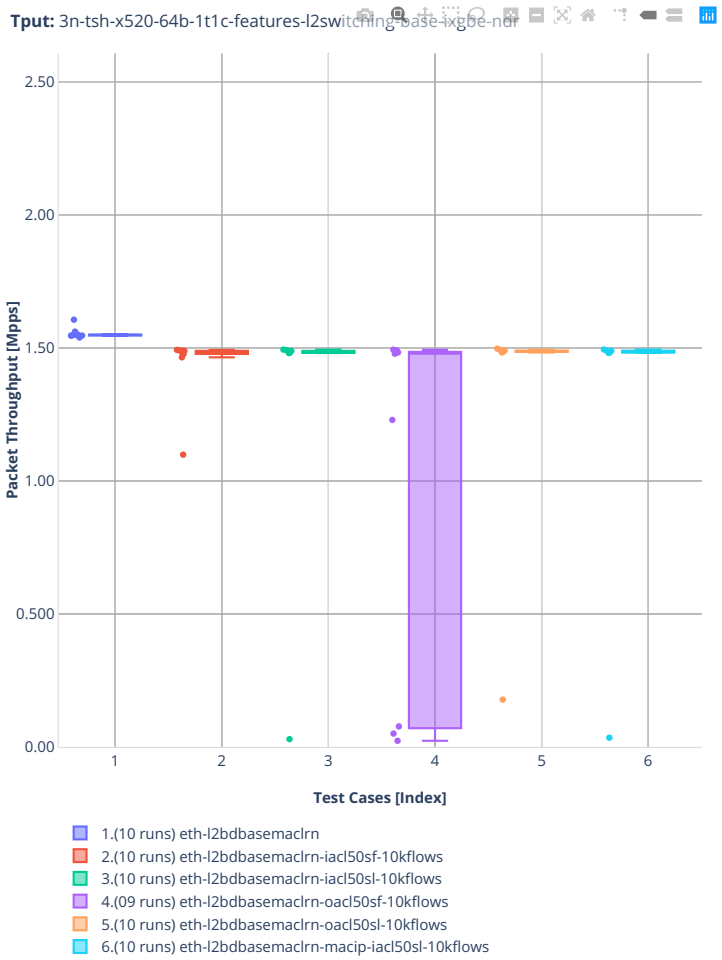
3n-tsh-x520

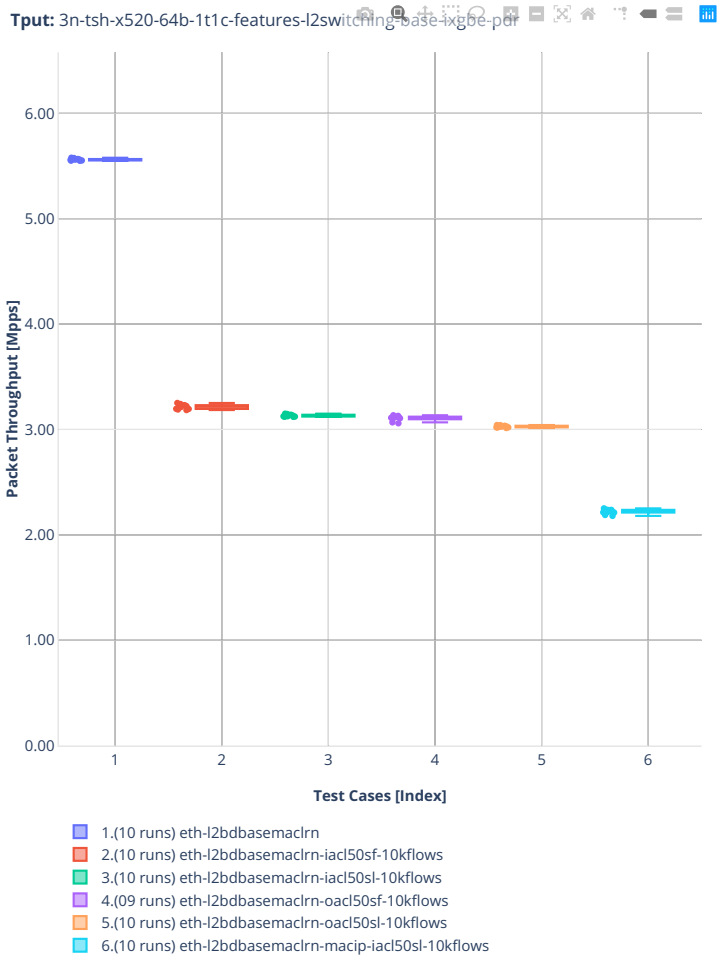
64b-1t1c-l2switching-base-ixgbe




Tput: 3n-tsh-x520-64b-1t1c-l2switching-base-1xgbe-pdr
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(09 runs) eth-l2patch
2.(10 runs) eth-l2xcbase
3.(10 runs) dot1q-l2bdbasemaclrn
4.(10 runs) eth-l2bdbasemaclrn

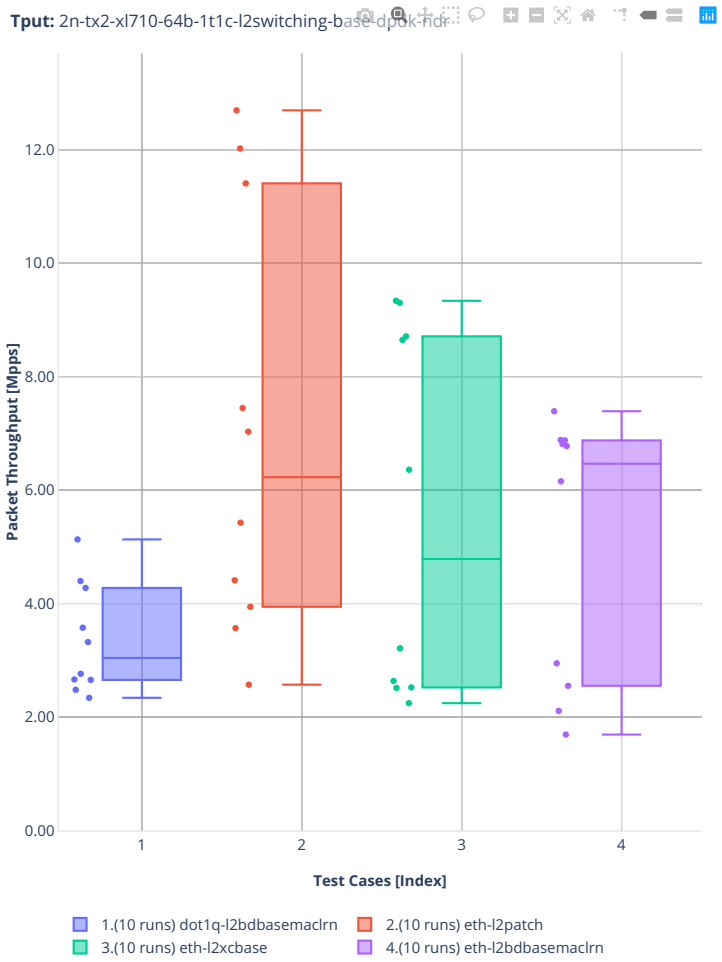
64b-1t1c-features-l2switching-base-ixgbe

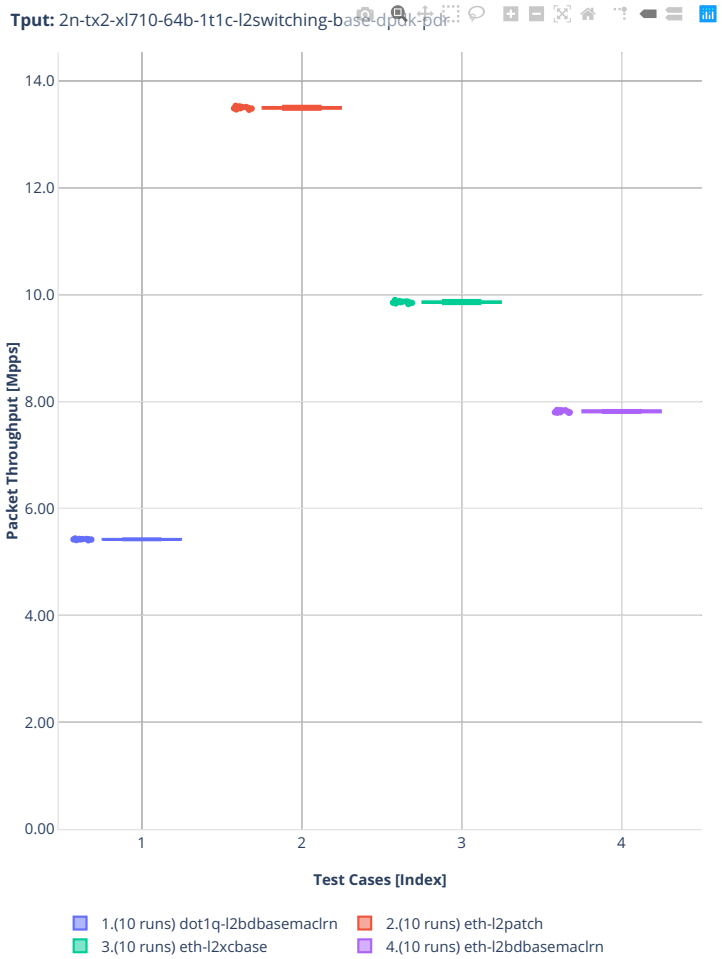



Tput: 3n-tsh-x520-64b-1t1c-features-l2switching-base-ixgbe-pdr
Packet Throughput [Mpps]
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) eth-l2bdbasemaclrn
2.(10 runs) eth-l2bdbasemaclrn-iacl50sf-10kflows
3.(10 runs) eth-l2bdbasemaclrn-iacl50sl-10kflows
4.(09 runs) eth-l2bdbasemaclrn-oacl50sf-10kflows
5.(10 runs) eth-l2bdbasemaclrn-oacl50sl-10kflows
6.(10 runs) eth-l2bdbasemaclrn-macip-iacl50sl-10kflows

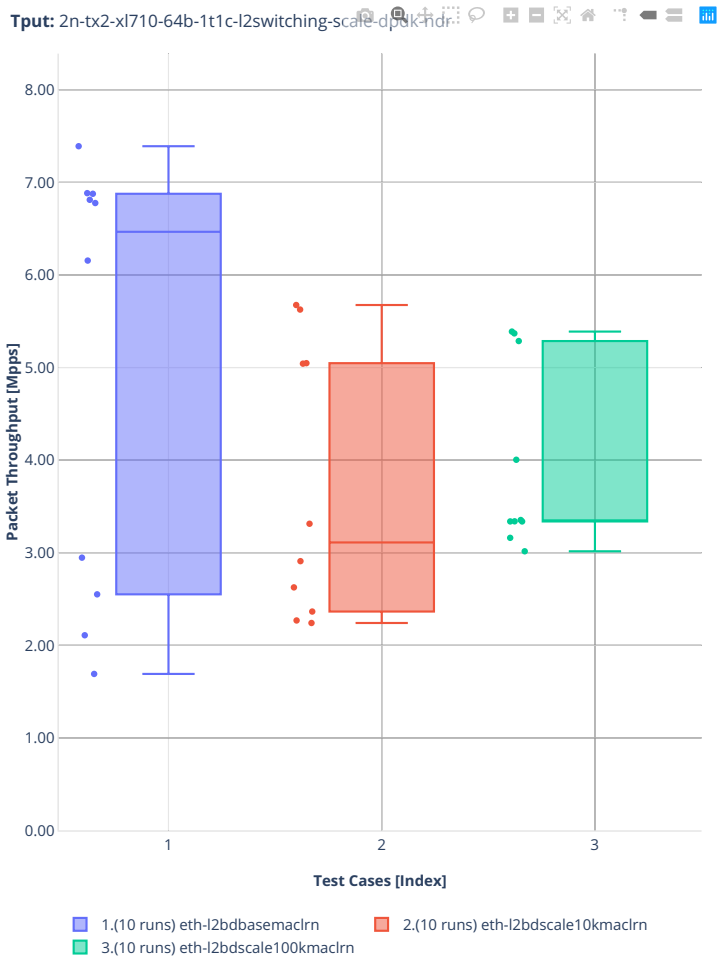
2n-tx2-xl710

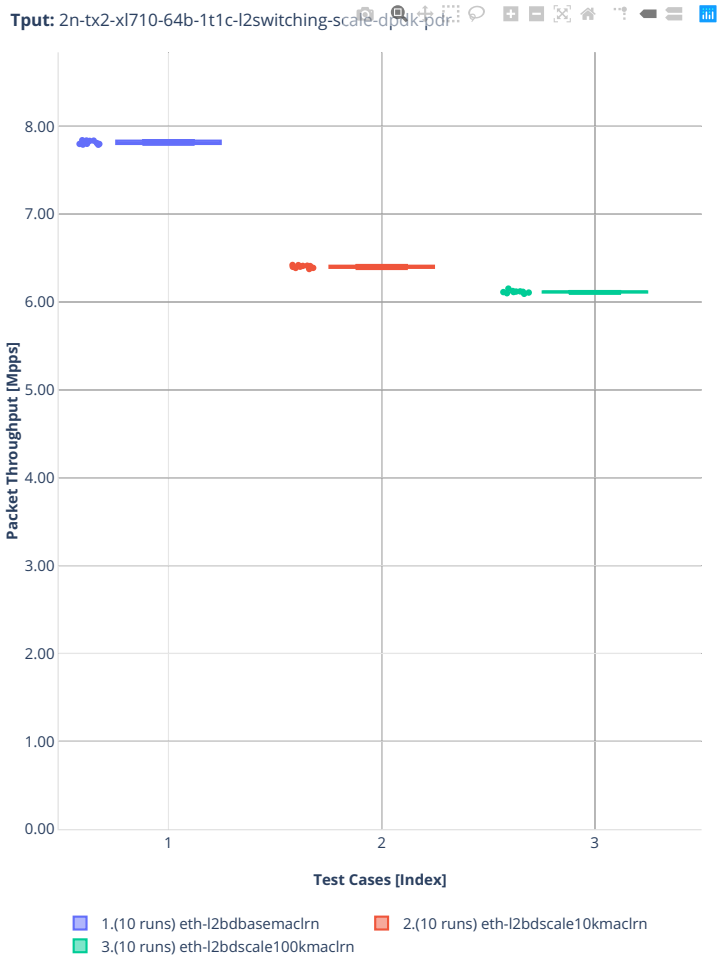
64b-1t1c-l2switching-base-dpdk



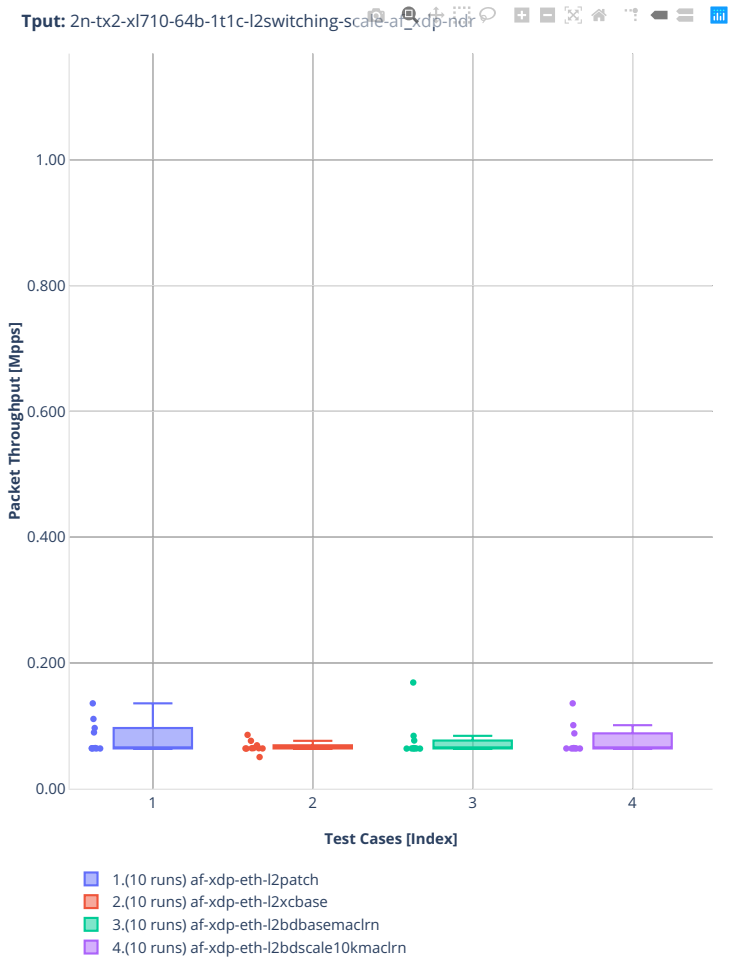

Tput: 2n-tx2-xl710-64b-1t1c-l2switching-base-dpdk-pdr
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) dot1q-l2bdbasemaclrn
2.(10 runs) eth-l2patch
3.(10 runs) eth-l2xcbase
4.(10 runs) eth-l2bdbasemaclrn

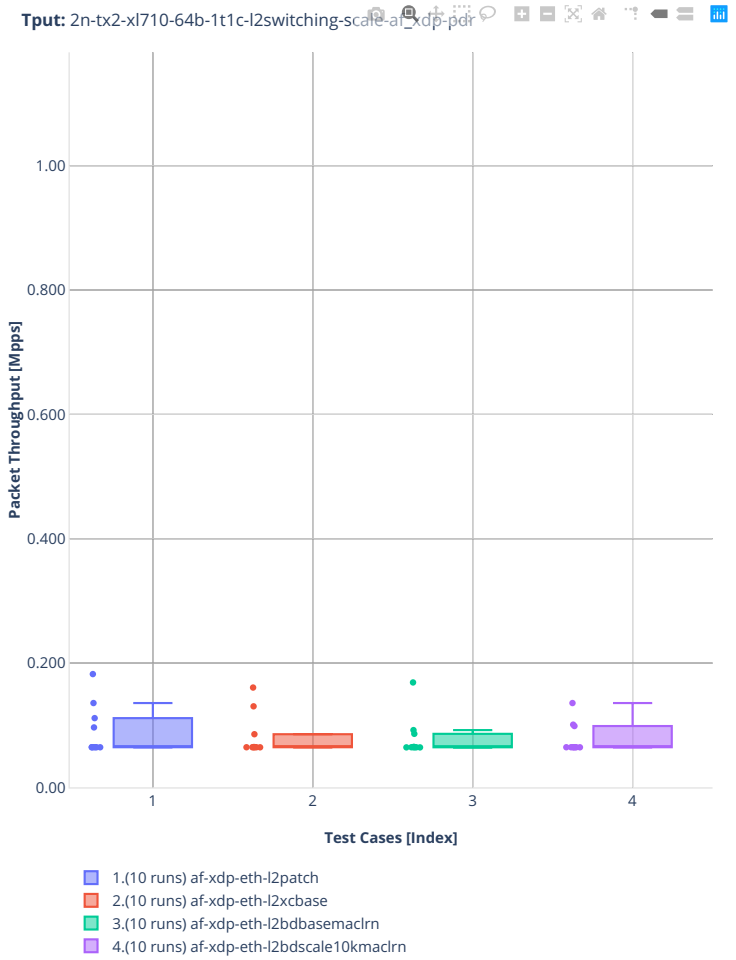
64b-1t1c-l2switching-scale-dpdk



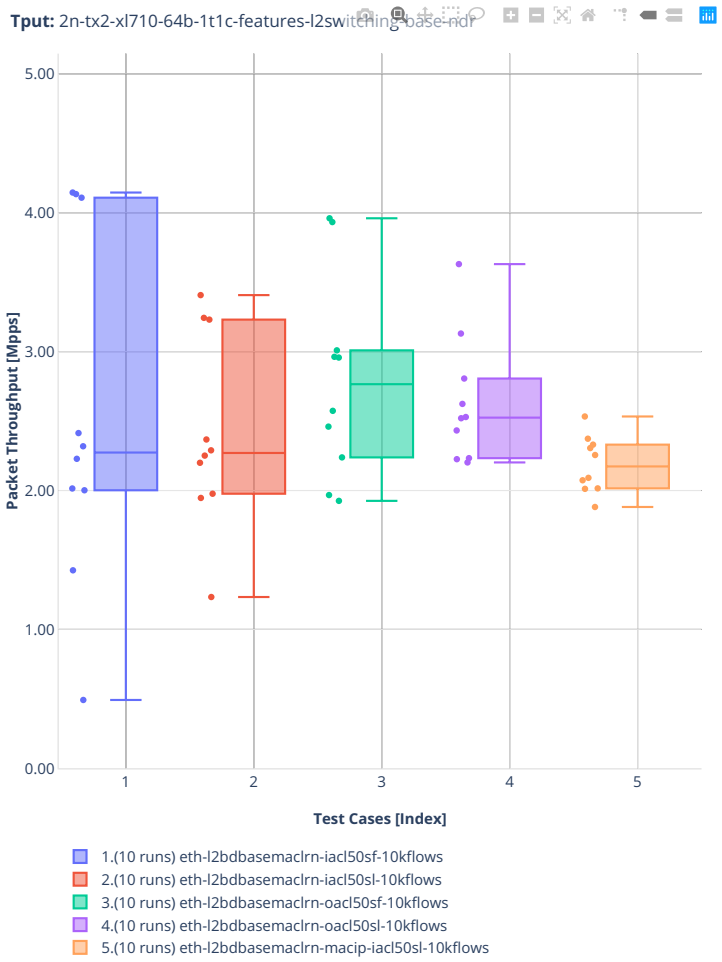

Tput: 2n-tx2-xl710-64b-1t1c-l2switching-scale-dpdk-pdr
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) eth-l2bdbasemaclrn
2.(10 runs) eth-l2bdscale10kmaclrn
3.(10 runs) eth-l2bdscale100kmaclrn

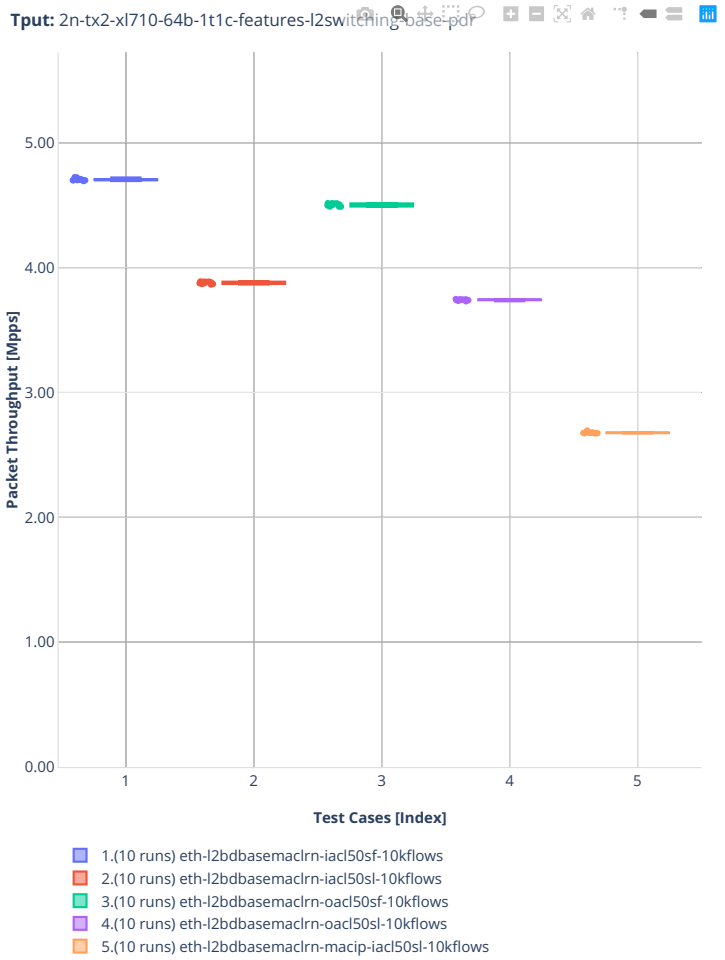
64b-1t1c-l2switching-scale-af-xdp




Tput: 2n-tx2-xl710-64b-1t1c-l2switching-scale-af_xdp-pdr
Packet Throughput [Mpps]
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) af-xdp-eth-l2patch
2.(10 runs) af-xdp-eth-l2xcbase
3.(10 runs) af-xdp-eth-l2bdbasemaclrn
4.(10 runs) af-xdp-eth-l2bdscale10kmaclrn

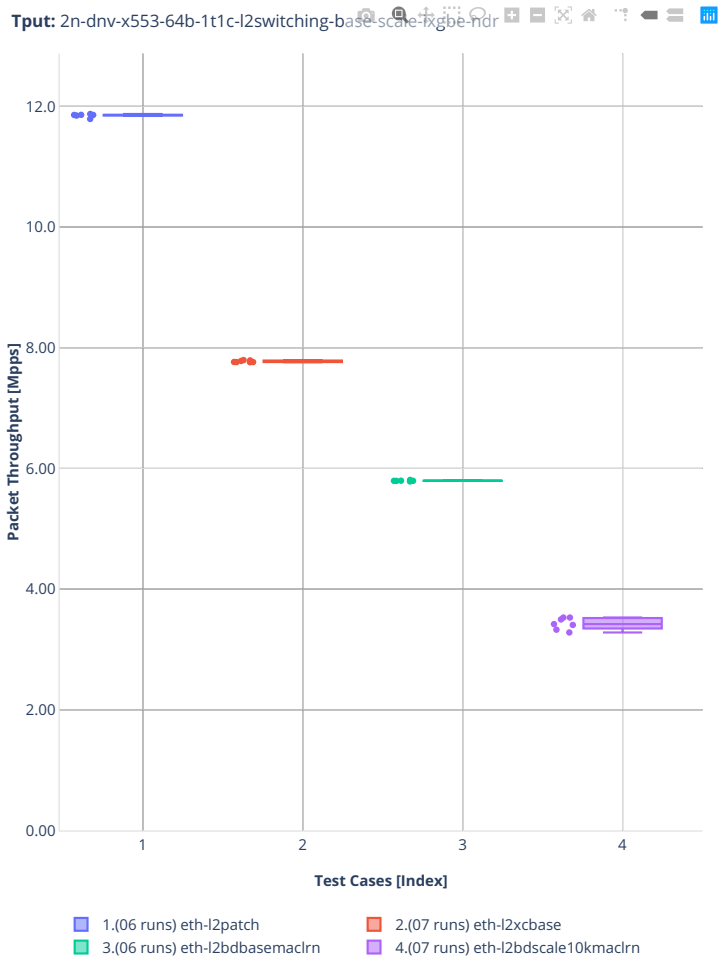
64b-1t1c-features-l2switching-base-dpdk

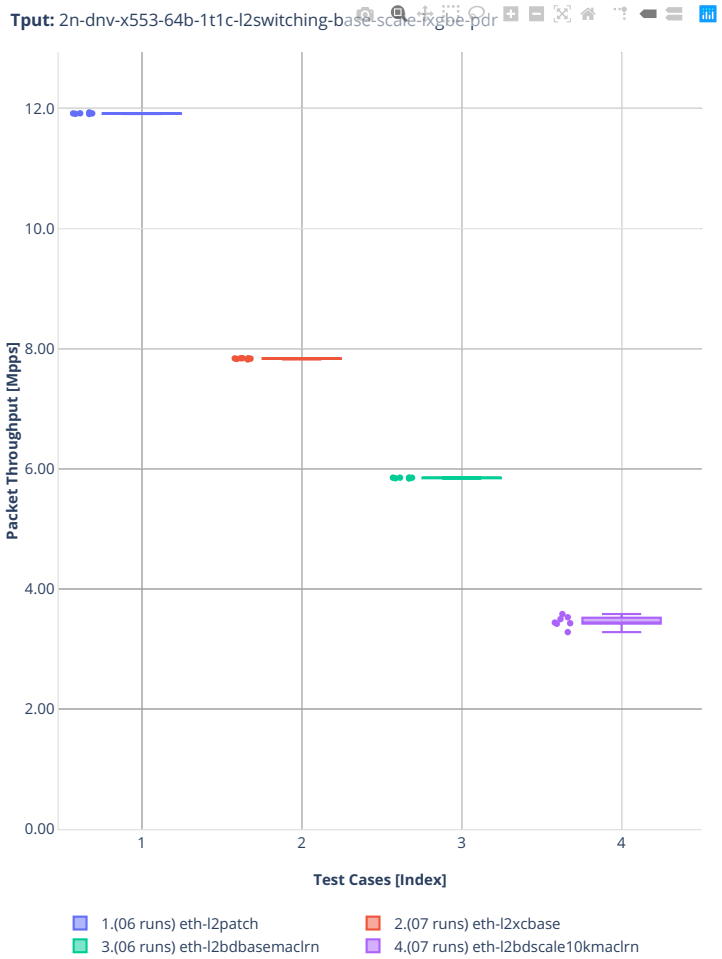



Tput: 2n-tx2-xl710-64b-1t1c-features-l2switching-base-pdr
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) eth-l2bdbasemaclrn-iacl50sf-10kflows
2.(10 runs) eth-l2bdbasemaclrn-iacl50sl-10kflows
3.(10 runs) eth-l2bdbasemaclrn-oacl50sf-10kflows
4.(10 runs) eth-l2bdbasemaclrn-oacl50sl-10kflows
5.(10 runs) eth-l2bdbasemaclrn-macip-iacl50sl-10kflows

2n-dnv-x553

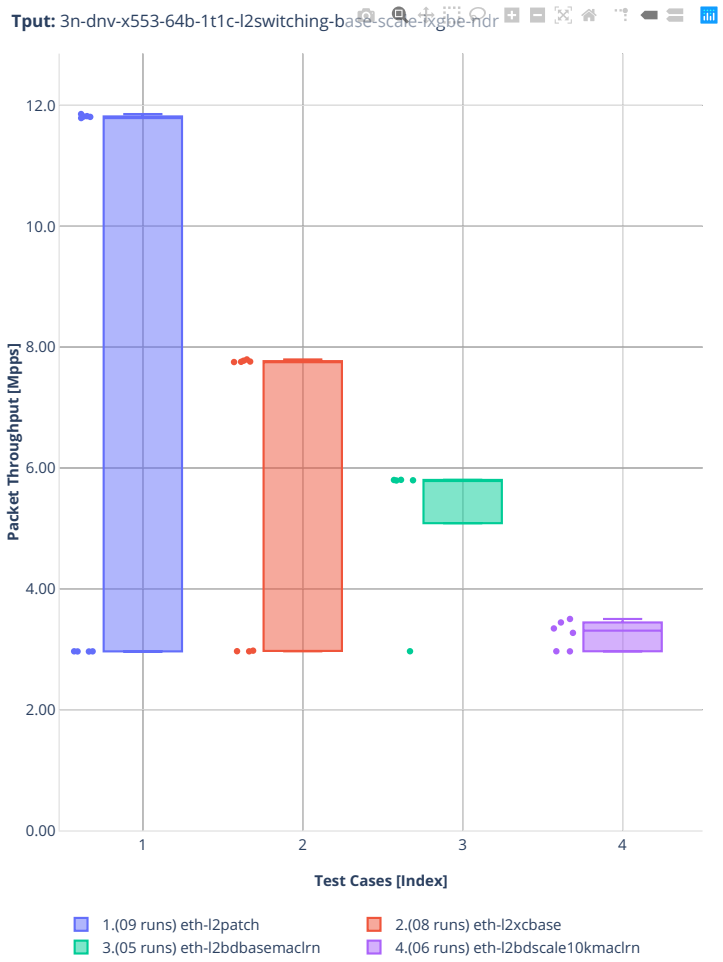
64b-1t1c-l2switching-base-scale-ixgbe

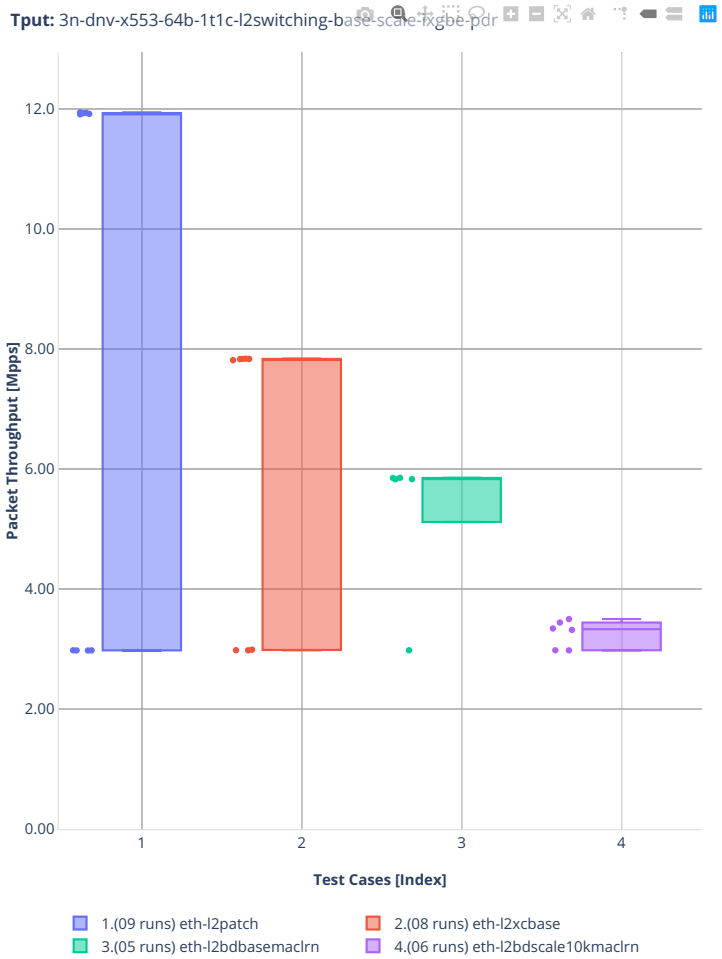



Tput: 2n-dnv-x553-64b-1t1c-l2switching-base-scale-ixgbe-pdr
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(06 runs) eth-l2patch
2.(07 runs) eth-l2xcbase
3.(06 runs) eth-l2bdbasemaclrn
4.(07 runs) eth-l2bdscale10kmaclrn

3n-dnv-x553

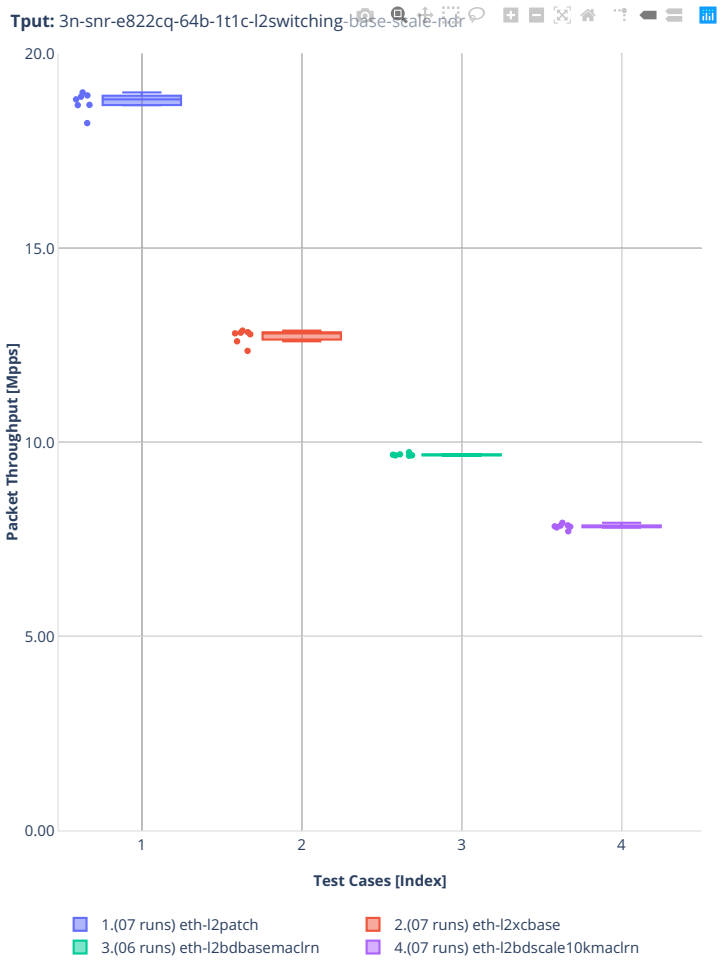
64b-1t1c-l2switching-base-scale-ixgbe

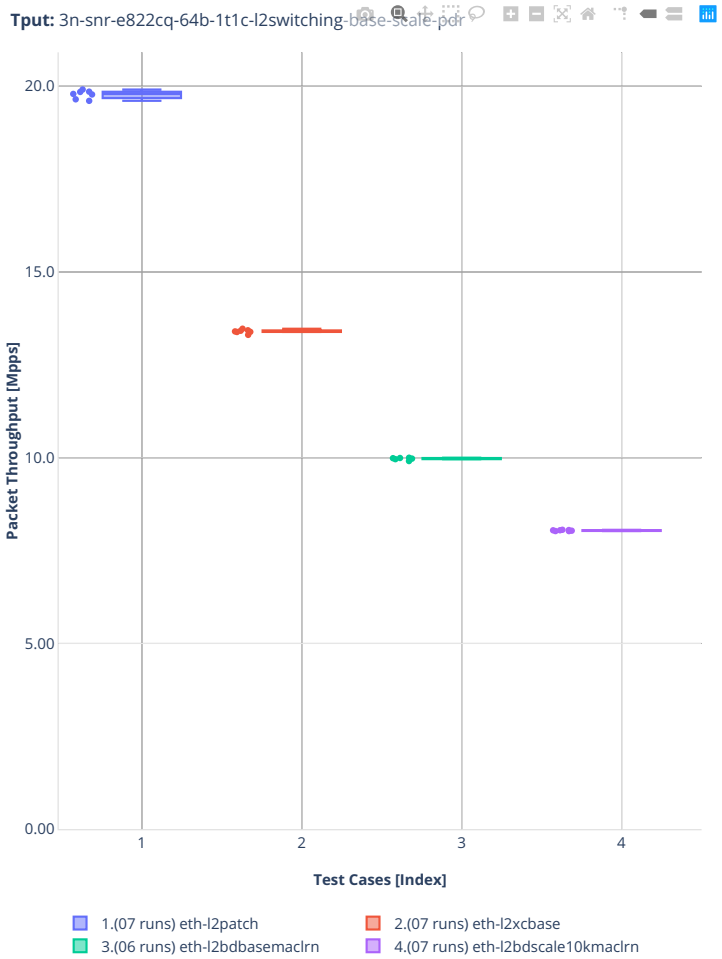



Tput: 3n-dnv-x553-64b-1t1c-l2switching-base-scale-ixgbe-pdr
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(09 runs) eth-l2patch
2.(08 runs) eth-l2xcbase
3.(05 runs) eth-l2bdbasemaclrn
4.(06 runs) eth-l2bdscale10kmaclrn

3n-snr-e822cq

64b-1t1c-l2switching-base-scale




Tput: 3n-snr-e822cq-64b-1t1c-l2switching-base-scale-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(07 runs) eth-l2patch
2.(07 runs) eth-l2xcbase
3.(06 runs) eth-l2bdbasemaclrn
4.(07 runs) eth-l2bdscale10kmaclrn

2.3.2 IPv4 Routing

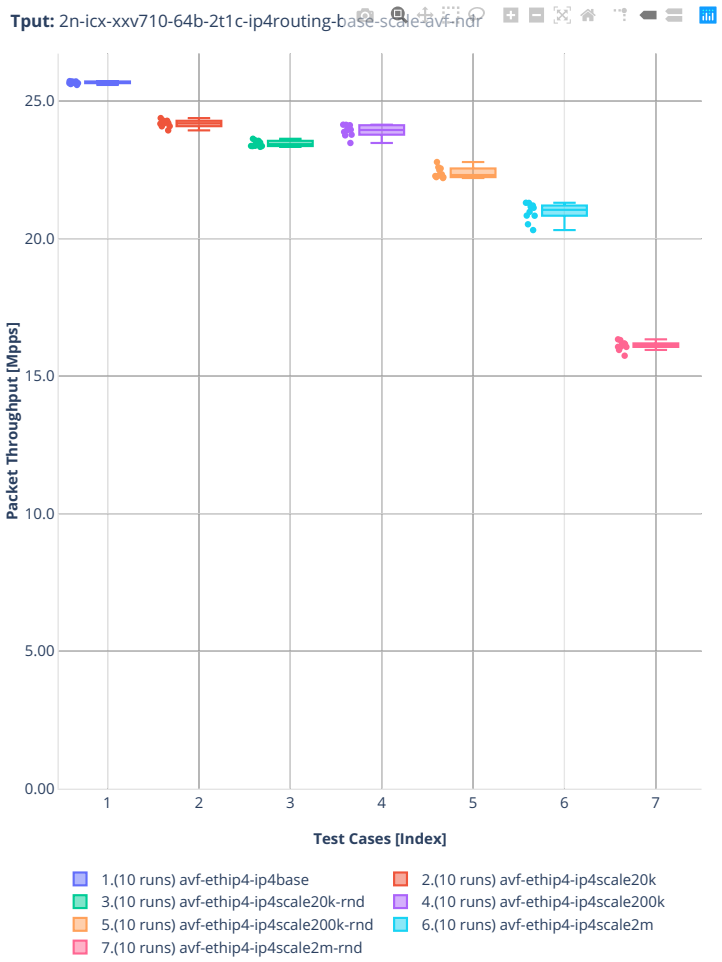
Following sections include summary graphs of VPP Phy-to-Phy performance with IPv4 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

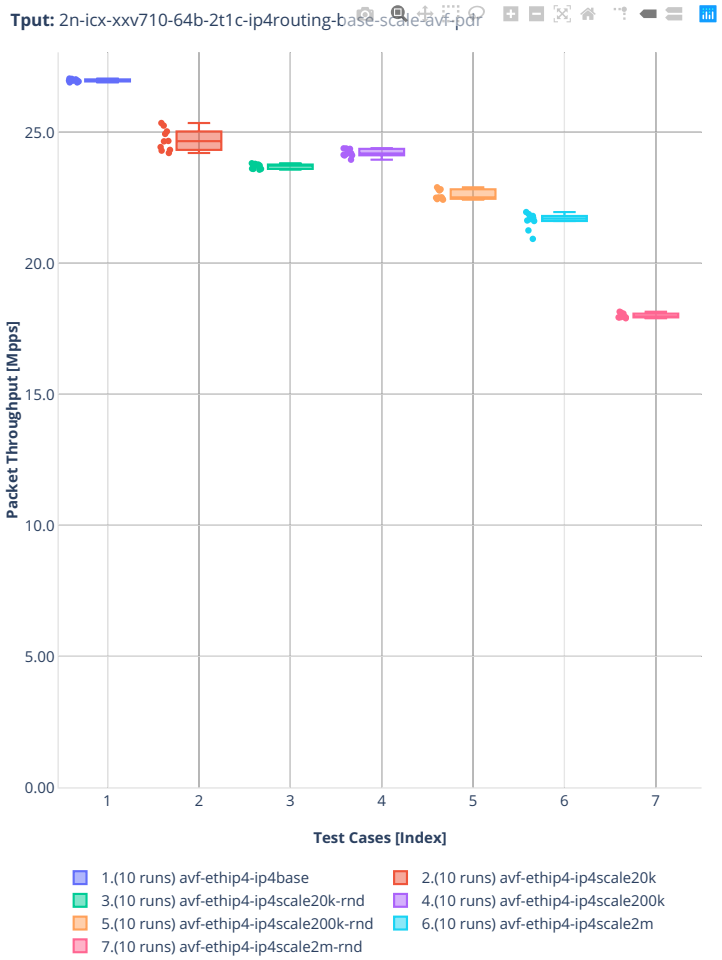
CSIT source code for the test cases used for plots can be found in CSIT git repository[115].

[115] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls2210

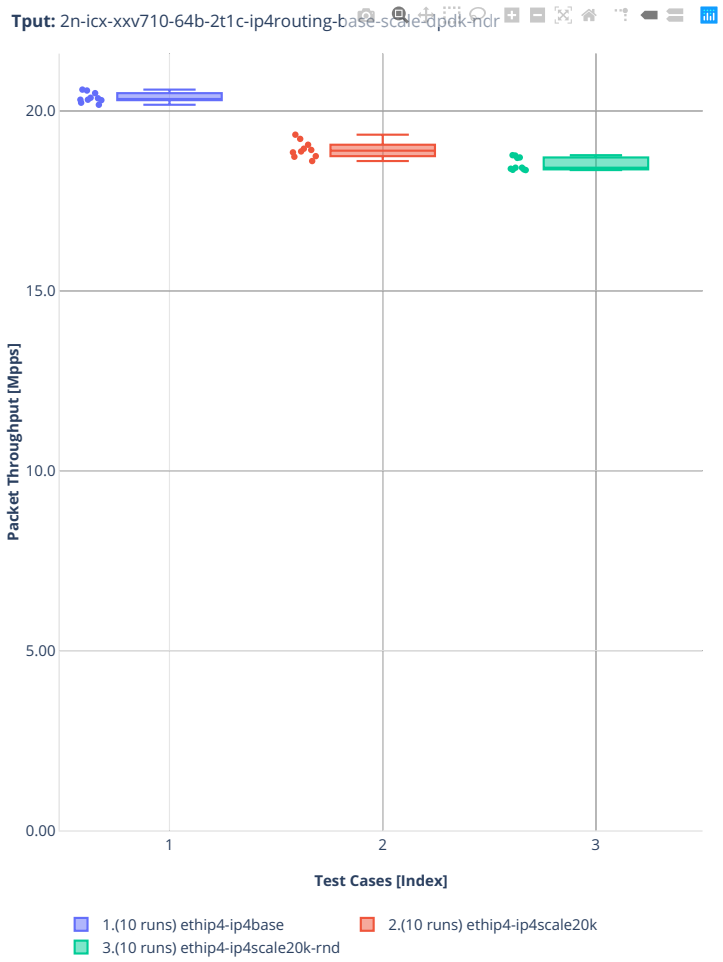
2n-icx-xxv710

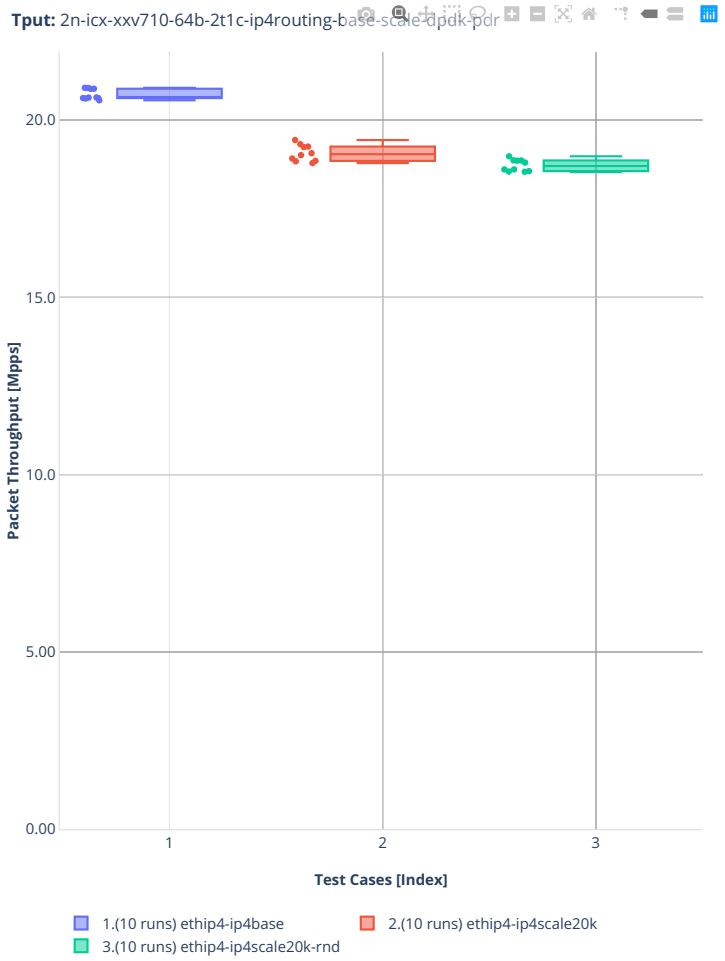
64b-2t1c-ip4routing-base-scale-avf



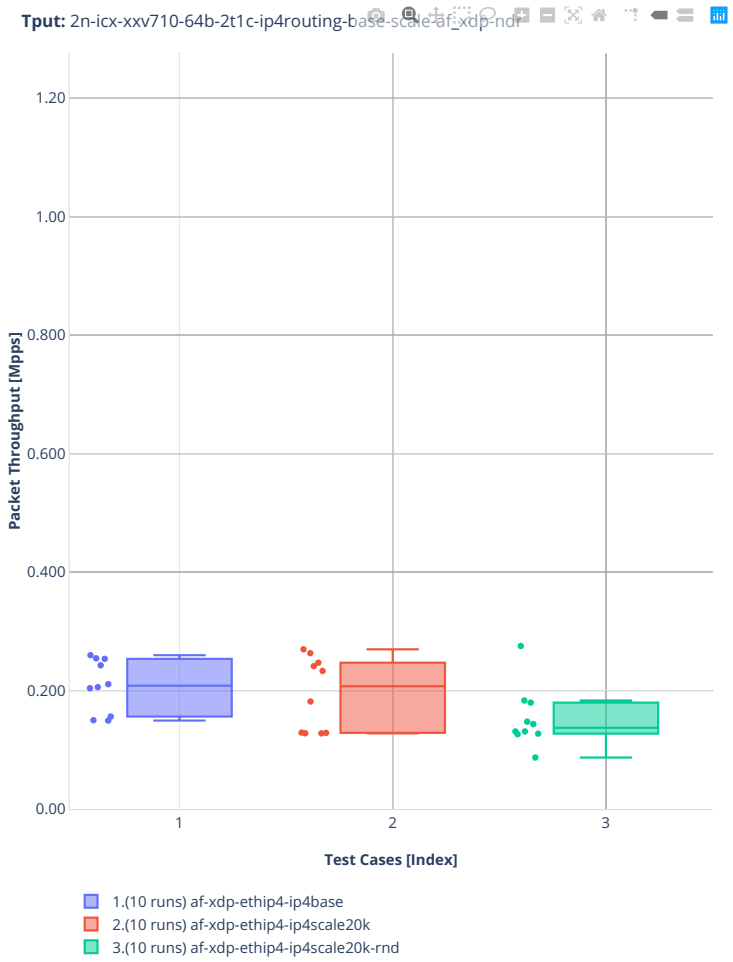

Tput: 2n-icx-xxv710-64b-2t1c-ip4routing-base-scale-avf-pdr
Packet Throughput [Mpps]
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
7
Test Cases [Index]
1.(10 runs) avf-ethip4-ip4base
2.(10 runs) avf-ethip4-ip4scale20k
3.(10 runs) avf-ethip4-ip4scale20k-rnd
4.(10 runs) avf-ethip4-ip4scale200k
5.(10 runs) avf-ethip4-ip4scale200k-rnd
6.(10 runs) avf-ethip4-ip4scale2m
7.(10 runs) avf-ethip4-ip4scale2m-rnd

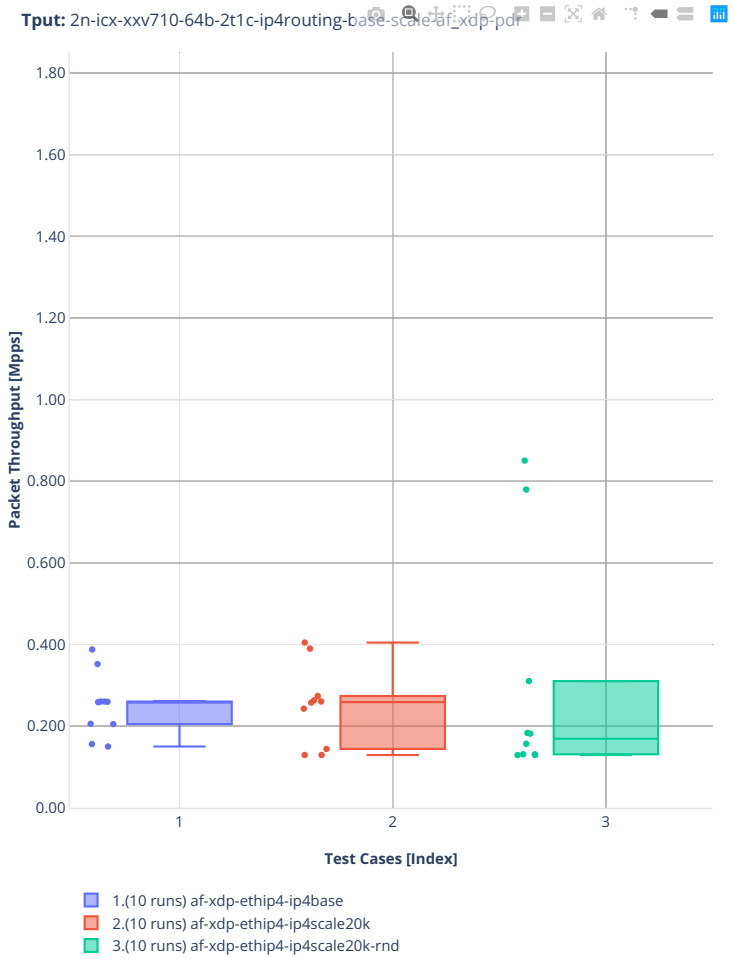
64b-2t1c-ip4routing-base-scale-dpdk



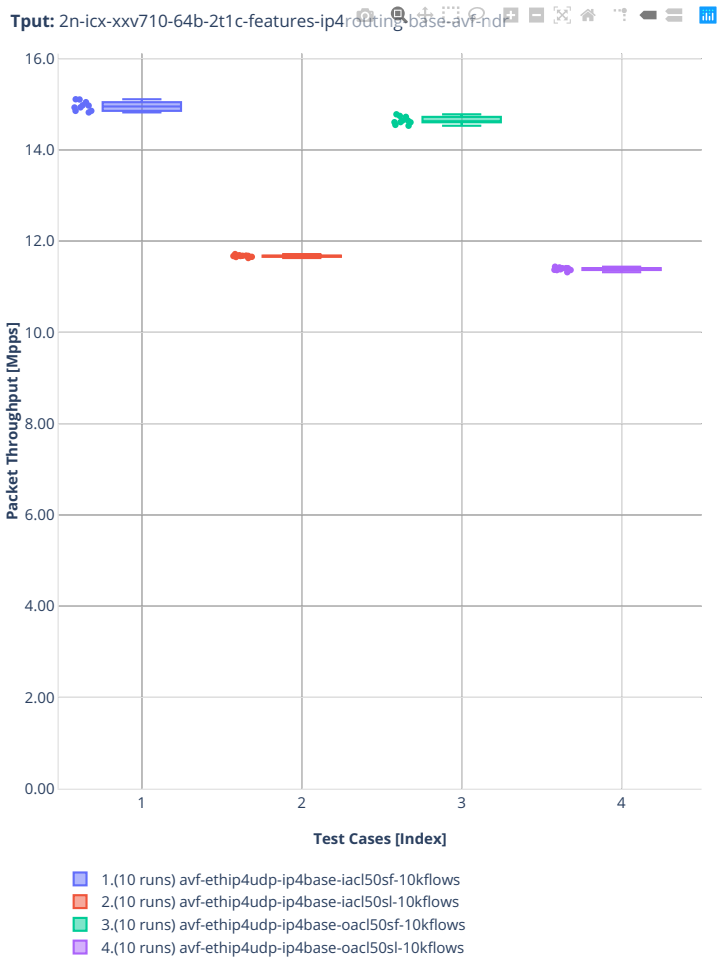

Tput: 2n-icx-xxv710-64b-2t1c-ip4routing-base-scale-dpdk-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4-ip4base
2.(10 runs) ethip4-ip4scale20k
3.(10 runs) ethip4-ip4scale20k-rnd

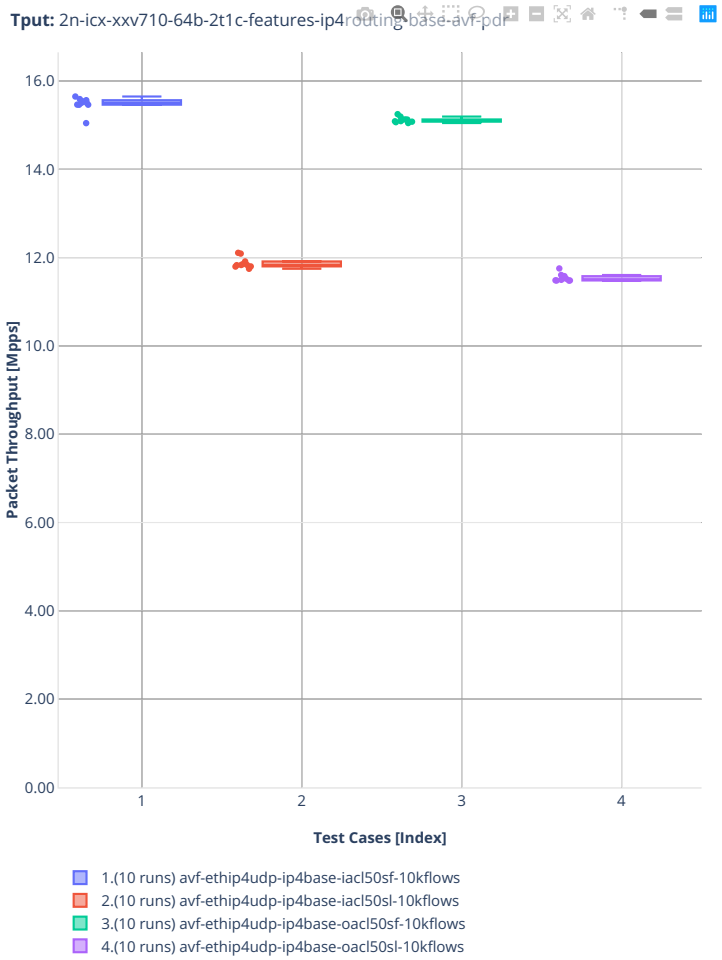
64b-2t1c-ip4routing-base-scale-af_xdp




Tput: 2n-icx-xxv710-64b-2t1c-ip4routing-base-scale-af_xdp-pdr
Packet Throughput [Mpps]
1.80
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
Test Cases [Index]
1.(10 runs) af-xdp-ethip4-ip4base
2.(10 runs) af-xdp-ethip4-ip4scale20k
3.(10 runs) af-xdp-ethip4-ip4scale20k-rnd

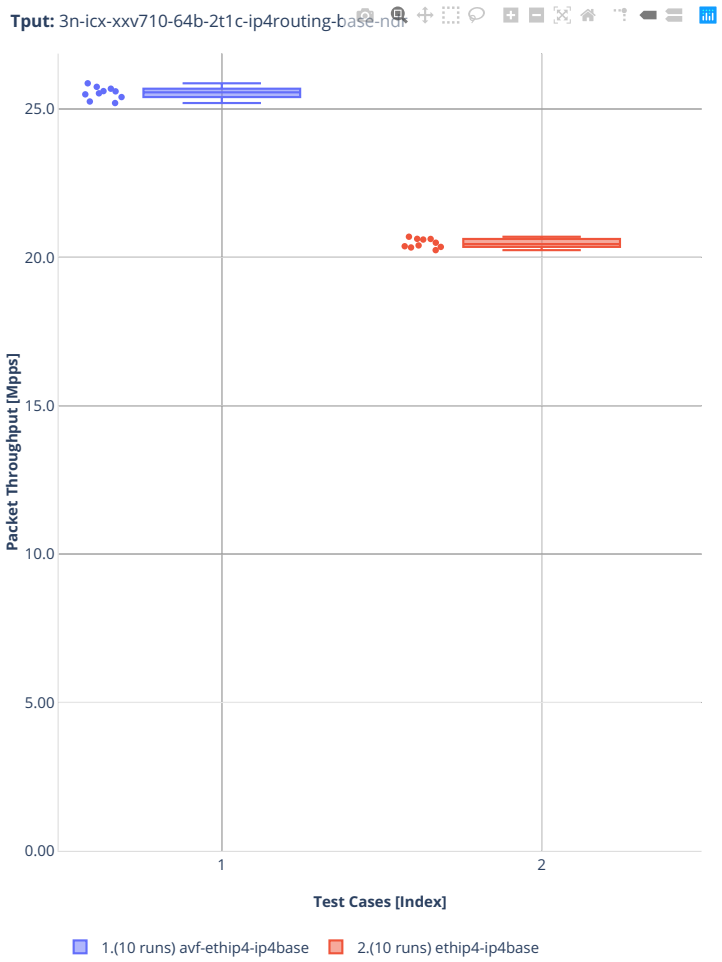
64b-2t1c-features-ip4routing-base-avf

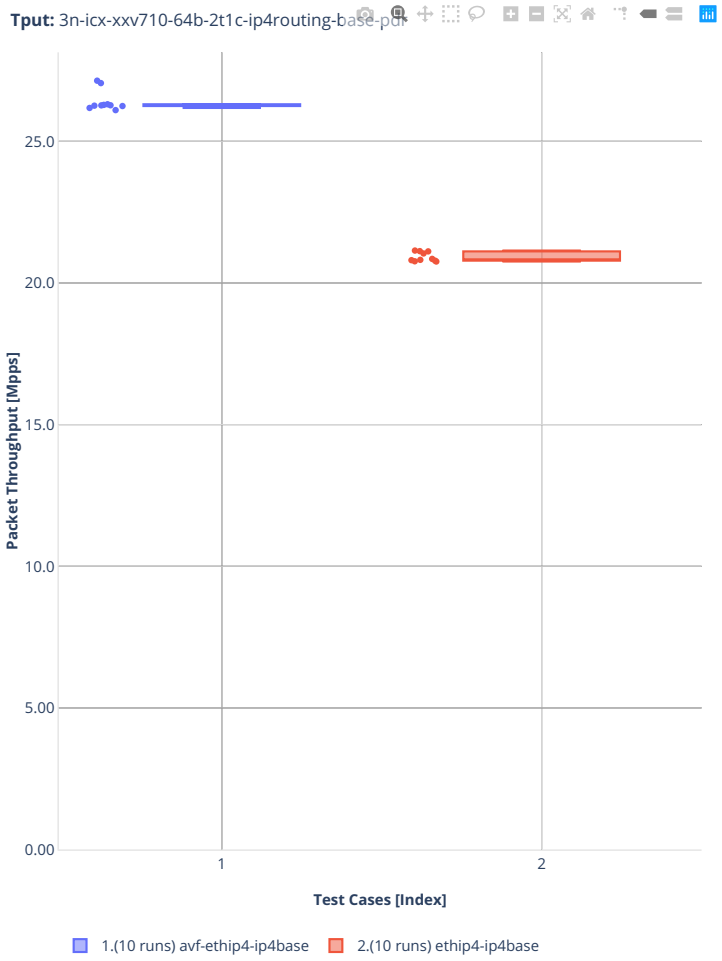



Tput: 2n-icx-xxv710-64b-2t1c-features-ip4routing-base-avf-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-iacl50sf-10kflows
2.(10 runs) avf-ethip4udp-ip4base-iacl50sl-10kflows
3.(10 runs) avf-ethip4udp-ip4base-oacl50sf-10kflows
4.(10 runs) avf-ethip4udp-ip4base-oacl50sl-10kflows

3n-icx-xxv710

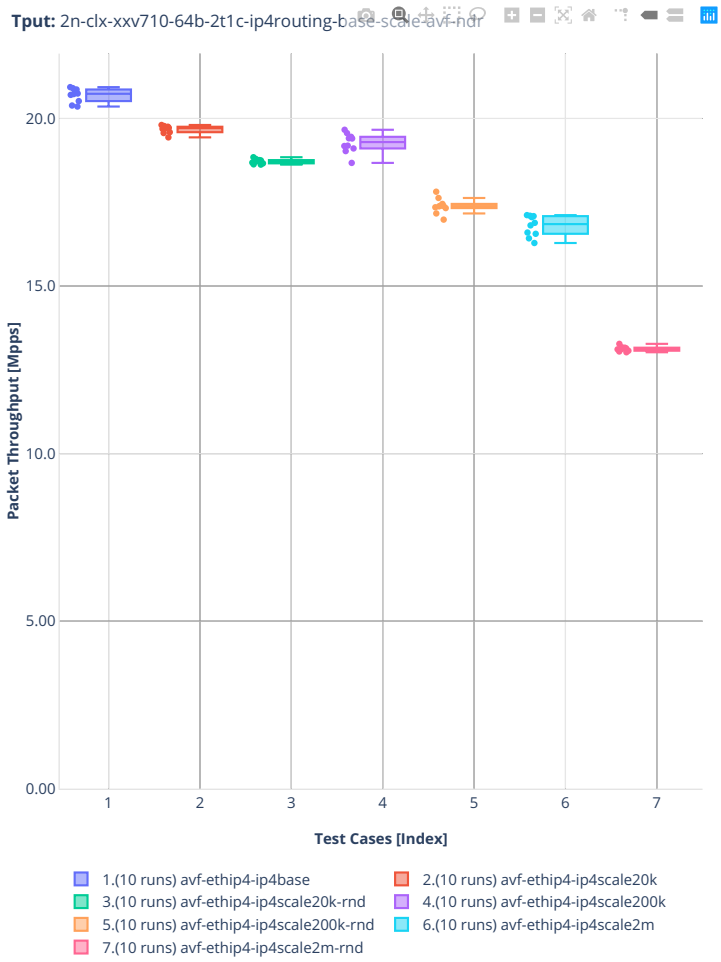
64b-2t1c-ip4routing-base

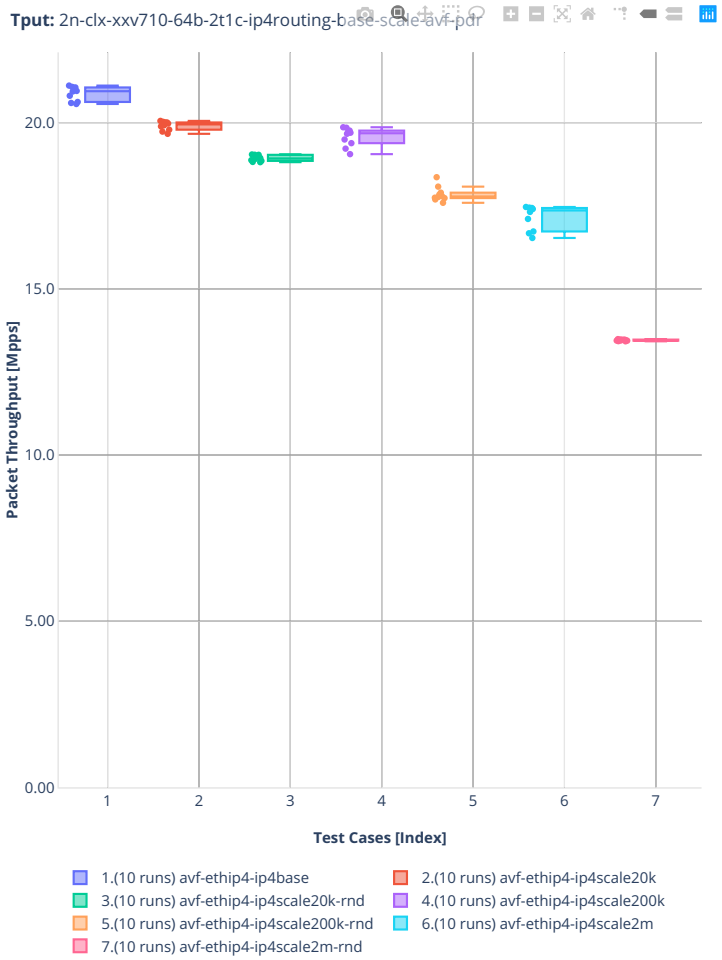



Tput: 3n-icx-xxv710-64b-2t1c-ip4routing-base-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) avf-ethip4-ip4base
2.(10 runs) ethip4-ip4base

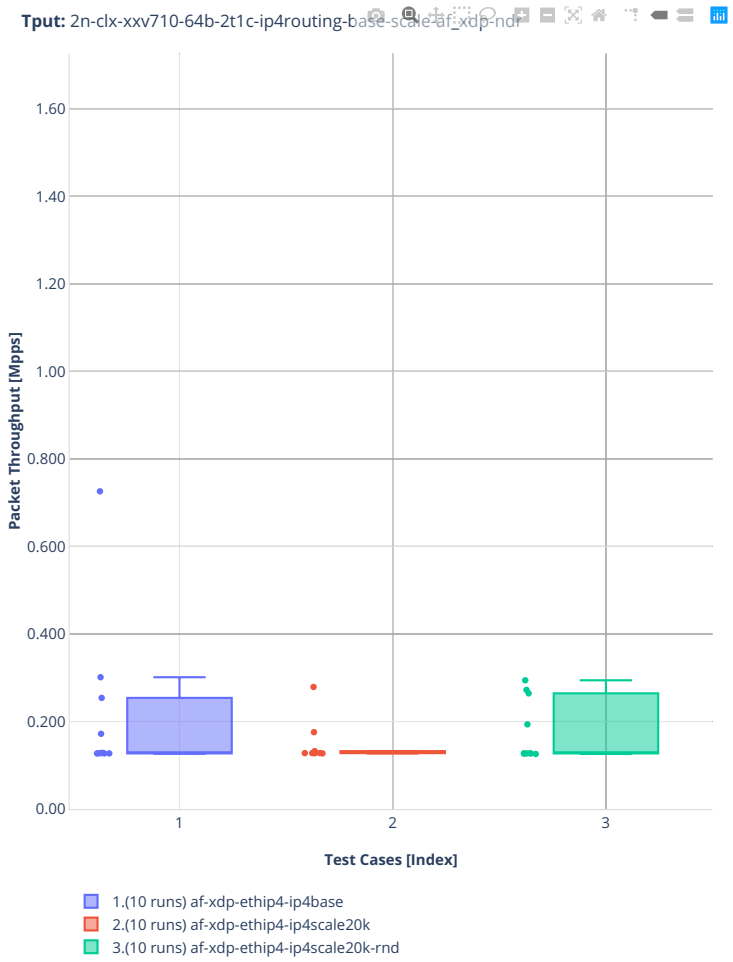
2n-clx-xxv710

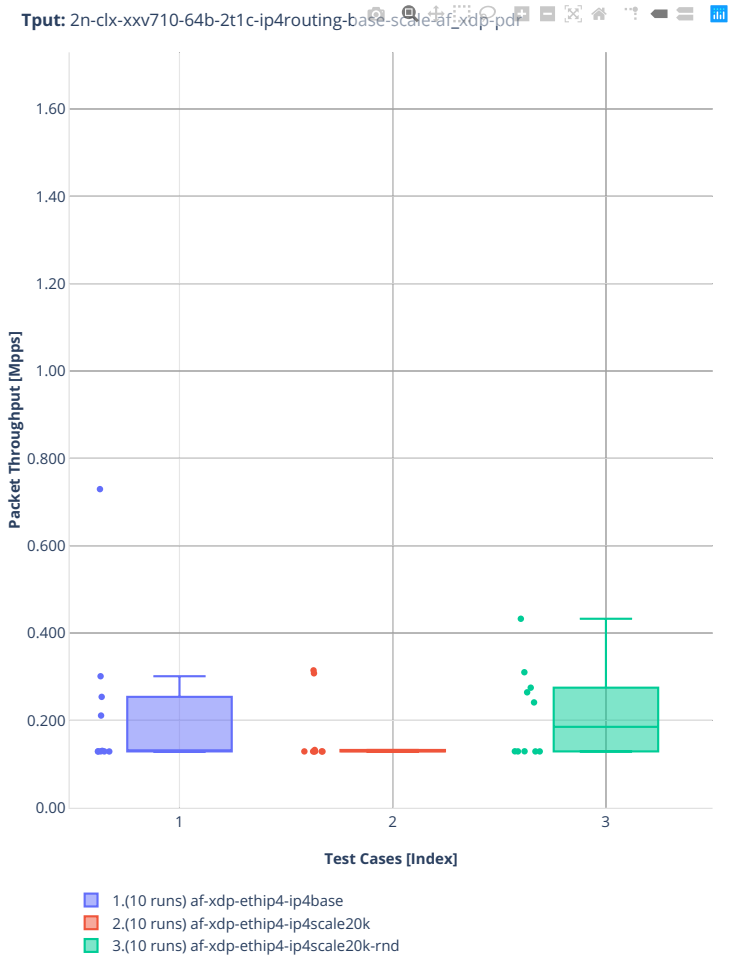
64b-2t1c-ip4routing-base-scale-avf



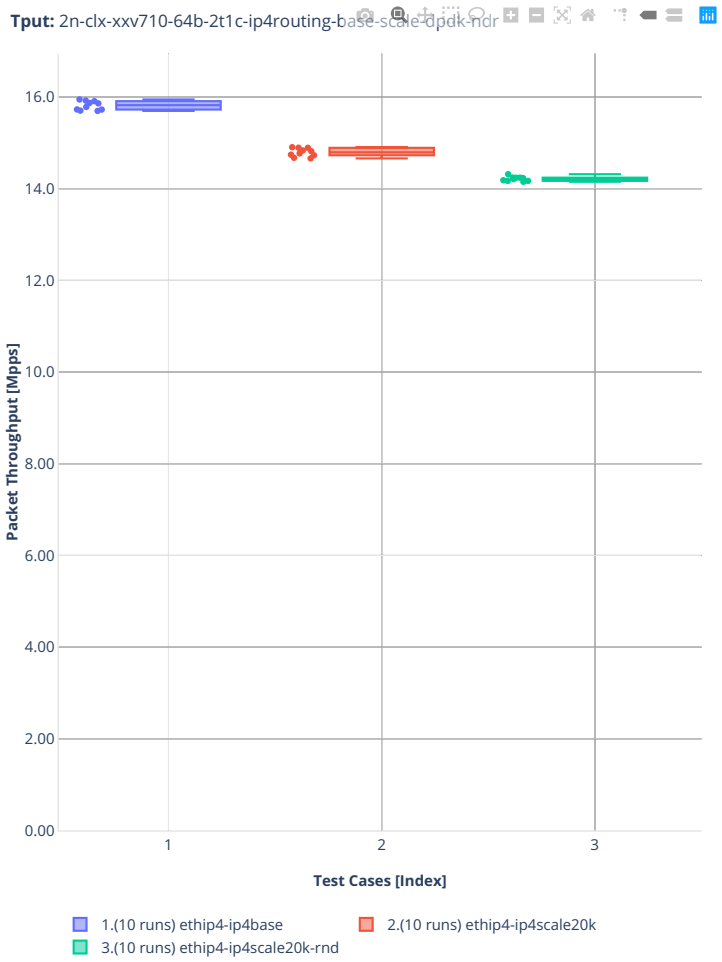

Tput: 2n-clx-xxv710-64b-2t1c-ip4routing-base-scale-avf-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
7
Test Cases [Index]
1.(10 runs) avf-ethip4-ip4base
2.(10 runs) avf-ethip4-ip4scale20k
3.(10 runs) avf-ethip4-ip4scale20k-rnd
4.(10 runs) avf-ethip4-ip4scale200k
5.(10 runs) avf-ethip4-ip4scale200k-rnd
6.(10 runs) avf-ethip4-ip4scale2m
7.(10 runs) avf-ethip4-ip4scale2m-rnd

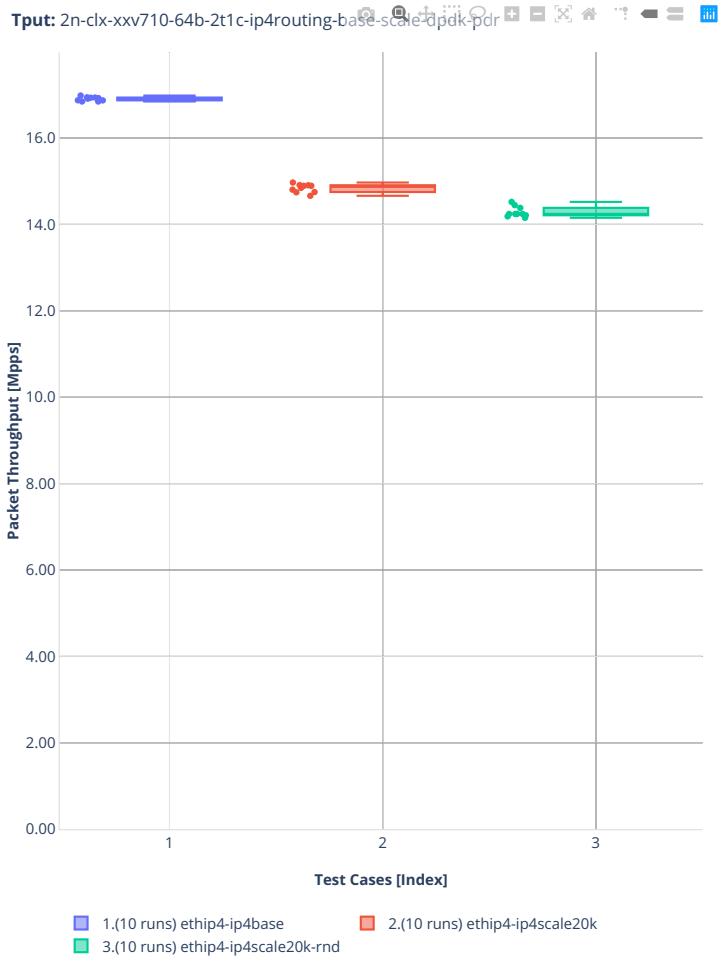
64b-2t1c-ip4routing-base-scale-af-xdp



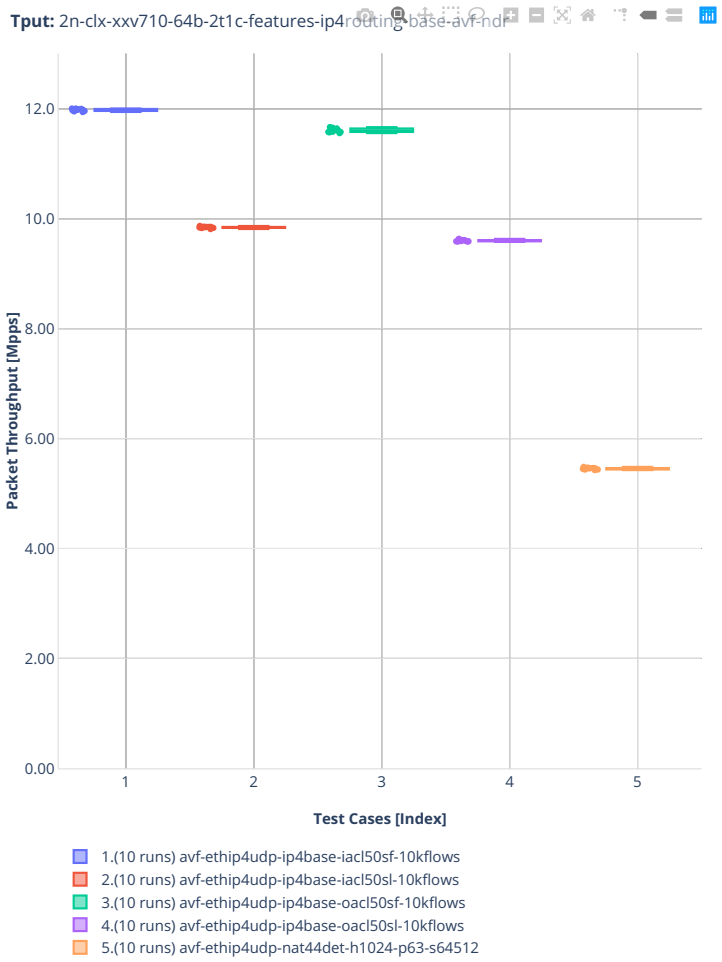

Tput: 2n-clx-xxv710-64b-2t1c-ip4routing-base-scale-af_xdp-pdr
Packet Throughput [Mpps]
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
Test Cases [Index]
1.(10 runs) af-xdp-ethip4-ip4base
2.(10 runs) af-xdp-ethip4-ip4scale20k
3.(10 runs) af-xdp-ethip4-ip4scale20k-rnd

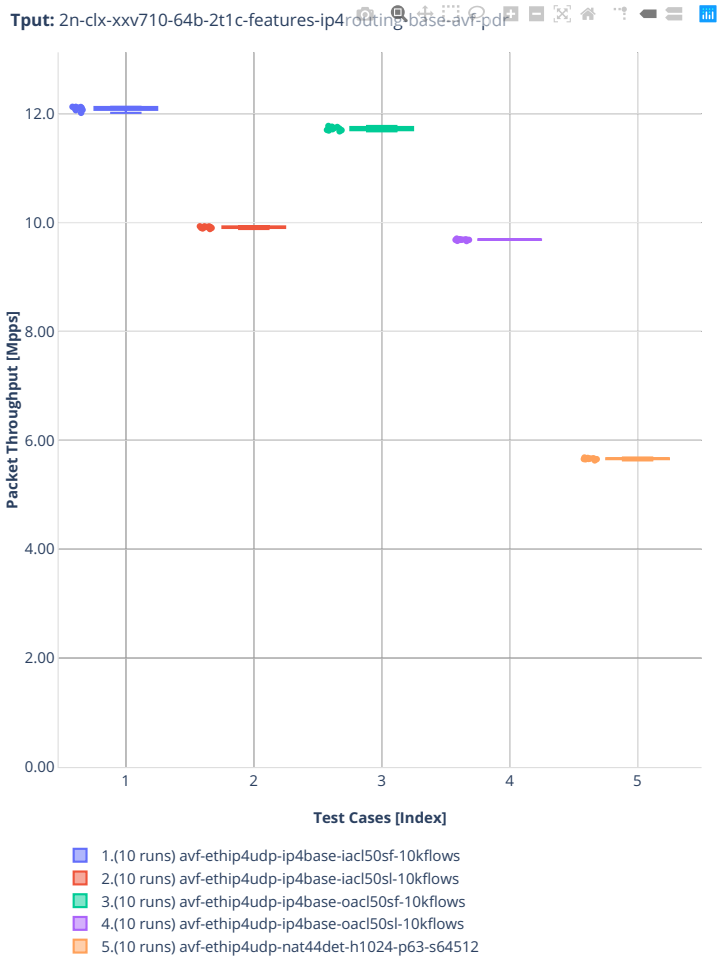
64b-2t1c-ip4routing-base-scale-dpdk




Tput: 2n-clx-xxv710-64b-2t1c-ip4routing-base-scale-dpdk-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4-ip4base
2.(10 runs) ethip4-ip4scale20k
3.(10 runs) ethip4-ip4scale20k-rnd

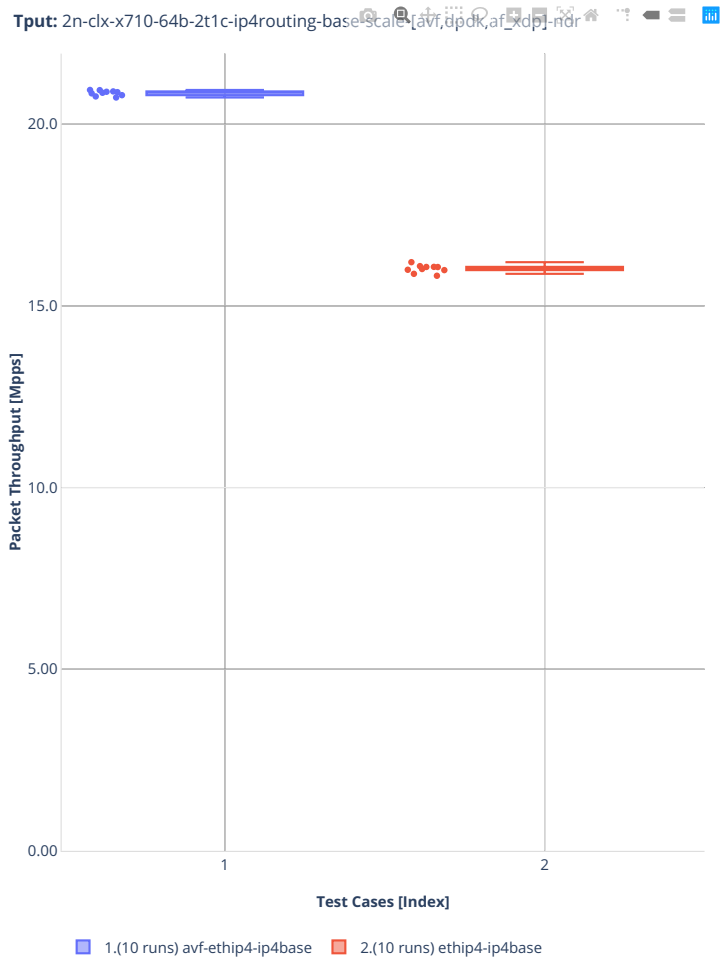
64b-2t1c-features-ip4routing-base-avf

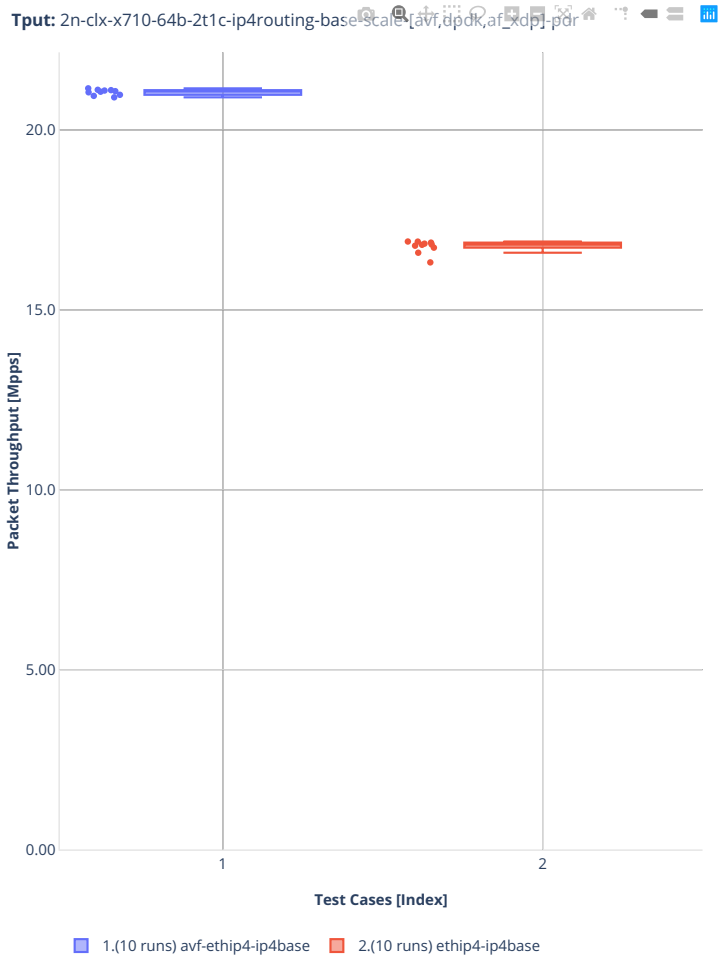



Tput: 2n-clx-xxv710-64b-2t1c-features-ip4routing-base-avf-pdr
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-iacl50sf-10kflows
2.(10 runs) avf-ethip4udp-ip4base-iacl50sl-10kflows
3.(10 runs) avf-ethip4udp-ip4base-oacl50sf-10kflows
4.(10 runs) avf-ethip4udp-ip4base-oacl50sl-10kflows
5.(10 runs) avf-ethip4udp-nat44det-h1024-p63-s64512

2n-clx-x710

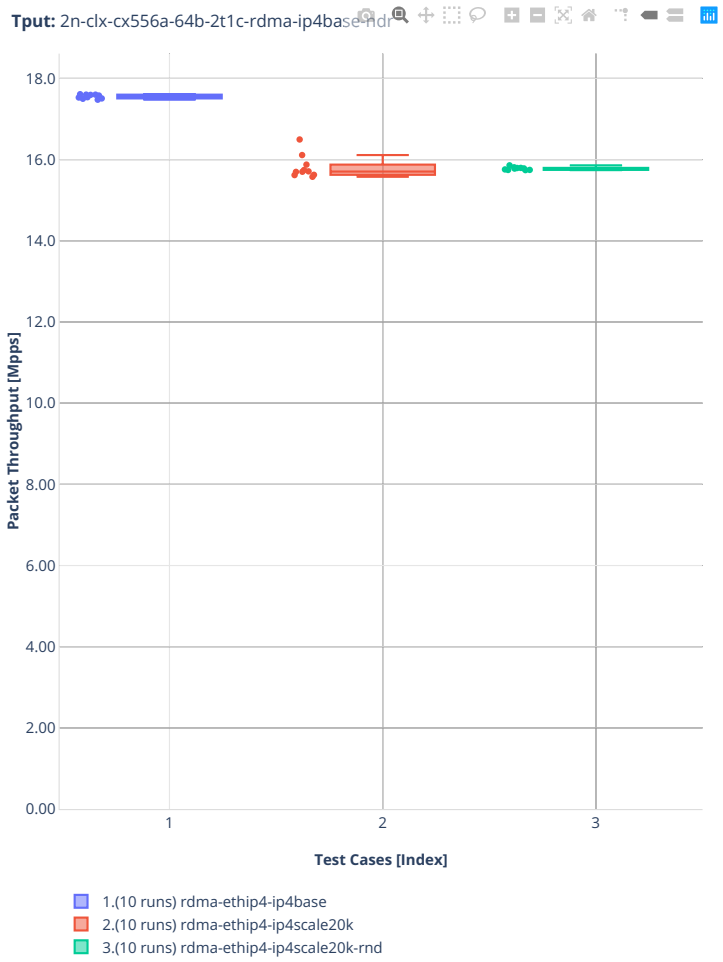
64b-2t1c-ip4routing-base-scale-[avf,dpdk]

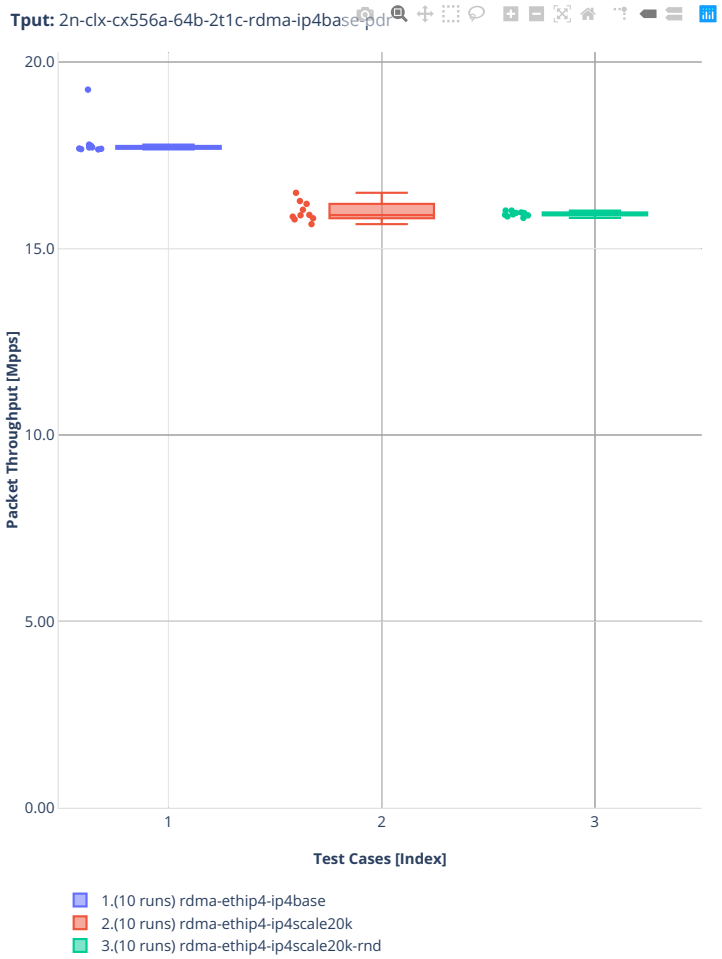



Tput: 2n-clx-x710-64b-2t1c-ip4routing-base-scale-[avf,dpdk,af_xdp]-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) avf-ethip4-ip4base
2.(10 runs) ethip4-ip4base

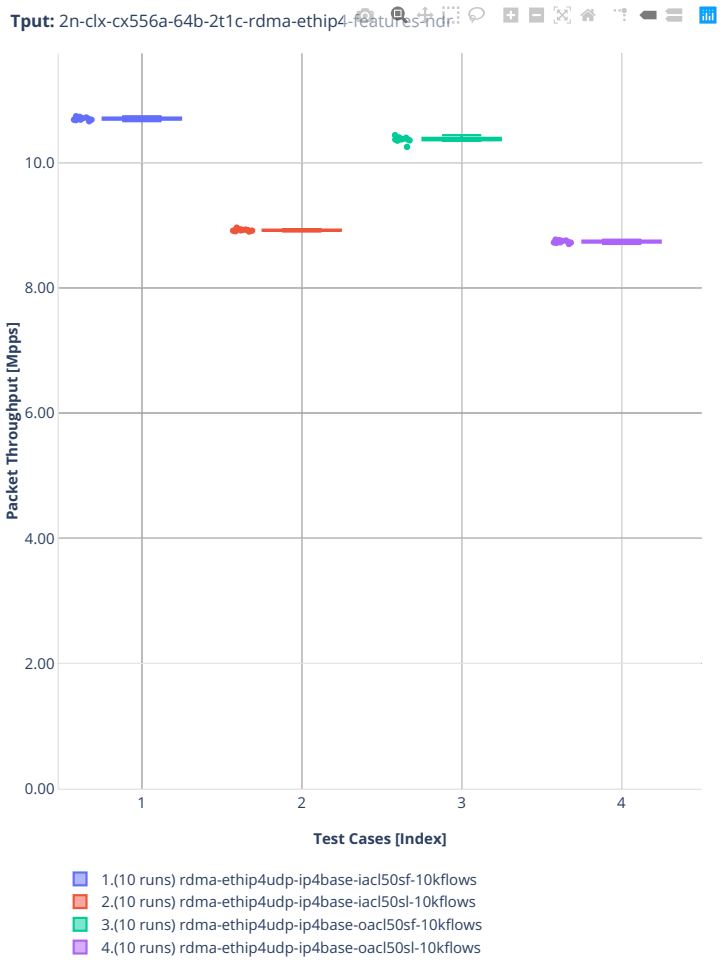
2n-clx-cx556a

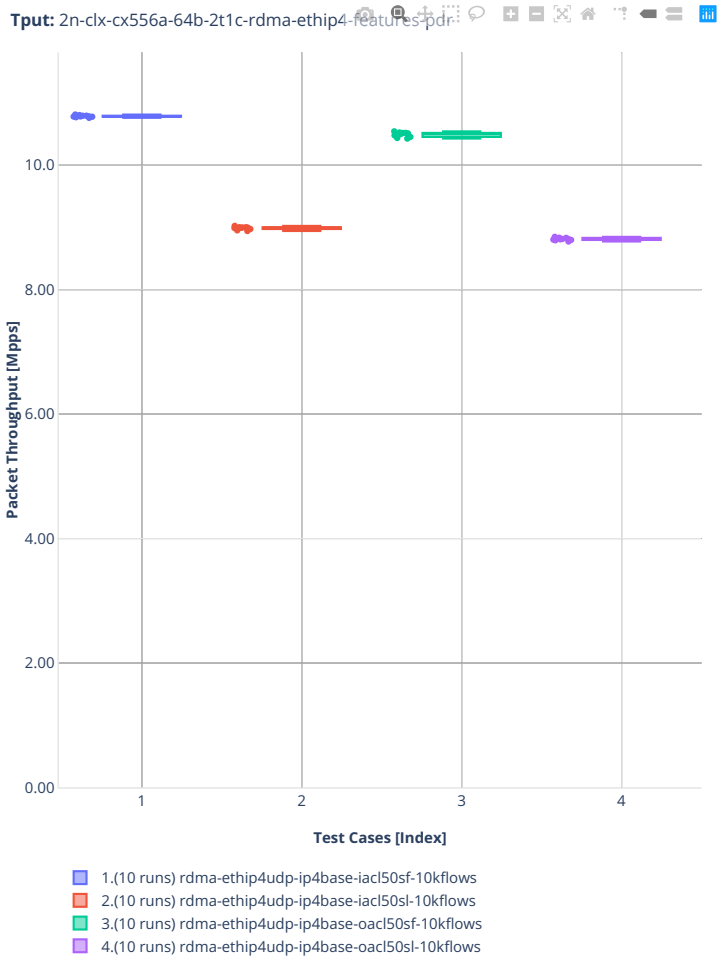
64b-2t1c-ip4routing-base-scale-rdma-core




Tput: 2n-clx-cx556a-64b-2t1c-rdma-ip4base-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) rdma-ethip4-ip4base
2.(10 runs) rdma-ethip4-ip4scale20k
3.(10 runs) rdma-ethip4-ip4scale20k-rnd

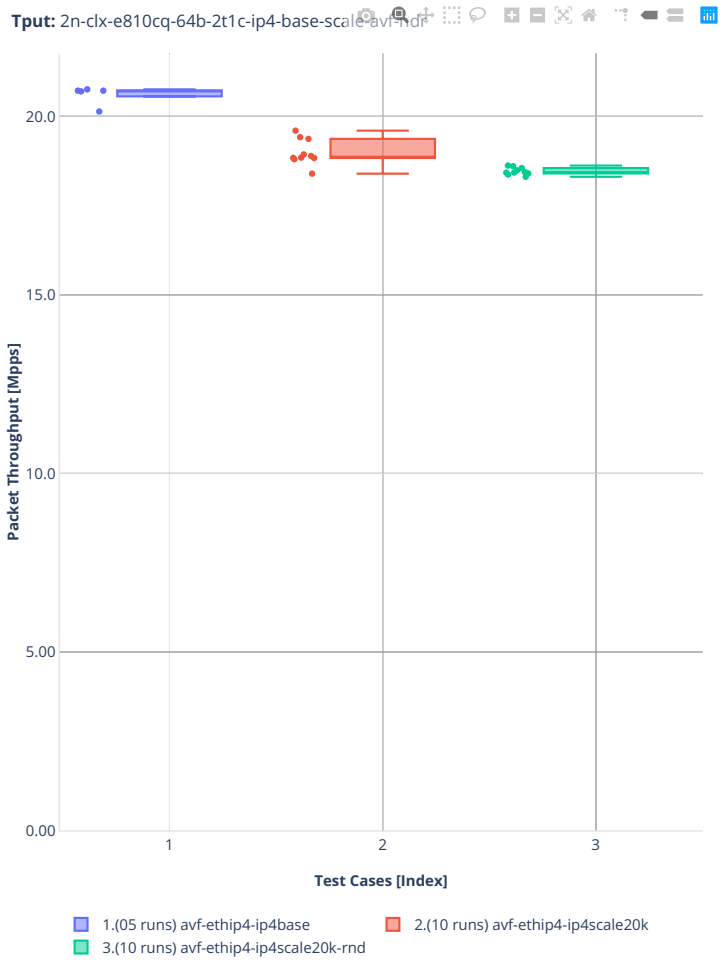
64b-2t1c-ip4routing-features

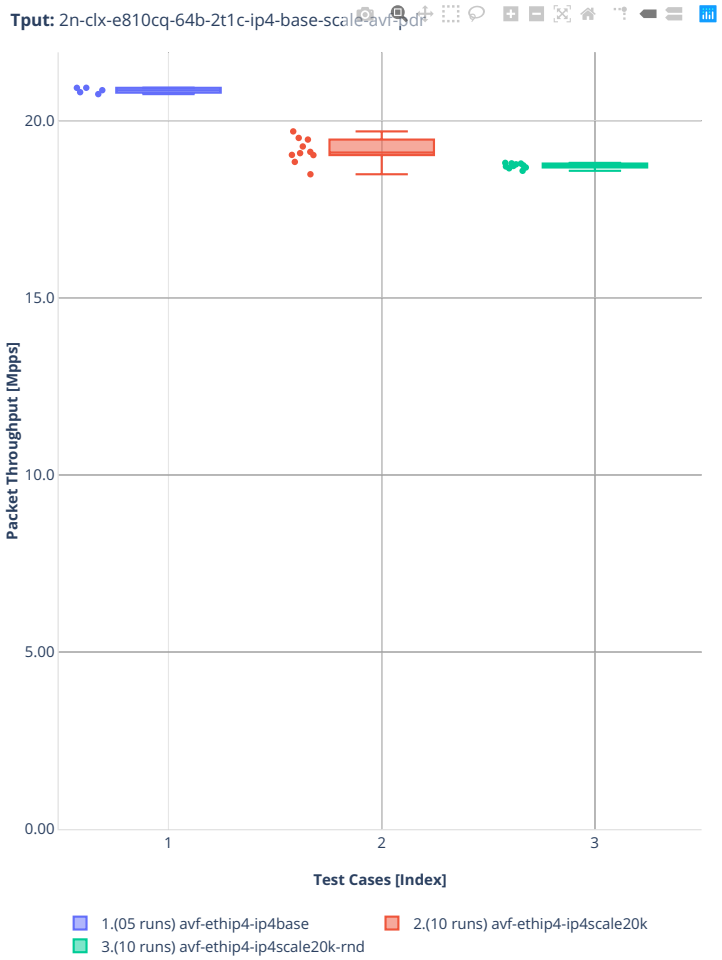



Tput: 2n-clx-cx556a-64b-2t1c-rdma-ethip4-features-pdr
Packet Throughput [Mpps]
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) rdma-ethip4udp-ip4base-iacl50sf-10kflows
2.(10 runs) rdma-ethip4udp-ip4base-iacl50sl-10kflows
3.(10 runs) rdma-ethip4udp-ip4base-oacl50sf-10kflows
4.(10 runs) rdma-ethip4udp-ip4base-oacl50sl-10kflows

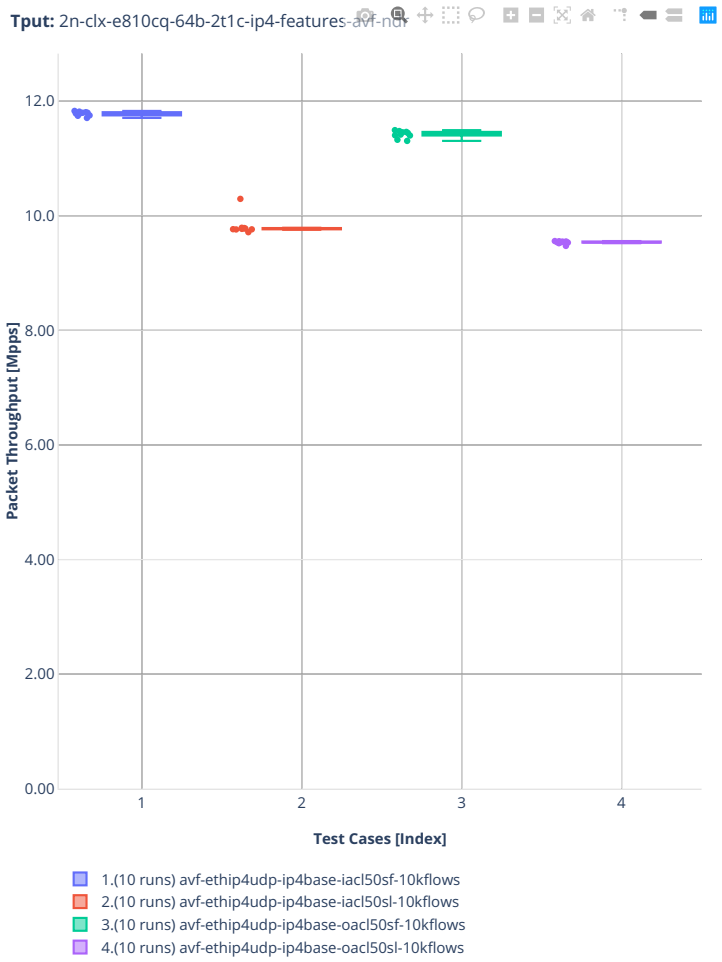
2n-clx-e810cq

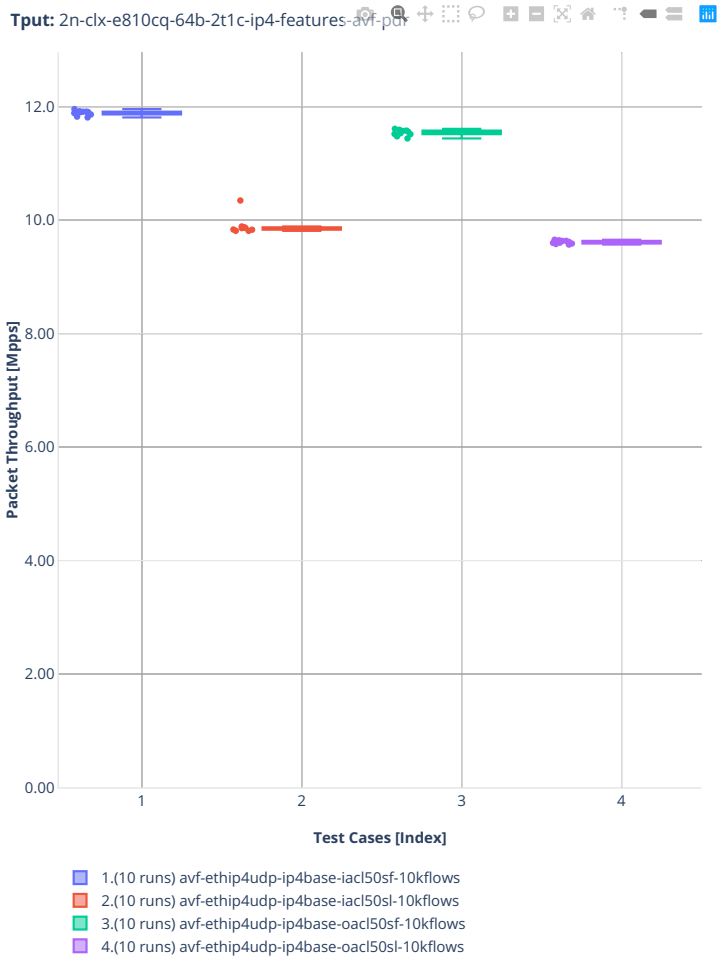
64b-2t1c-ip4routing-base-scale-avf



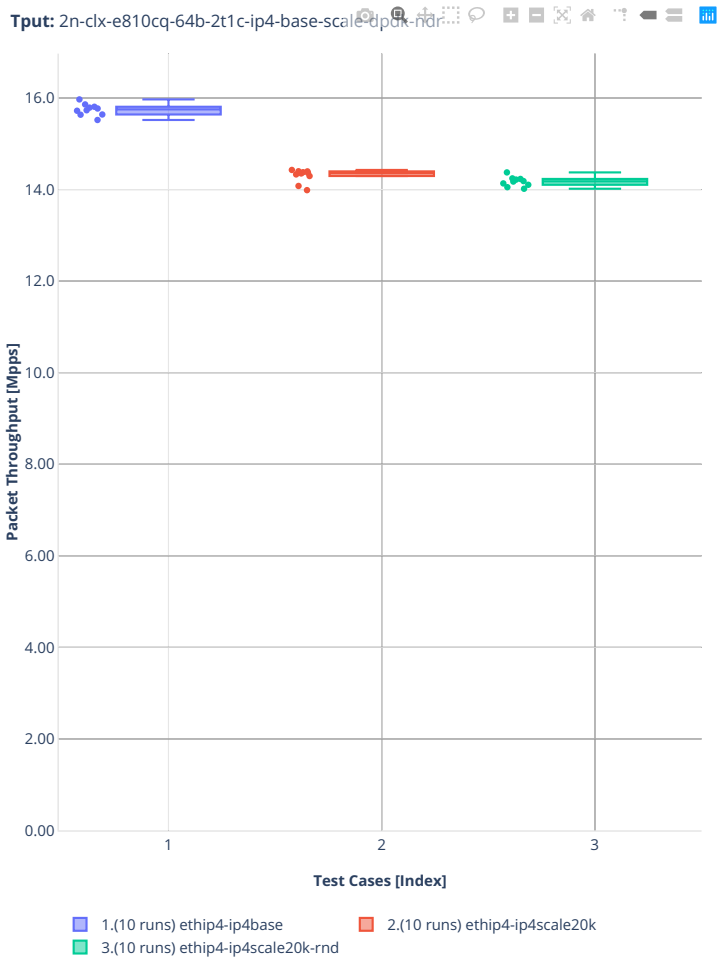

Tput: 2n-clx-e810cq-64b-2t1c-ip4-base-scale-avf-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(05 runs) avf-ethip4-ip4base
2.(10 runs) avf-ethip4-ip4scale20k
3.(10 runs) avf-ethip4-ip4scale20k-rnd

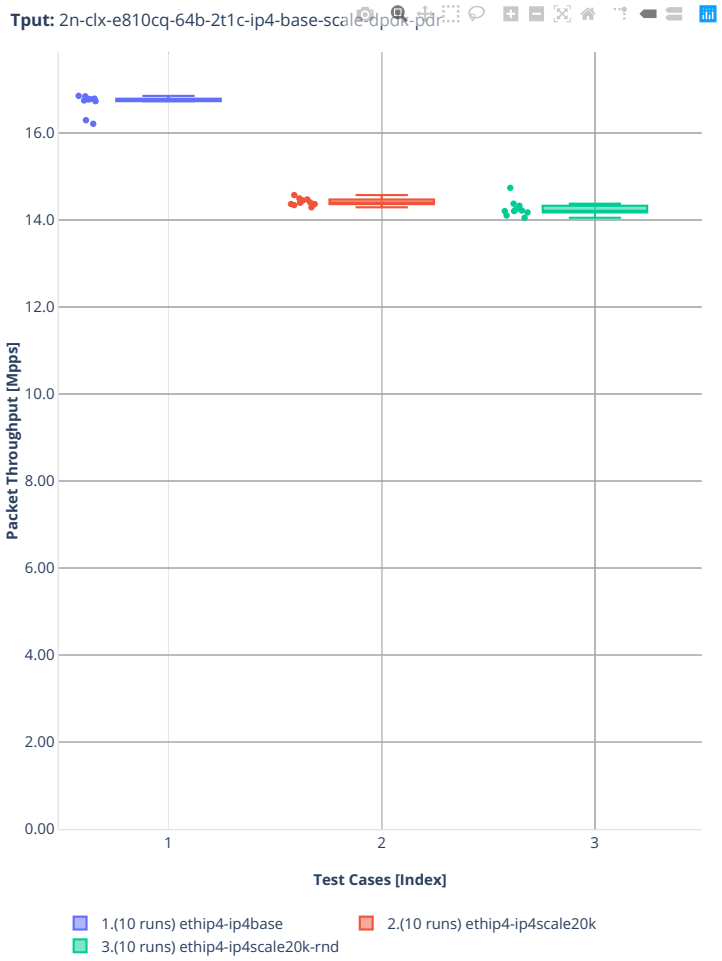
64b-2t1c-ip4routing-features-avf




Tput: 2n-clx-e810cq-64b-2t1c-ip4-features-avf-pdr
Packet Throughput [Mpps]
12.0
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-iacl50sf-10kflows
2.(10 runs) avf-ethip4udp-ip4base-iacl50sl-10kflows
3.(10 runs) avf-ethip4udp-ip4base-oacl50sf-10kflows
4.(10 runs) avf-ethip4udp-ip4base-oacl50sl-10kflows

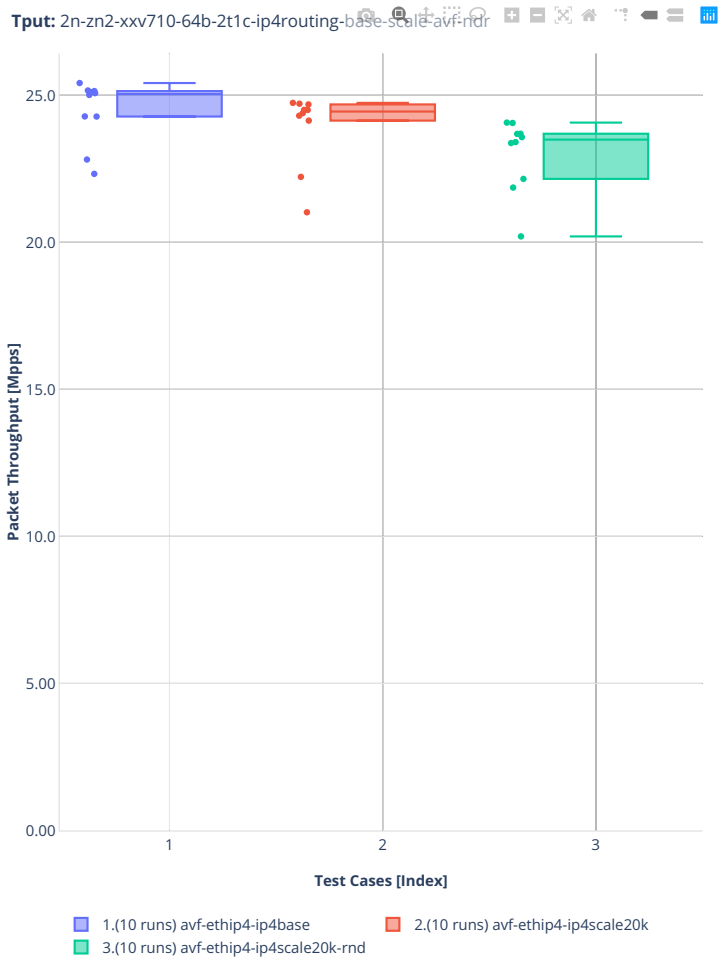
64b-2t1c-ip4routing-base-scale-dpdk

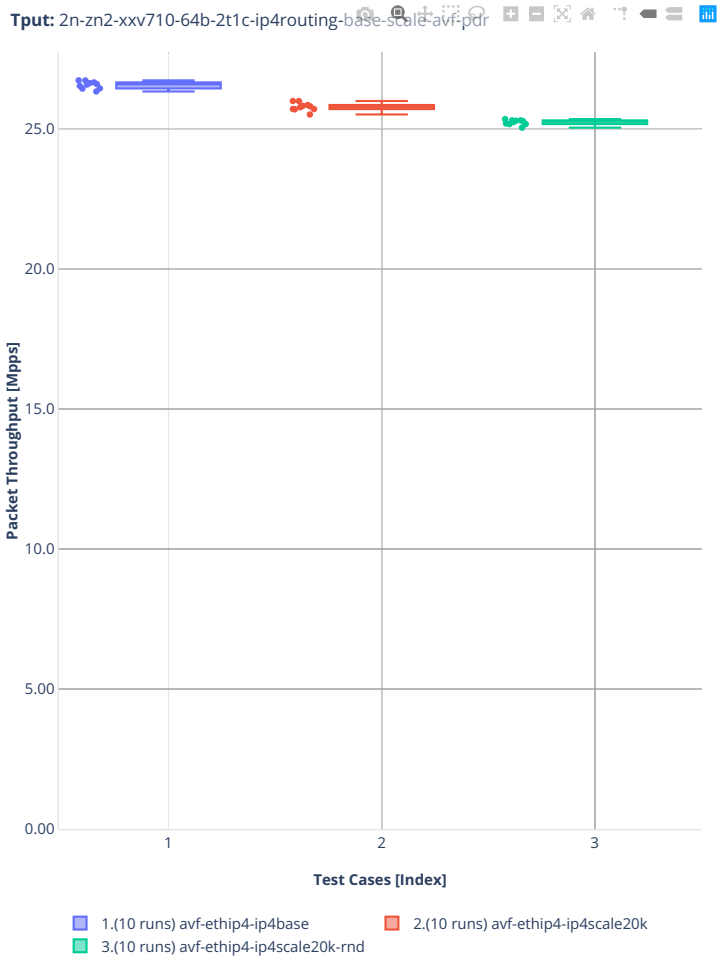



Tput: 2n-clx-e810cq-64b-2t1c-ip4-base-scale-dpdk-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4-ip4base
2.(10 runs) ethip4-ip4scale20k
3.(10 runs) ethip4-ip4scale20k-rnd

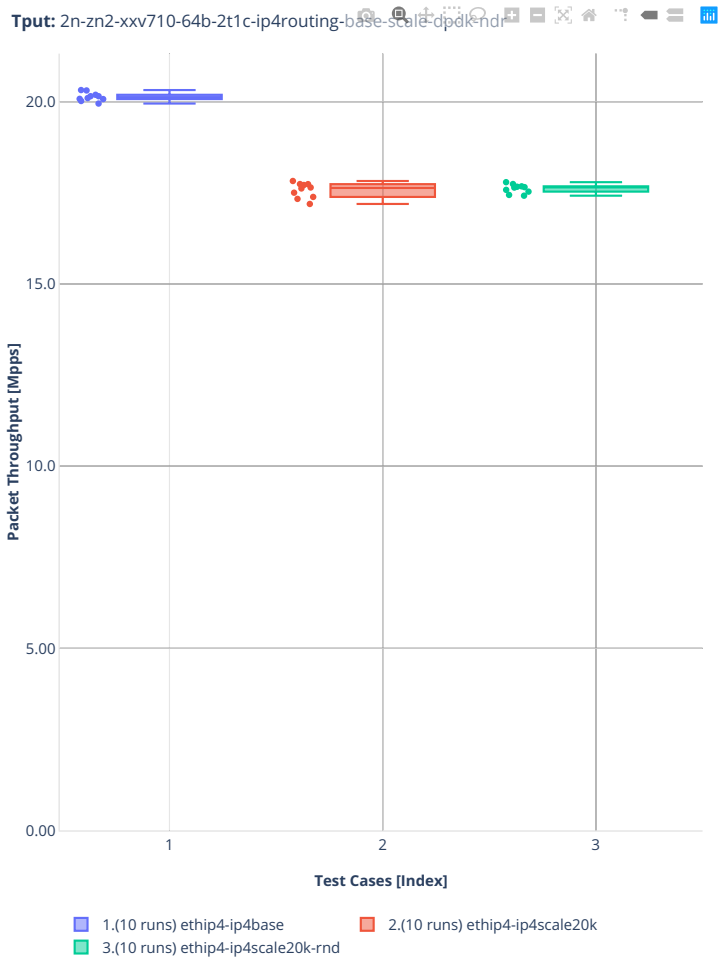
2n-zn2-xxv710

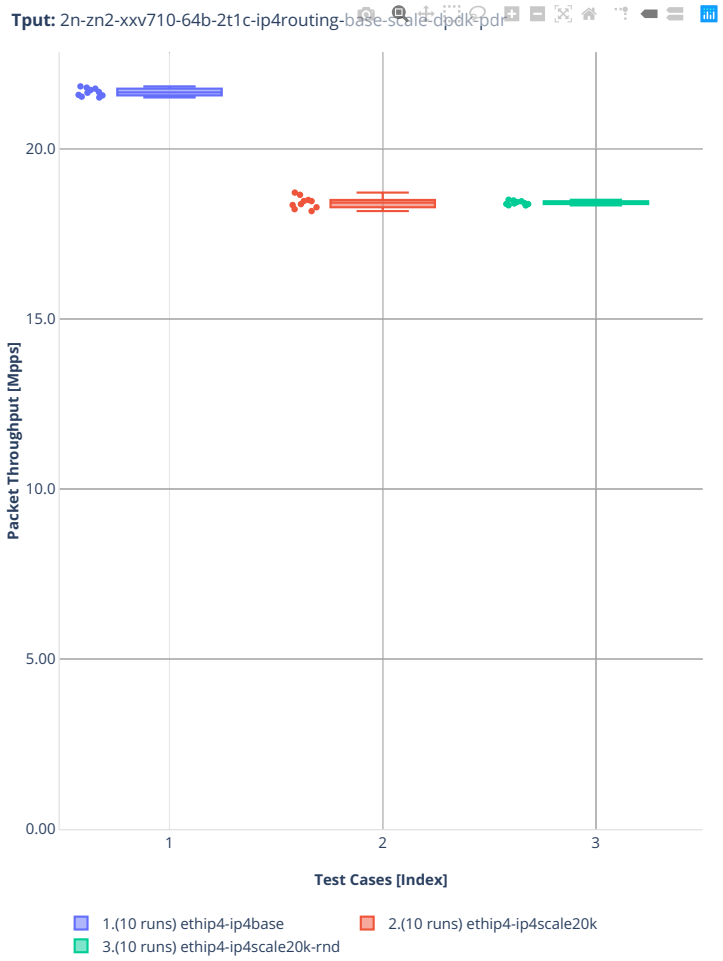
64b-2t1c-ip4routing-base-scale-avf



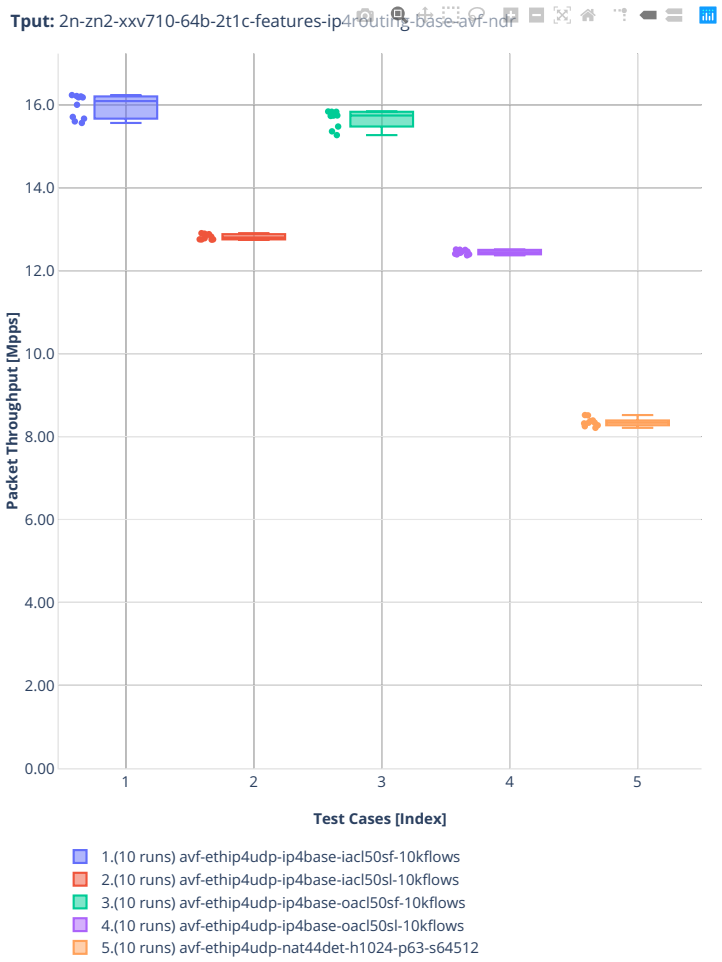

Tput: 2n-zn2-xxv710-64b-2t1c-ip4routing-base-scale-avf-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-ethip4-ip4base
2.(10 runs) avf-ethip4-ip4scale20k
3.(10 runs) avf-ethip4-ip4scale20k-rnd

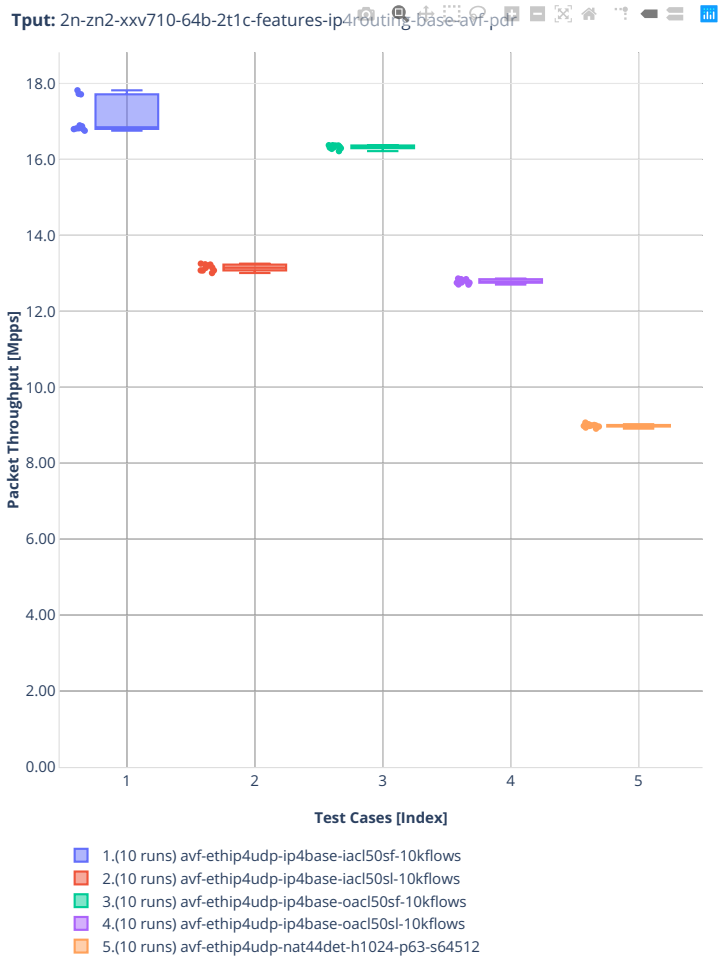
64b-2t1c-ip4routing-base-scale-dpdk




Tput: 2n-zn2-xxv710-64b-2t1c-ip4routing-base-scale-dpdk-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4-ip4base
2.(10 runs) ethip4-ip4scale20k
3.(10 runs) ethip4-ip4scale20k-rnd

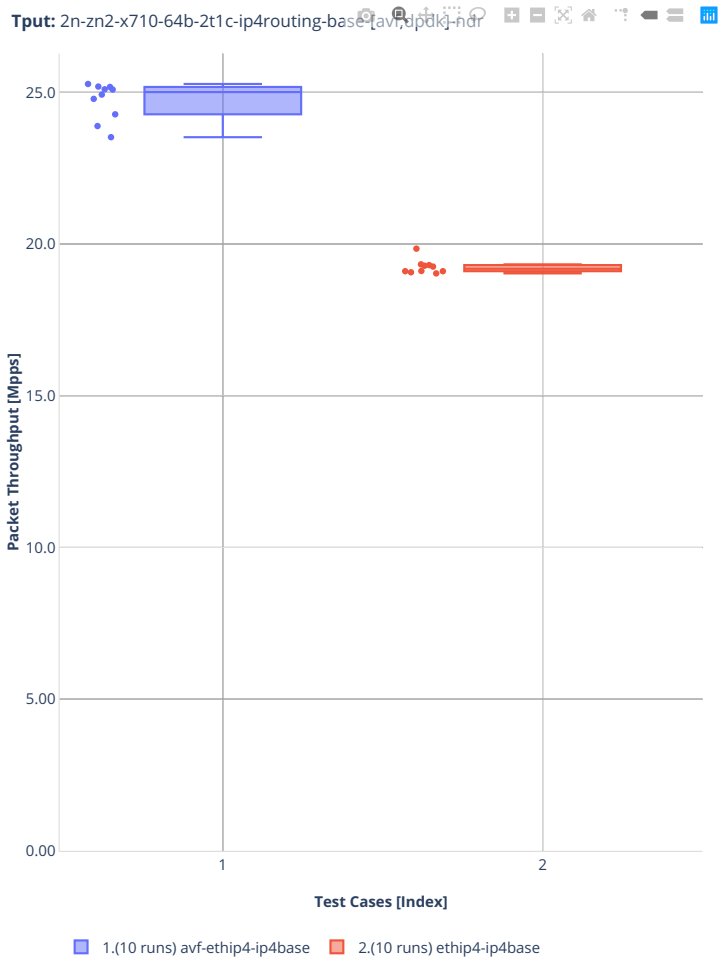
64b-2t1c-features-ip4routing-base-avf

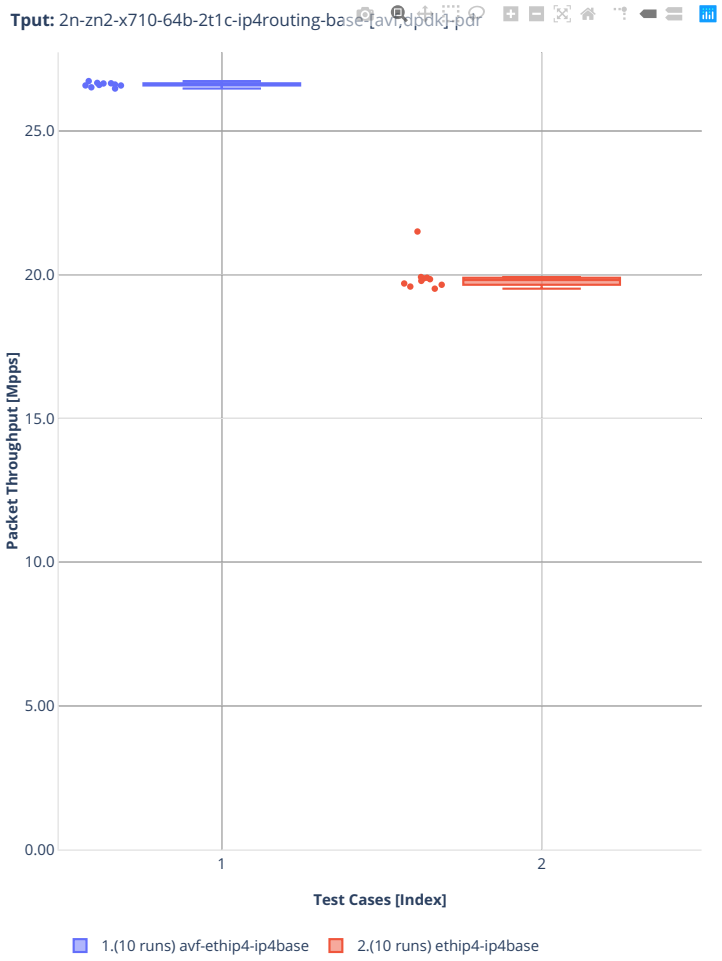



Tput: 2n-zn2-xxv710-64b-2t1c-features-ip4routing-base-avf-pdr
18.0
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-iacl50sf-10kflows
2.(10 runs) avf-ethip4udp-ip4base-iacl50sl-10kflows
3.(10 runs) avf-ethip4udp-ip4base-oacl50sf-10kflows
4.(10 runs) avf-ethip4udp-ip4base-oacl50sl-10kflows
5.(10 runs) avf-ethip4udp-nat44det-h1024-p63-s64512

2n-zn2-x710

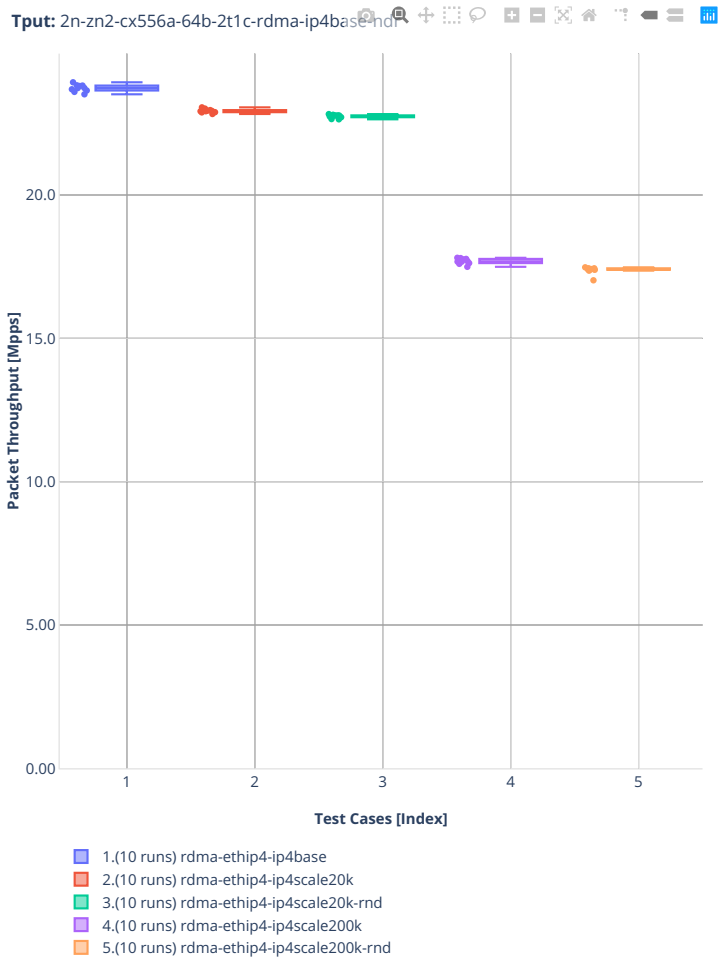
64b-2t1c-ip4routing-base-[avf,dpdk]

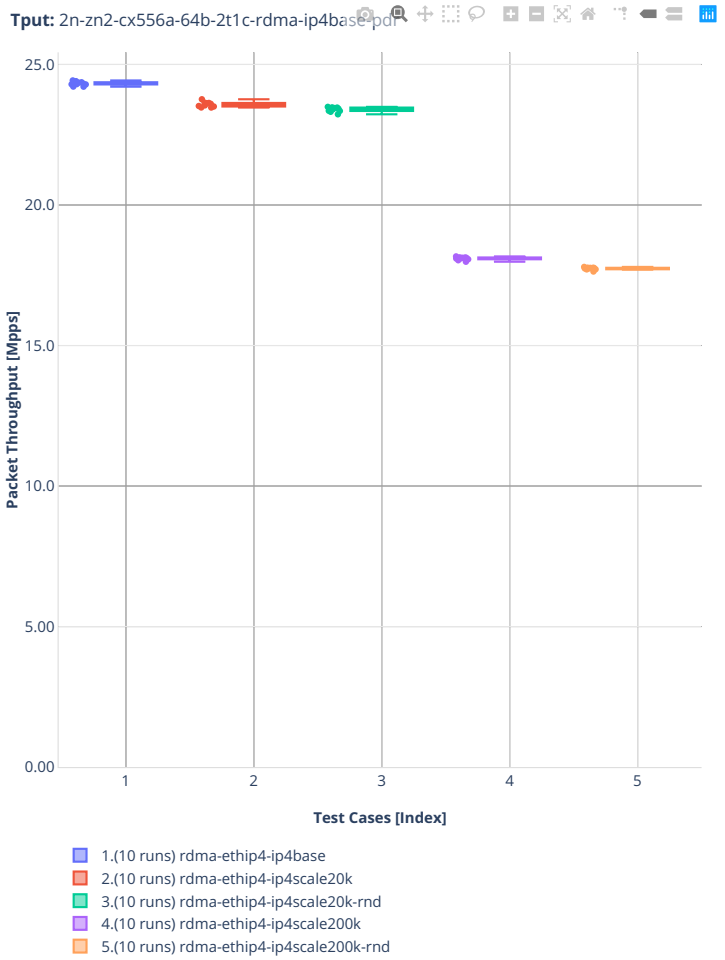



Tput: 2n-zn2-x710-64b-2t1c-ip4routing-base-[avf,dpdk]-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) avf-ethip4-ip4base
2.(10 runs) ethip4-ip4base

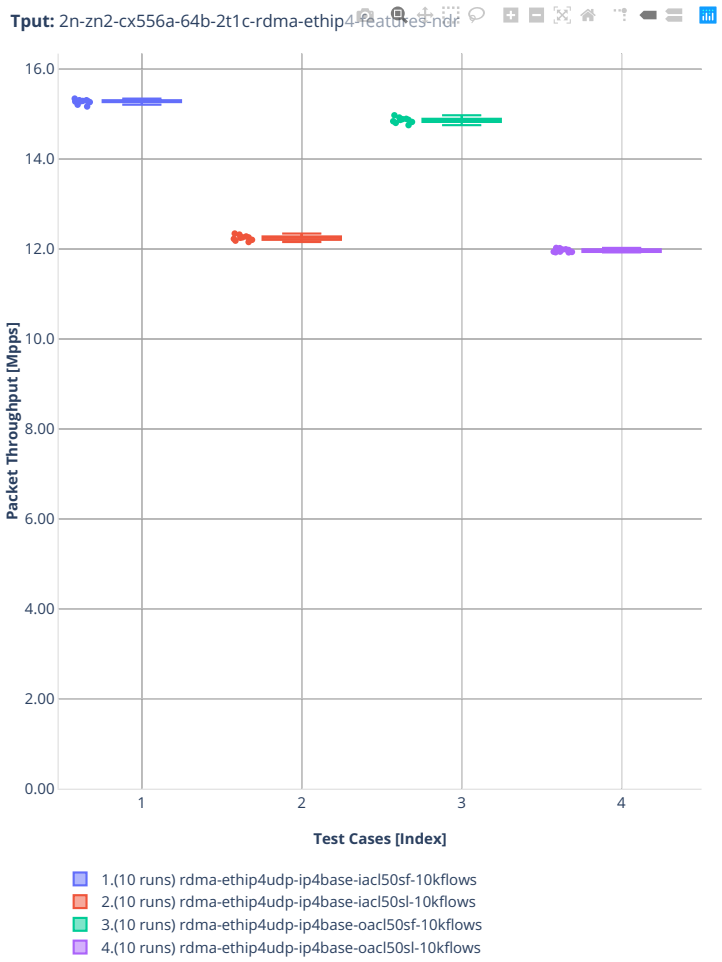
2n-zn2-cx556a

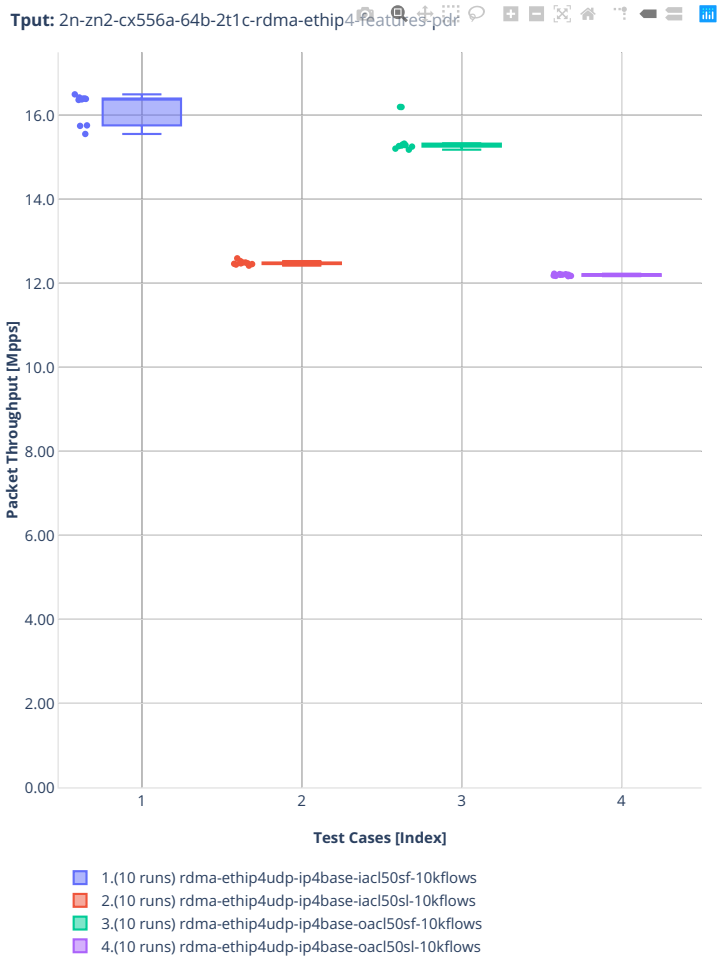
64b-2t1c-ip4routing-base-scale-rdma-core




Tput: 2n-zn2-cx556a-64b-2t1c-rdma-ip4base-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) rdma-ethip4-ip4base
2.(10 runs) rdma-ethip4-ip4scale20k
3.(10 runs) rdma-ethip4-ip4scale20k-rnd
4.(10 runs) rdma-ethip4-ip4scale200k
5.(10 runs) rdma-ethip4-ip4scale200k-rnd

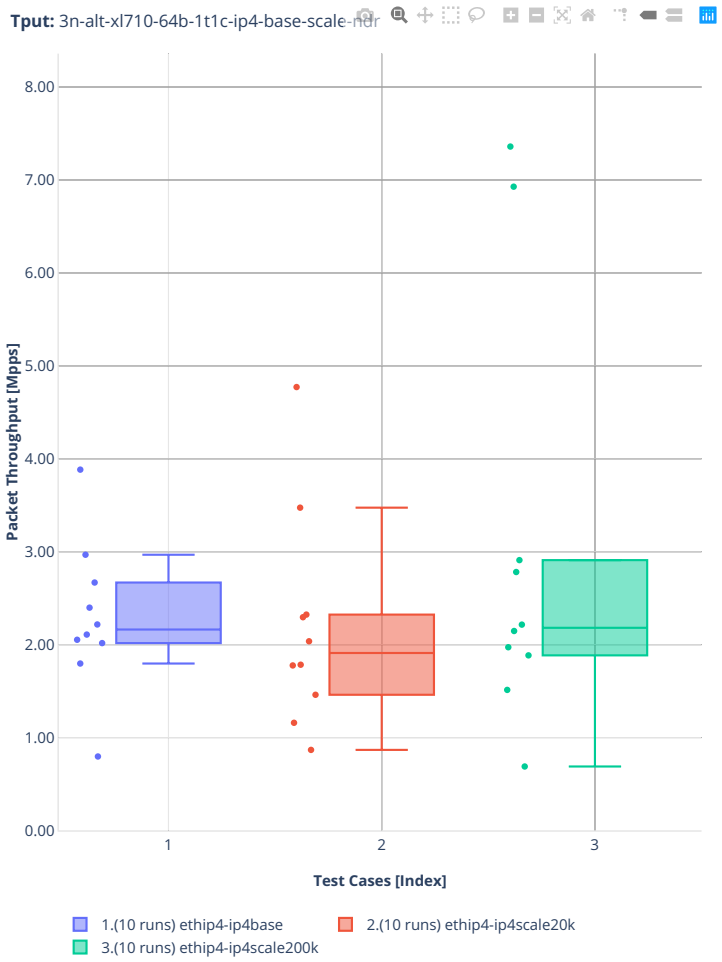
64b-2t1c-ip4routing-features

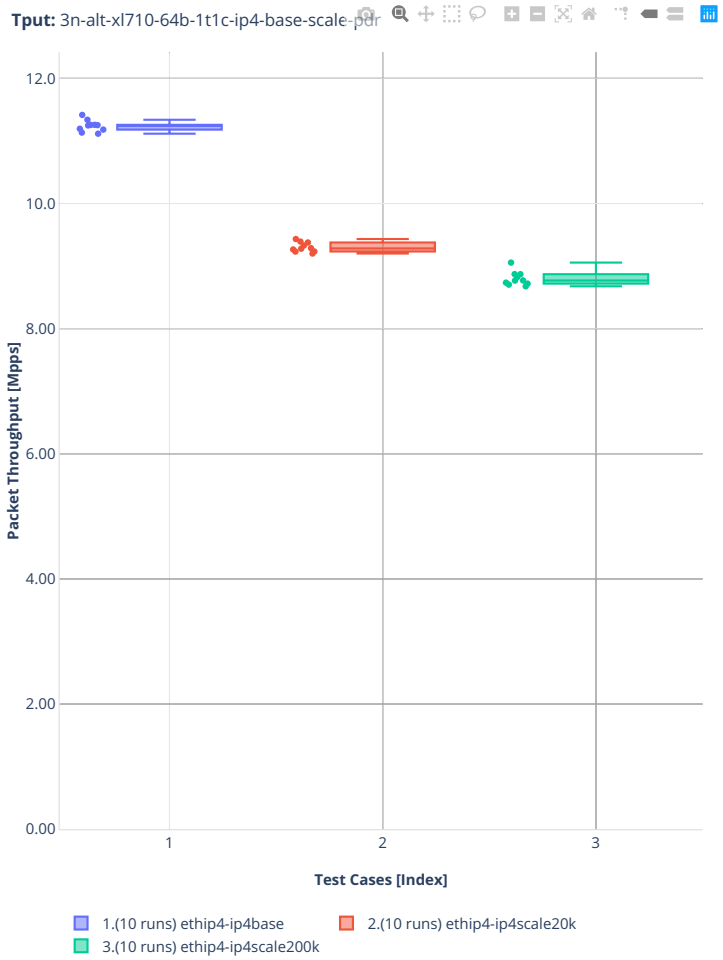



Tput: 2n-zn2-cx556a-64b-2t1c-rdma-ethip4-features-pdr
Packet Throughput [Mpps]
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) rdma-ethip4udp-ip4base-iacl50sf-10kflows
2.(10 runs) rdma-ethip4udp-ip4base-iacl50sl-10kflows
3.(10 runs) rdma-ethip4udp-ip4base-oacl50sf-10kflows
4.(10 runs) rdma-ethip4udp-ip4base-oacl50sl-10kflows

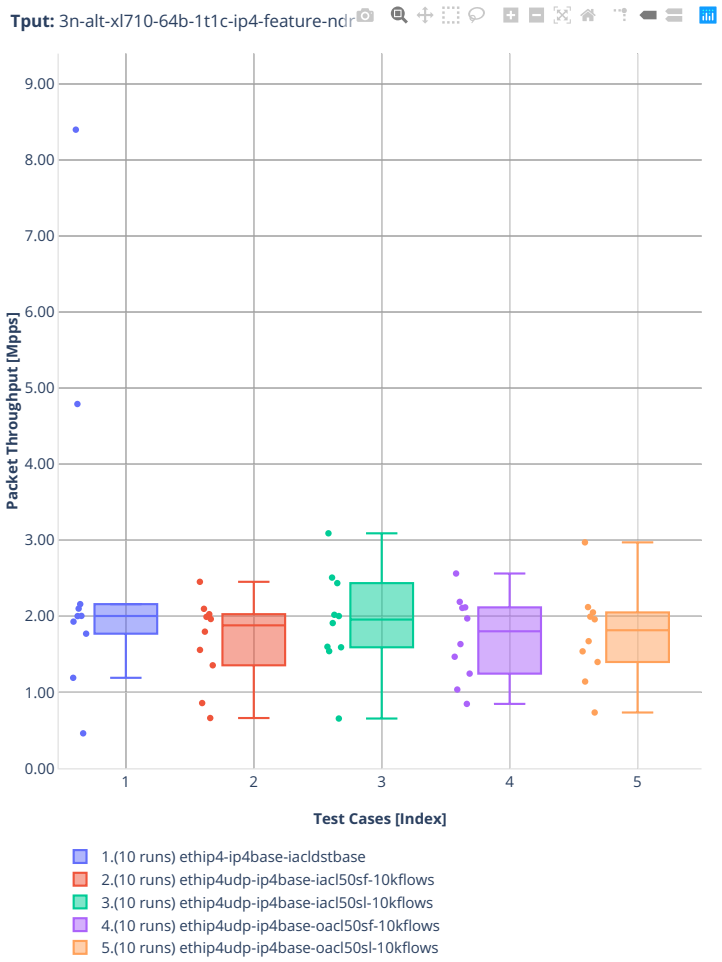
3n-alt-xl710

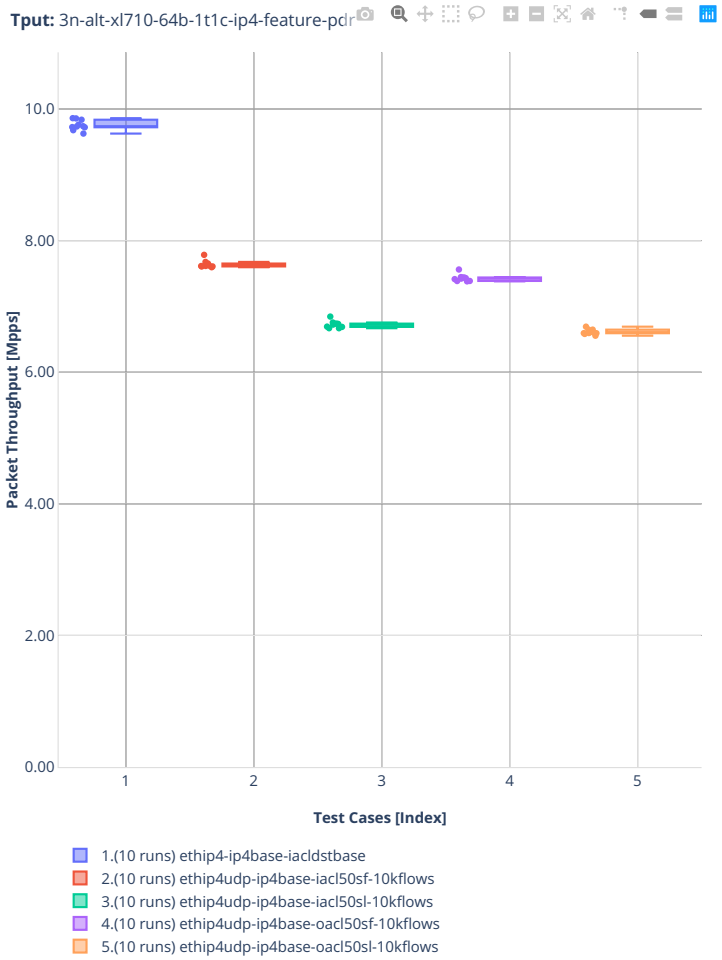
64b-1t1c-ip4routing-base-scale




Tput: 3n-alt-xl710-64b-1t1c-ip4-base-scale-pdr
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4-ip4base
2.(10 runs) ethip4-ip4scale20k
3.(10 runs) ethip4-ip4scale200k

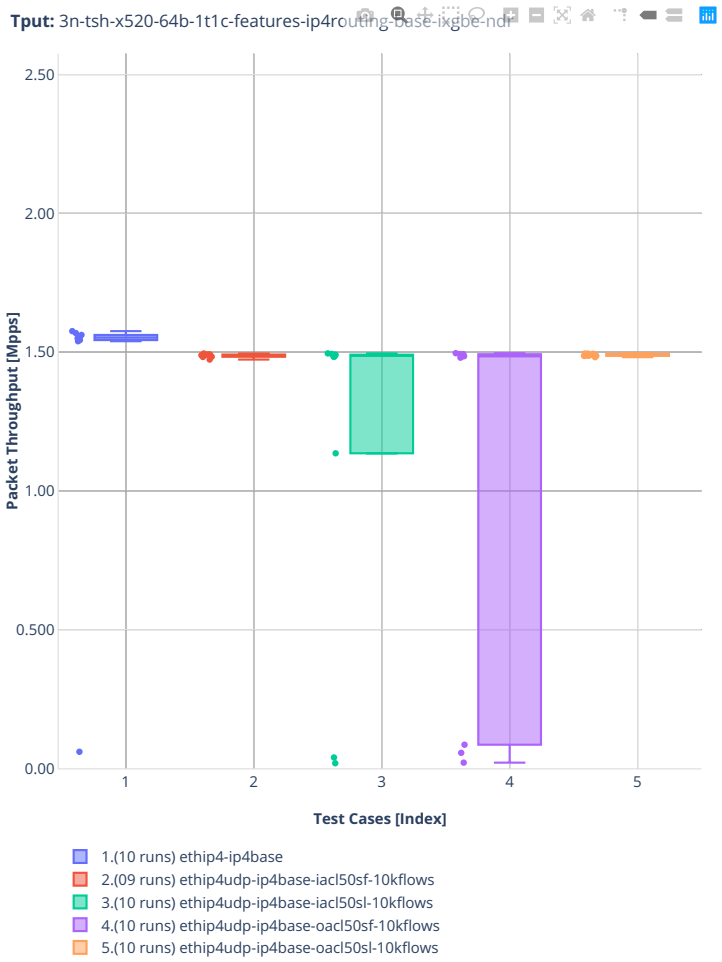
64b-1t1c-ip4routing-features

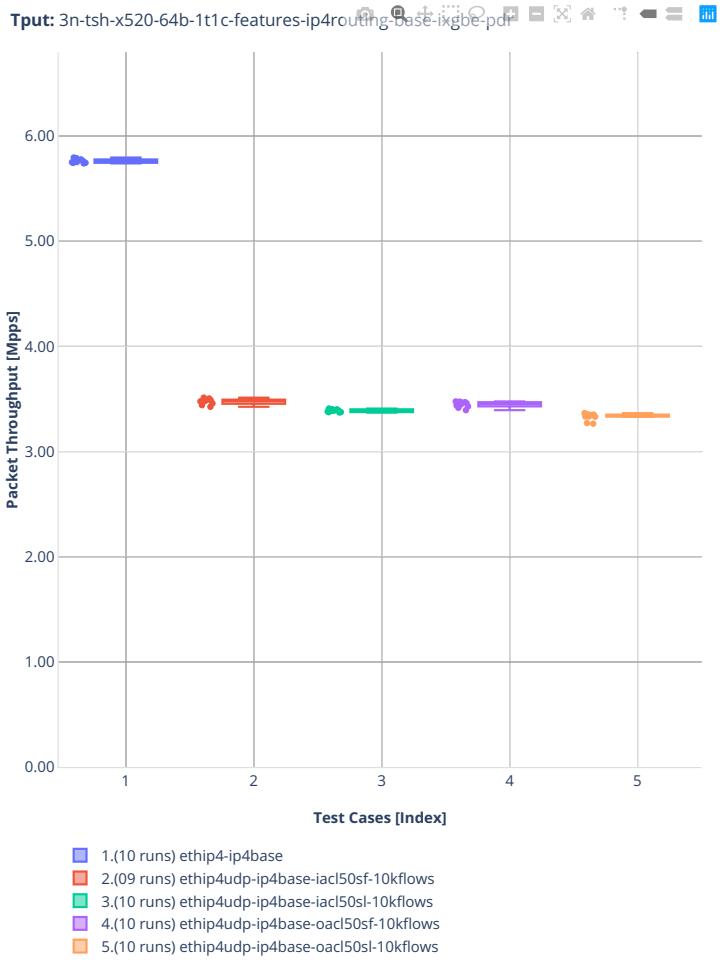



Tput: 3n-alt-xl710-64b-1t1c-ip4-feature-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) ethip4-ip4base-iacldstbase
2.(10 runs) ethip4udp-ip4base-iacl50sf-10kflows
3.(10 runs) ethip4udp-ip4base-iacl50sl-10kflows
4.(10 runs) ethip4udp-ip4base-oacl50sf-10kflows
5.(10 runs) ethip4udp-ip4base-oacl50sl-10kflows

3n-tsh-x520

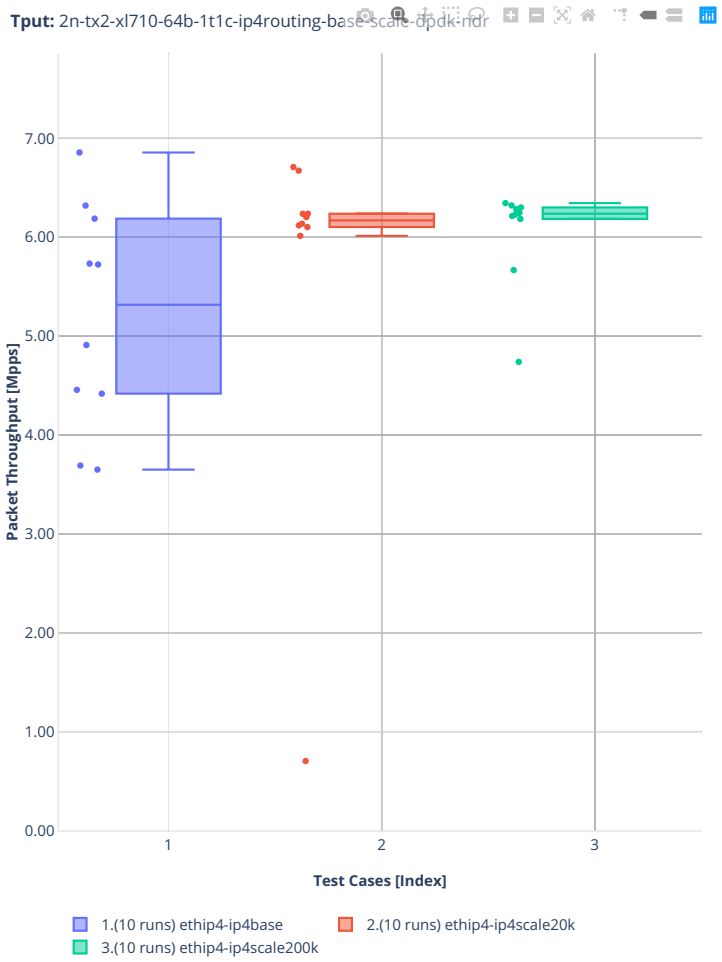
64b-1t1c-features-ip4routing-base-ixgbe

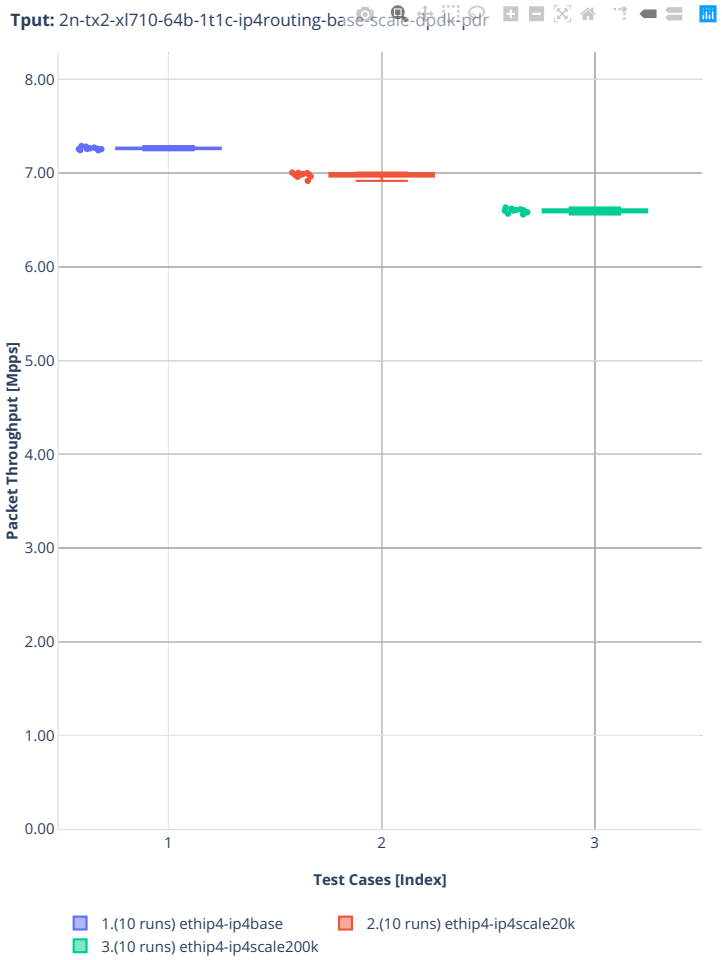



Tput: 3n-tsh-x520-64b-1t1c-features-ip4routing-base-ixgbe-pdr
Packet Throughput [Mpps]
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
5
Test Cases [Index]
1.(10 runs) ethip4-ip4base
2.(09 runs) ethip4udp-ip4base-iacl50sf-10kflows
3.(10 runs) ethip4udp-ip4base-iacl50sl-10kflows
4.(10 runs) ethip4udp-ip4base-oacl50sf-10kflows
5.(10 runs) ethip4udp-ip4base-oacl50sl-10kflows

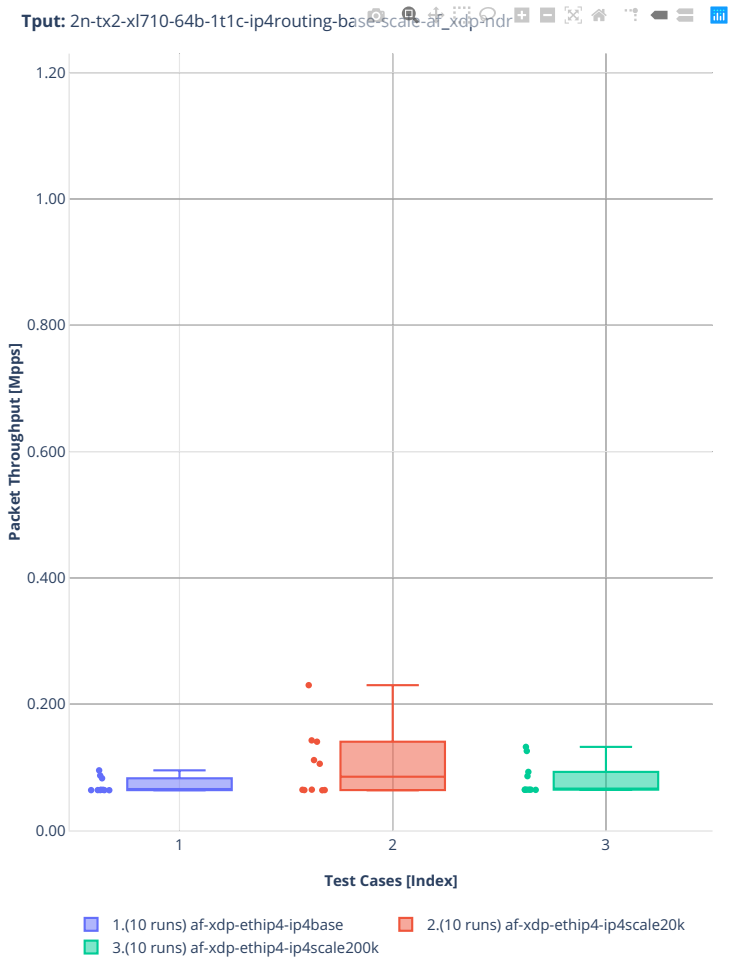
2n-tx2-xl710

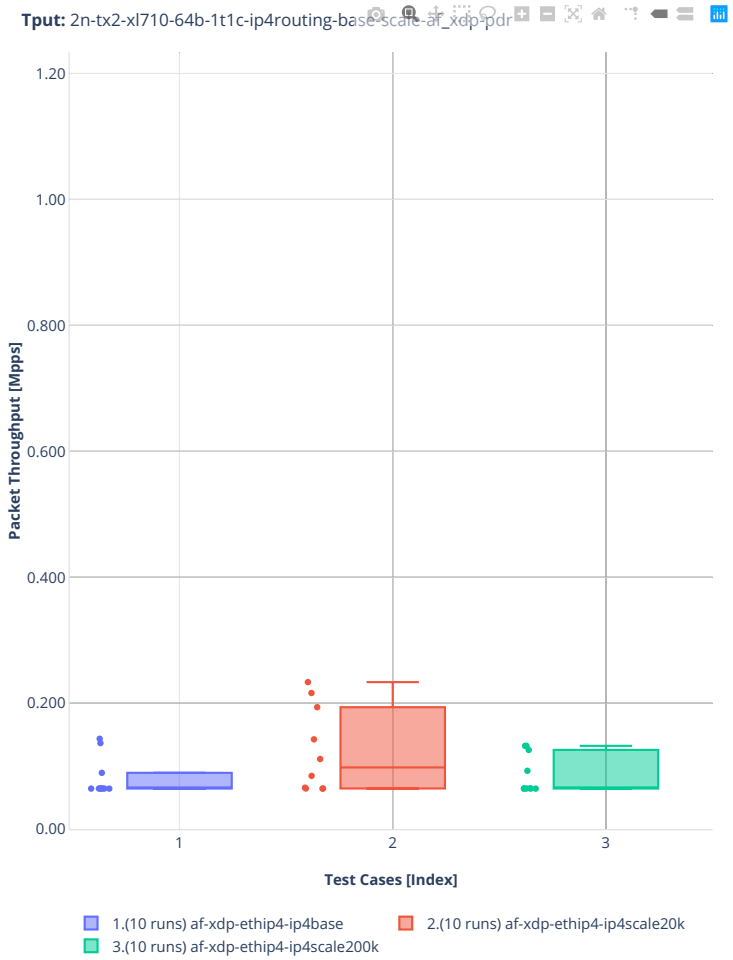
64b-1t1c-ip4routing-base-scale-dpdk



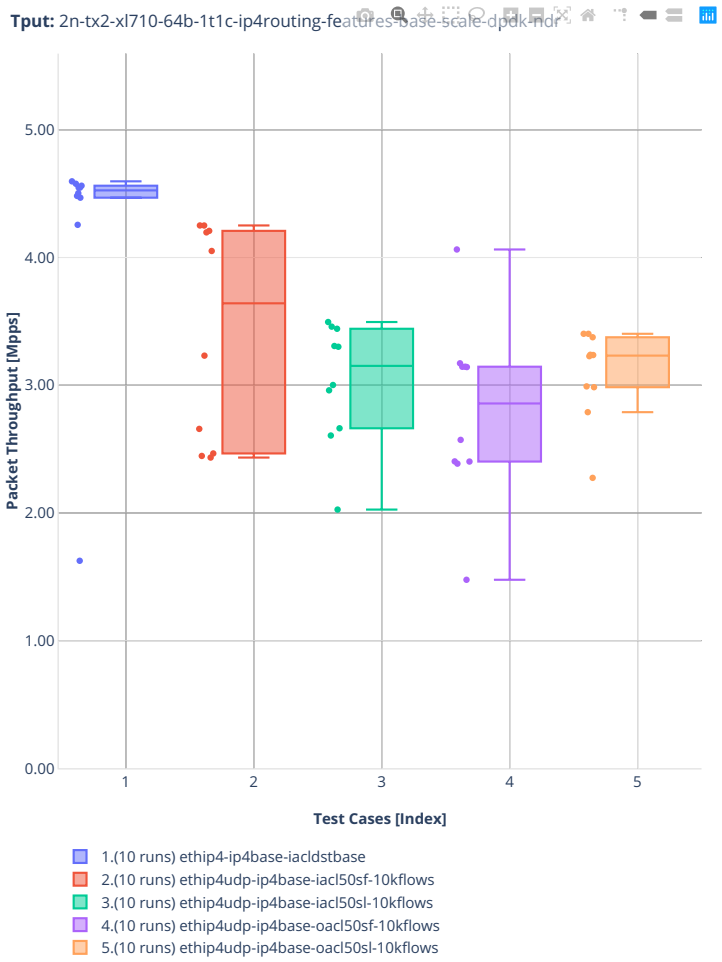

Tput: 2n-tx2-xl710-64b-1t1c-ip4routing-base-scale-dpdk-pdr
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4-ip4base
2.(10 runs) ethip4-ip4scale20k
3.(10 runs) ethip4-ip4scale200k

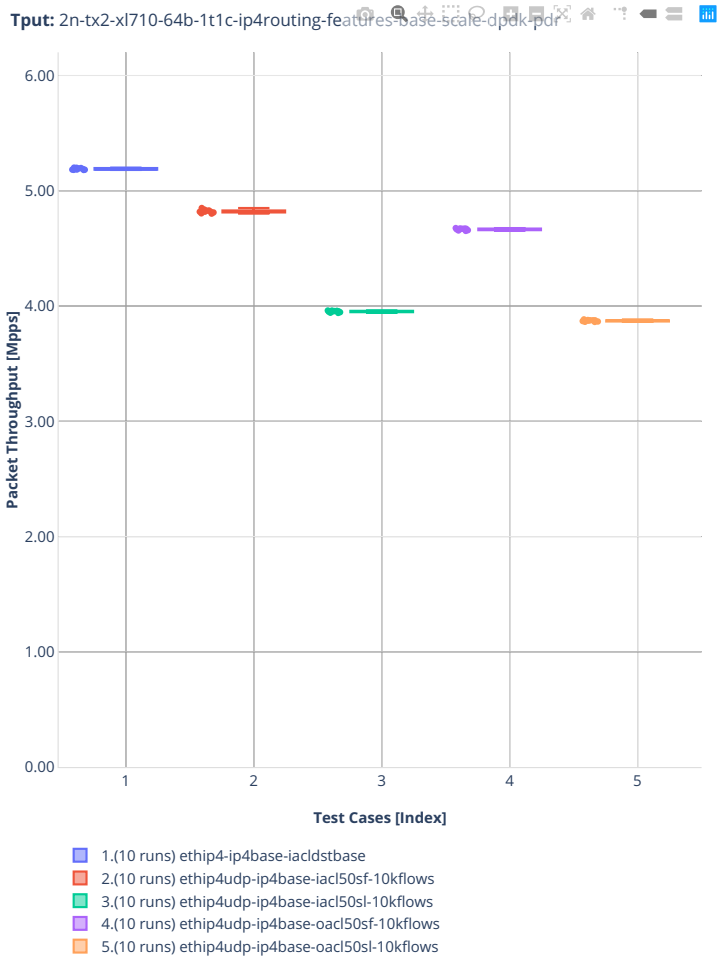
64b-1t1c-ip4routing-base-scale-af-xdp




Tput: 2n-tx2-xl710-64b-1t1c-ip4routing-base-scale-af_xdp-pdr
1.20
1.00
0.800
0.600
0.400
0.200
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) af-xdp-ethip4-ip4base
2.(10 runs) af-xdp-ethip4-ip4scale20k
3.(10 runs) af-xdp-ethip4-ip4scale200k

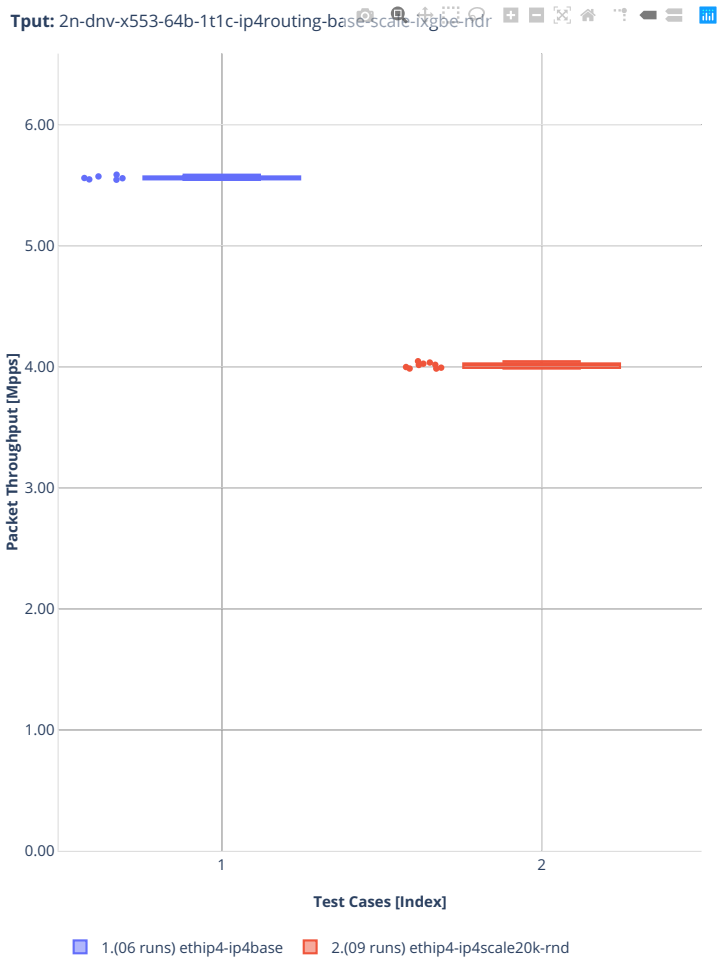
64b-1t1c-features-ip4routing-base-dpdk

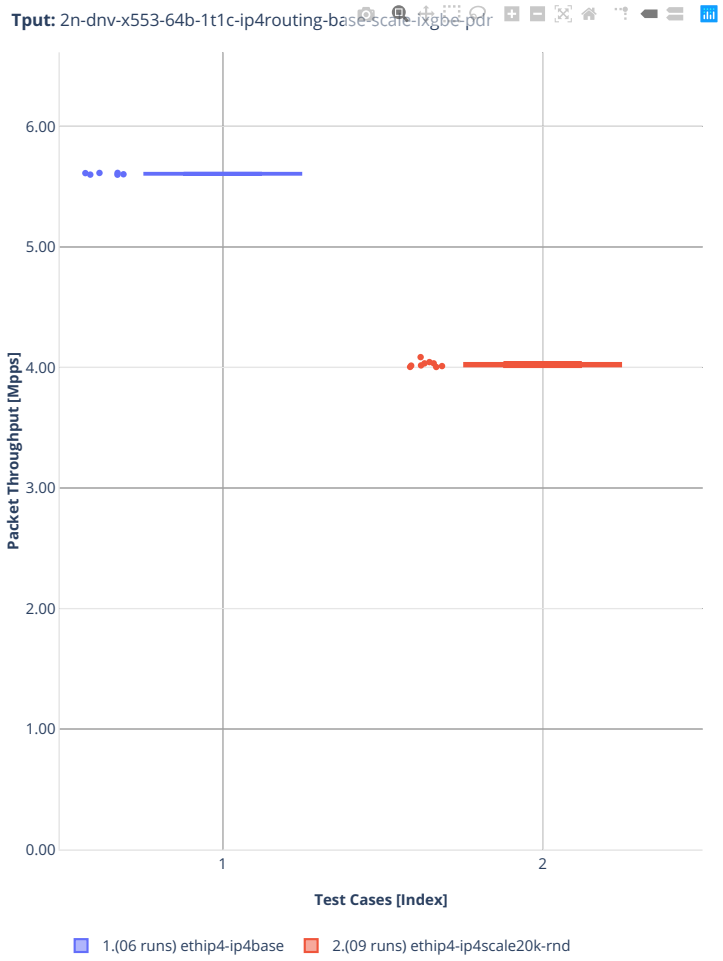



Tput: 2n-tx2-xl710-64b-1t1c-ip4routing-features-base-scale-dpdk-pdr
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) ethip4-ip4base-iacldstbase
2.(10 runs) ethip4udp-ip4base-iacl50sf-10kflows
3.(10 runs) ethip4udp-ip4base-iacl50sl-10kflows
4.(10 runs) ethip4udp-ip4base-oacl50sf-10kflows
5.(10 runs) ethip4udp-ip4base-oacl50sl-10kflows

2n-dnv-x553

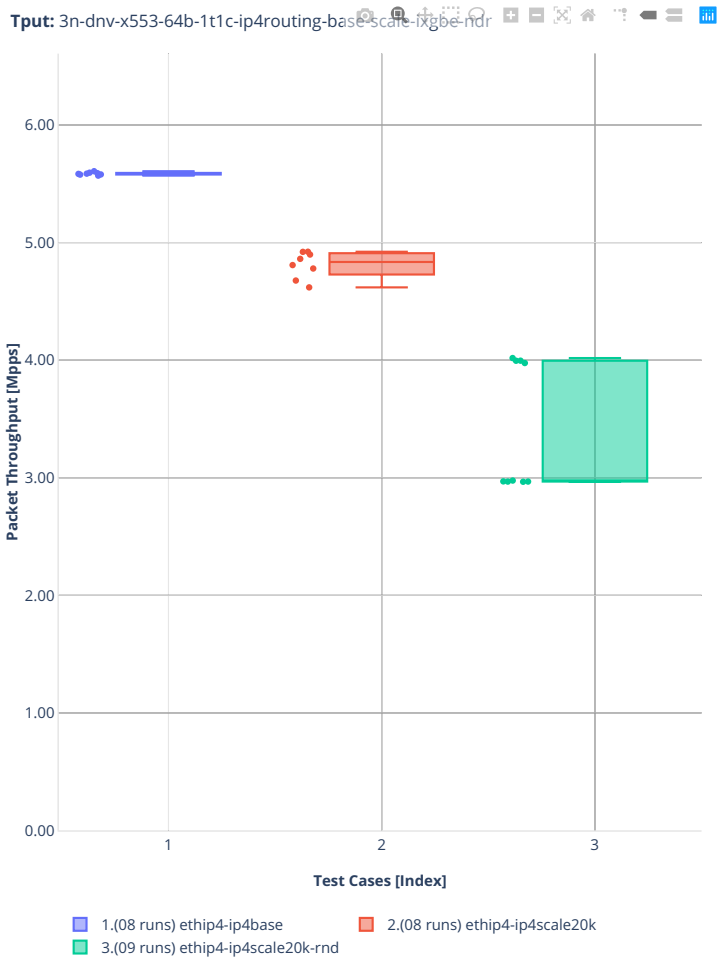
64b-1t1c-ip4routing-base-scale-ixgbe

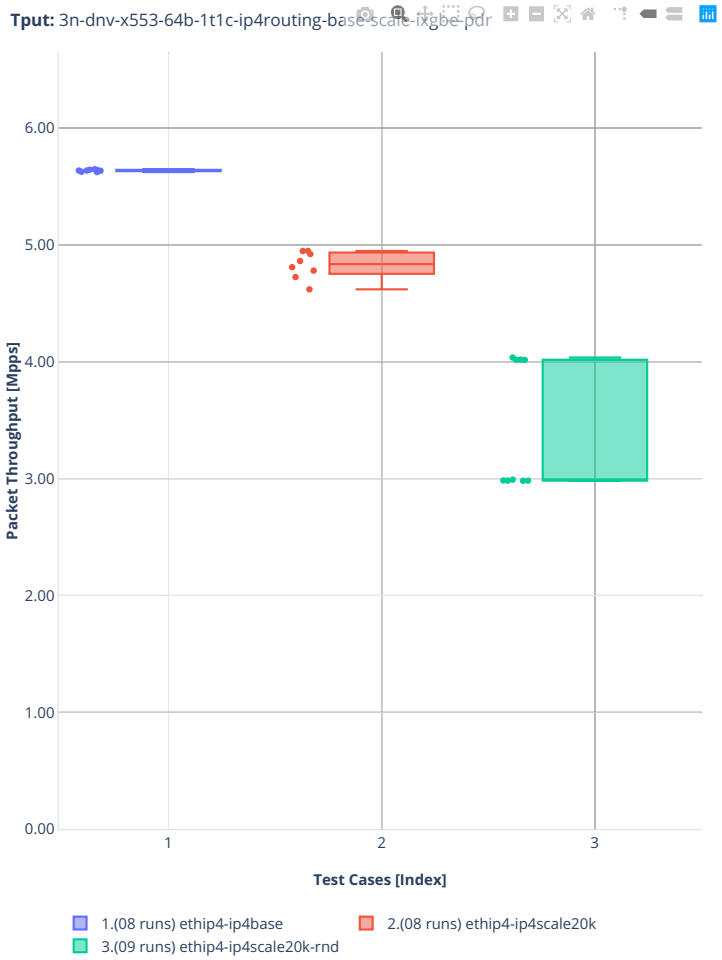



Tput: 2n-dnv-x553-64b-1t1c-ip4routing-base-scale-ixgbe-pdr
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(06 runs) ethip4-ip4base
2.(09 runs) ethip4-ip4scale20k-rnd

3n-dnv-x553

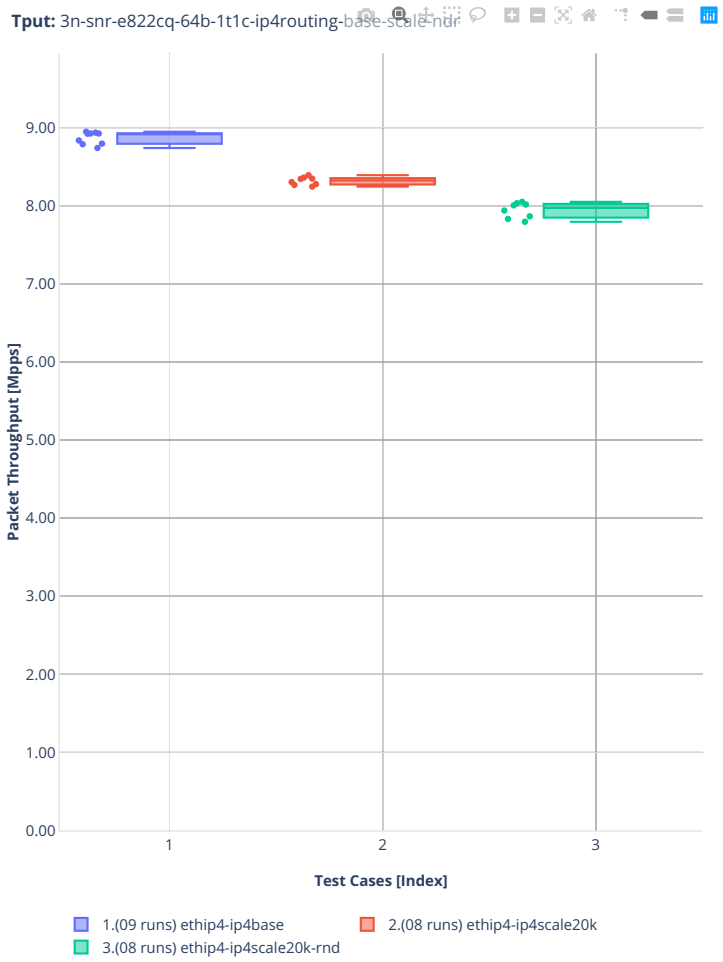
64b-1t1c-ip4routing-base-scale-ixgbe

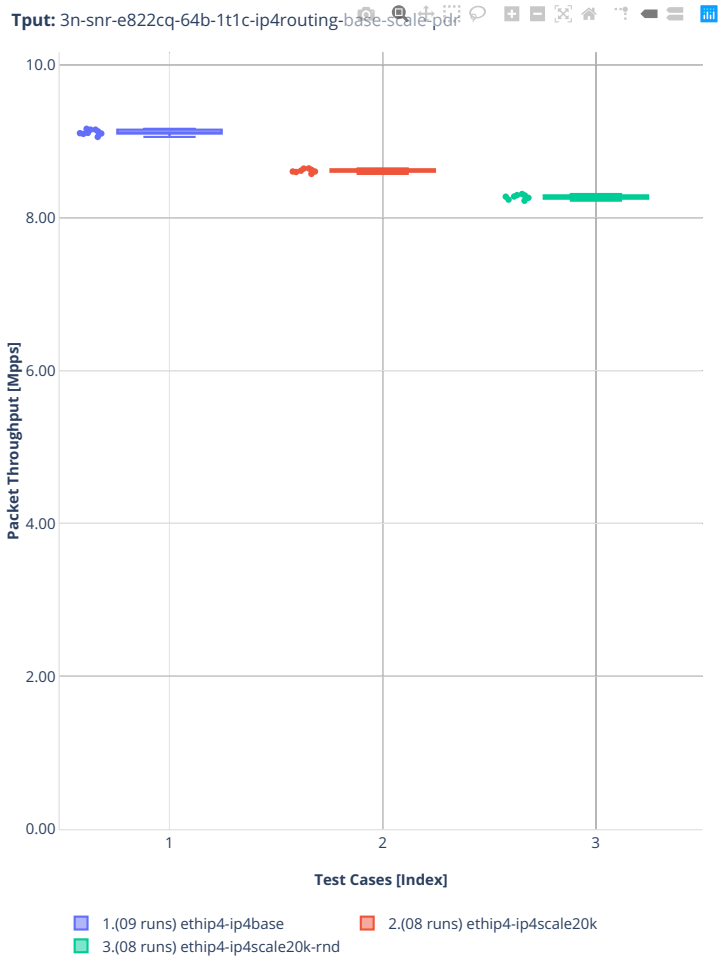



Tput: 3n-dnv-x553-64b-1t1c-ip4routing-base-scale-ixgbe-pdr
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(08 runs) ethip4-ip4base
2.(08 runs) ethip4-ip4scale20k
3.(09 runs) ethip4-ip4scale20k-rnd

3n-snr-e822cq

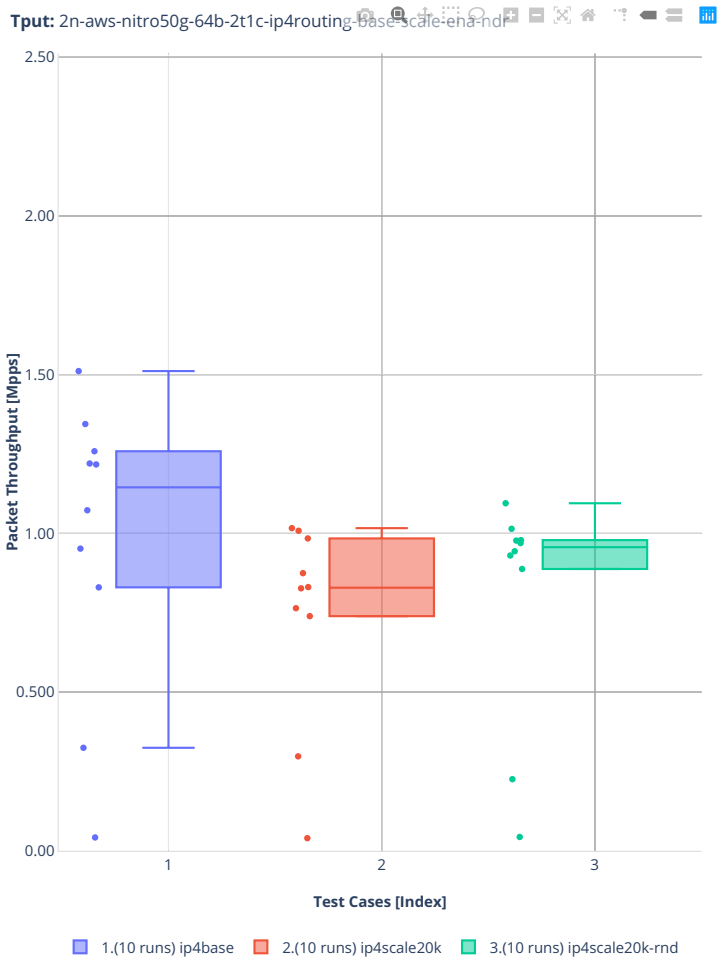
64b-1t1c-ip4routing-base-scale

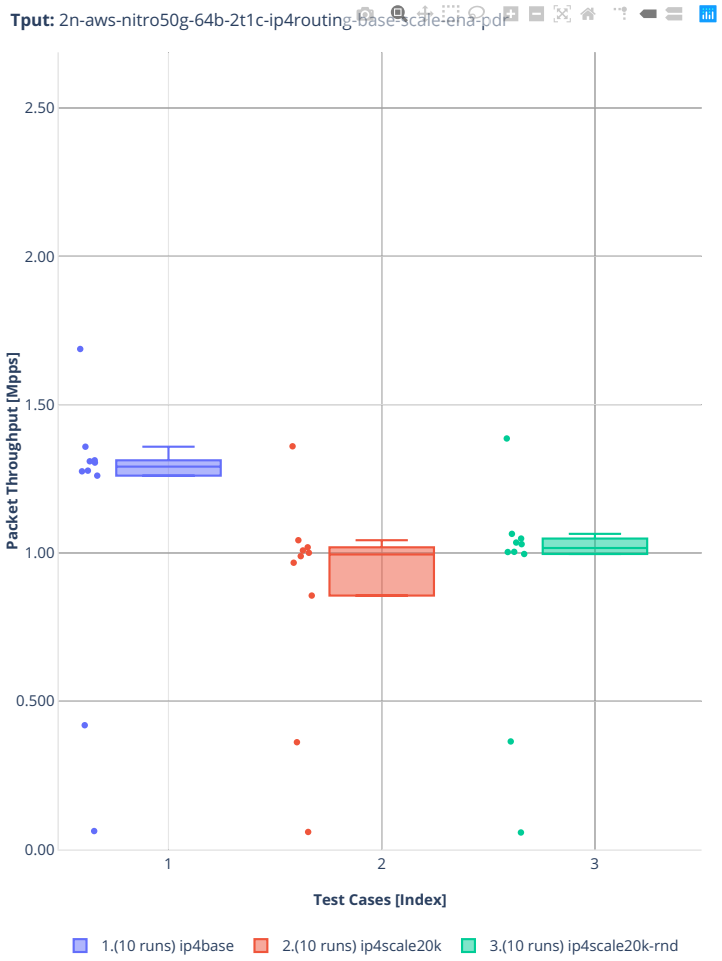



Tput: 3n-snr-e822cq-64b-1t1c-ip4routing-base-scale-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(09 runs) ethip4-ip4base
2.(08 runs) ethip4-ip4scale20k
3.(08 runs) ethip4-ip4scale20k-rnd

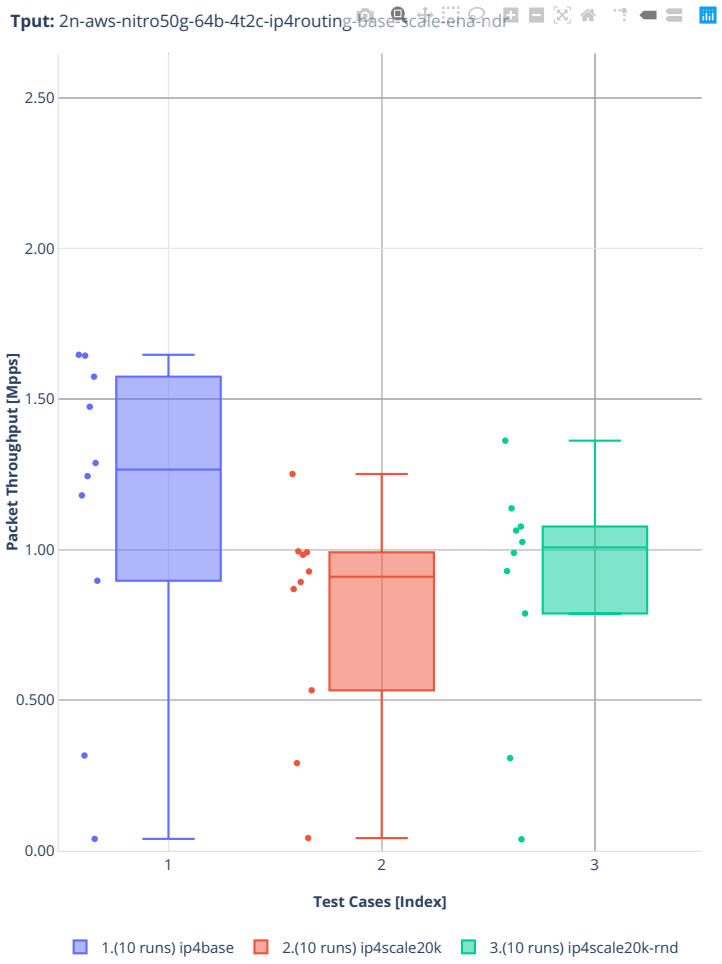
2n-aws-nitro50g

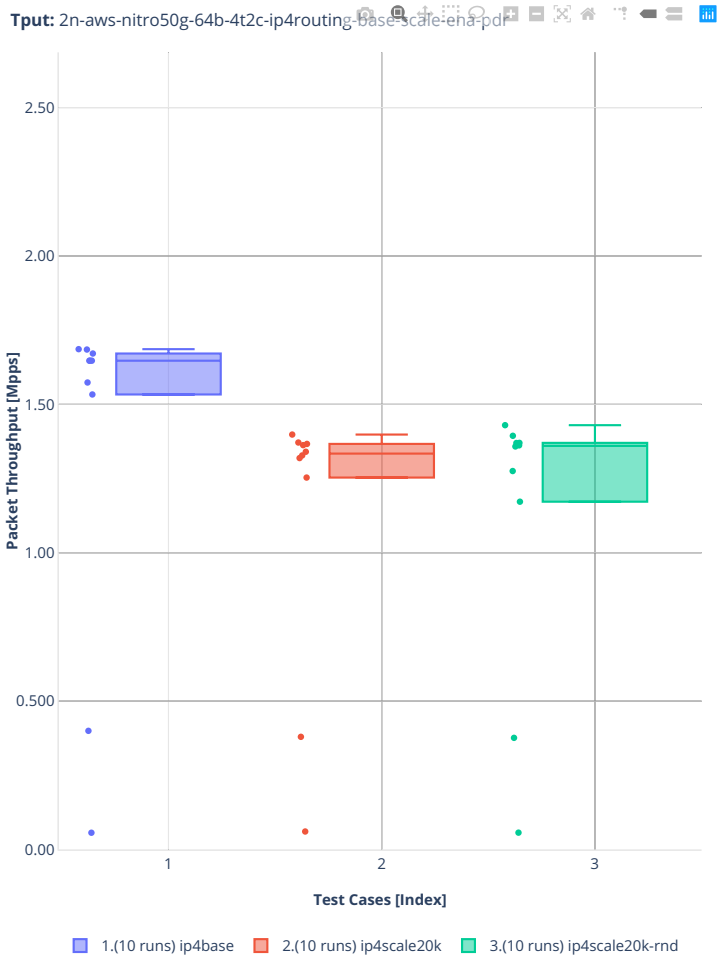
64b-2t1c-ip4routing-base-scale-ena



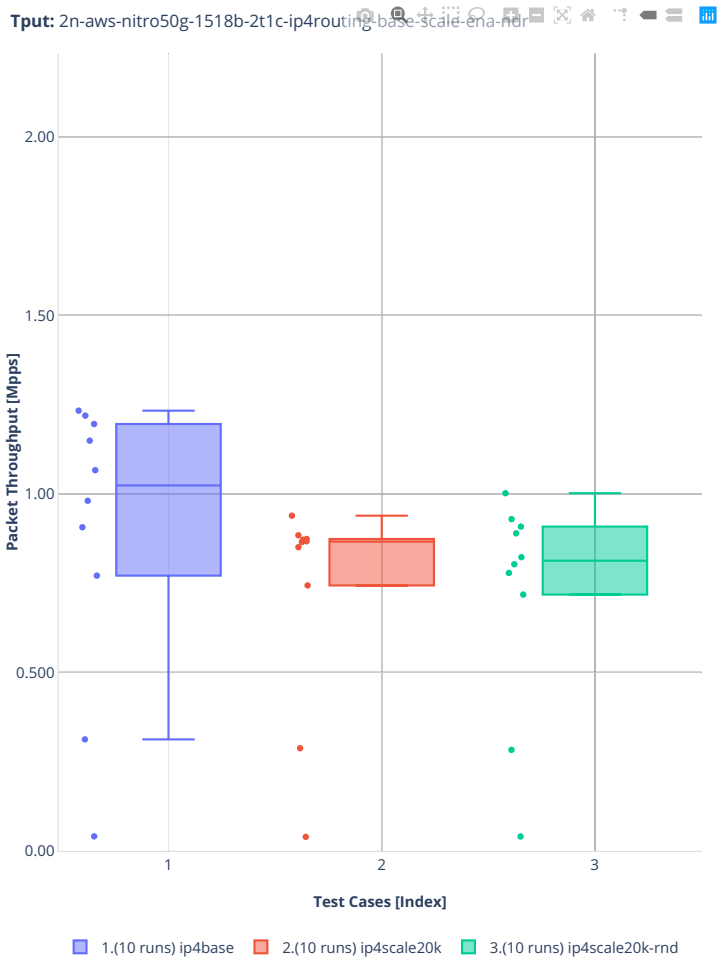

Tput: 2n-aws-nitro50g-64b-2t1c-ip4routing-base-scale-ena-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip4base
2.(10 runs) ip4scale20k
3.(10 runs) ip4scale20k-rnd

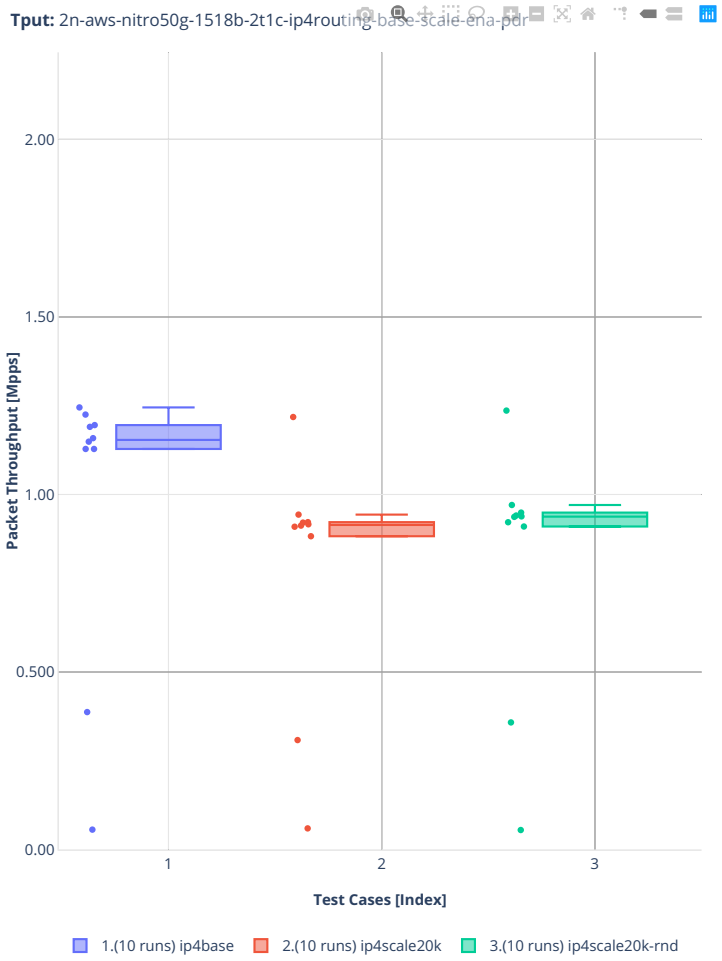
64b-4t2c-ip4routing-base-scale-ena



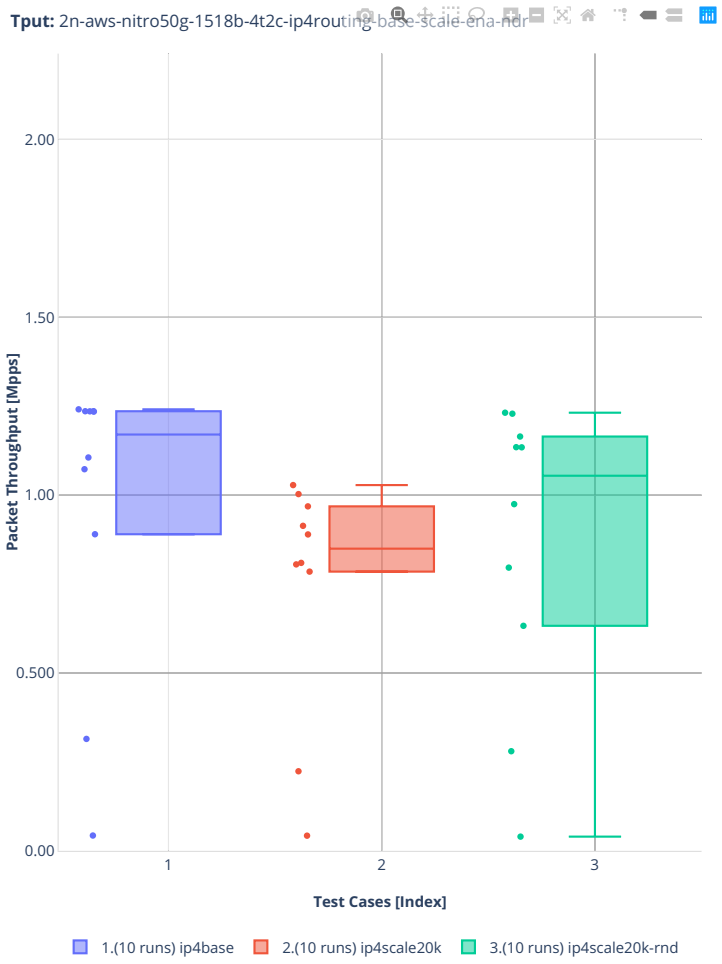

Tput: 2n-aws-nitro50g-64b-4t2c-ip4routing-base-scale-ena-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip4base
2.(10 runs) ip4scale20k
3.(10 runs) ip4scale20k-rnd

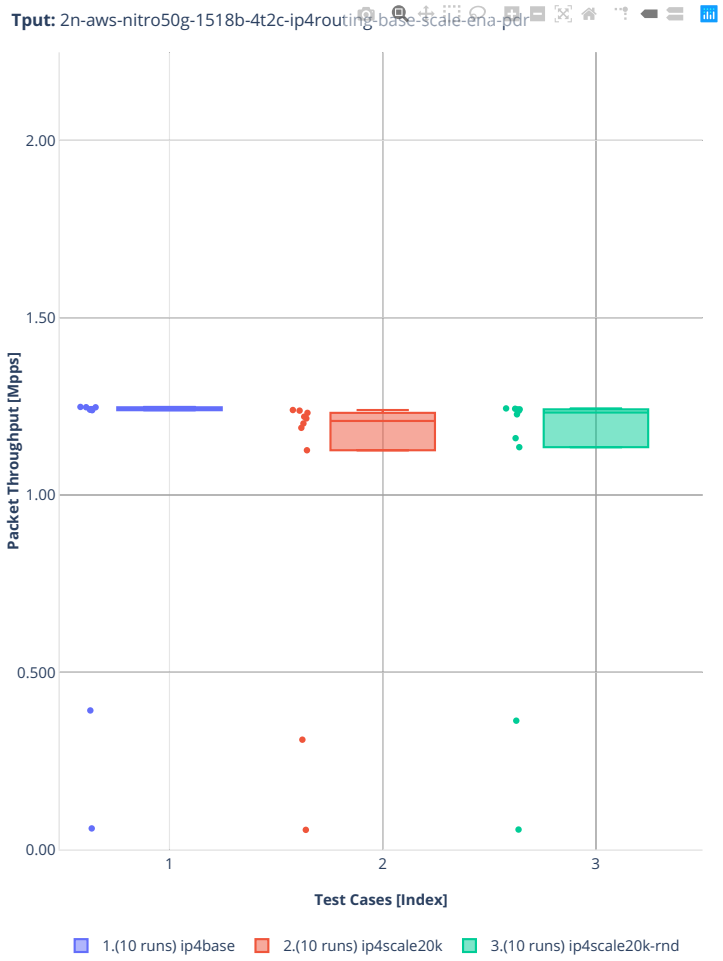
1518b-2t1c-ip4routing-base-scale-ena




Tput: 2n-aws-nitro50g-1518b-2t1c-ip4routing-base-scale-ena-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip4base
2.(10 runs) ip4scale20k
3.(10 runs) ip4scale20k-rnd

1518b-4t2c-ip4routing-base-scale-ena




Tput: 2n-aws-nitro50g-1518b-4t2c-ip4routing-base-scale-ena-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip4base
2.(10 runs) ip4scale20k
3.(10 runs) ip4scale20k-rnd

2.3.3 IPv6 Routing

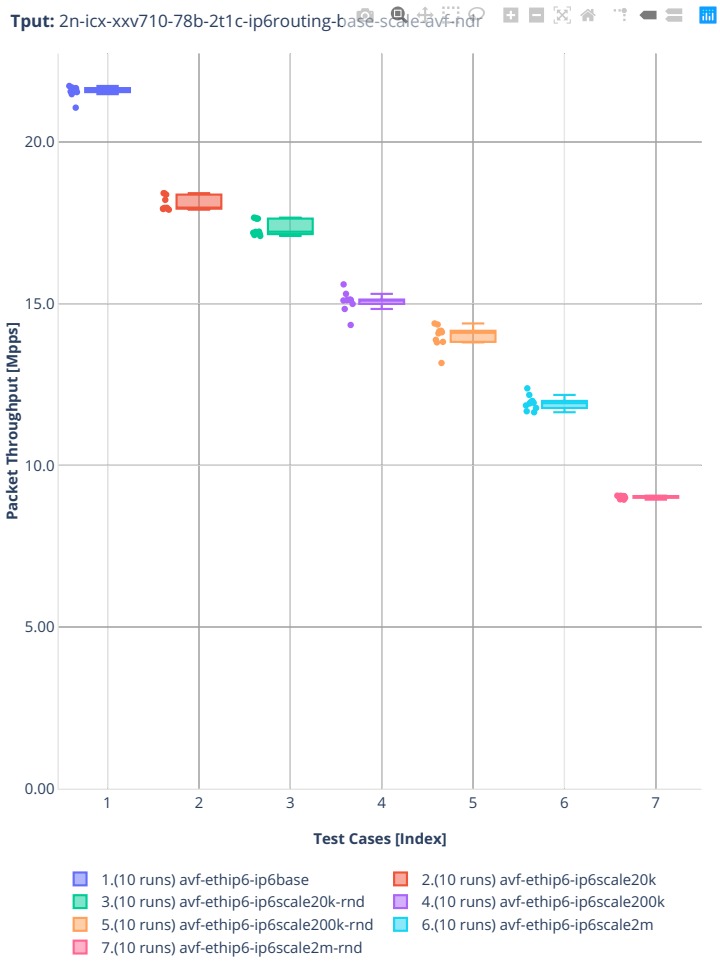
Following sections include summary graphs of VPP Phy-to-Phy performance with IPv6 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

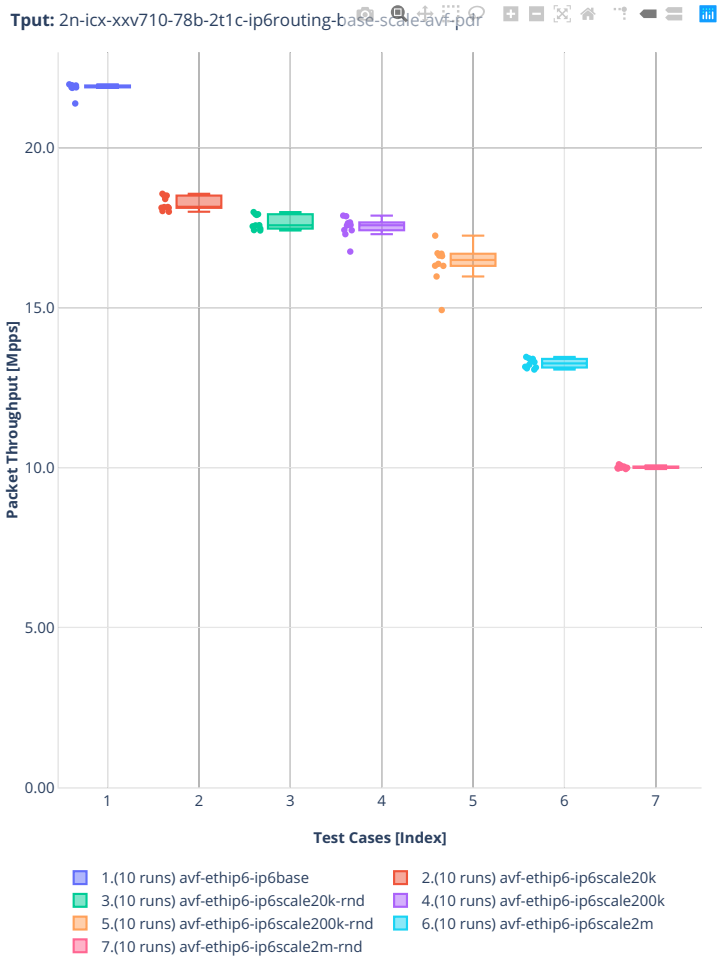
CSIT source code for the test cases used for plots can be found in CSIT git repository[116].

[116] https://git.fd.io/csit/tree/tests/vpp/perf/ip6?h=rls2210

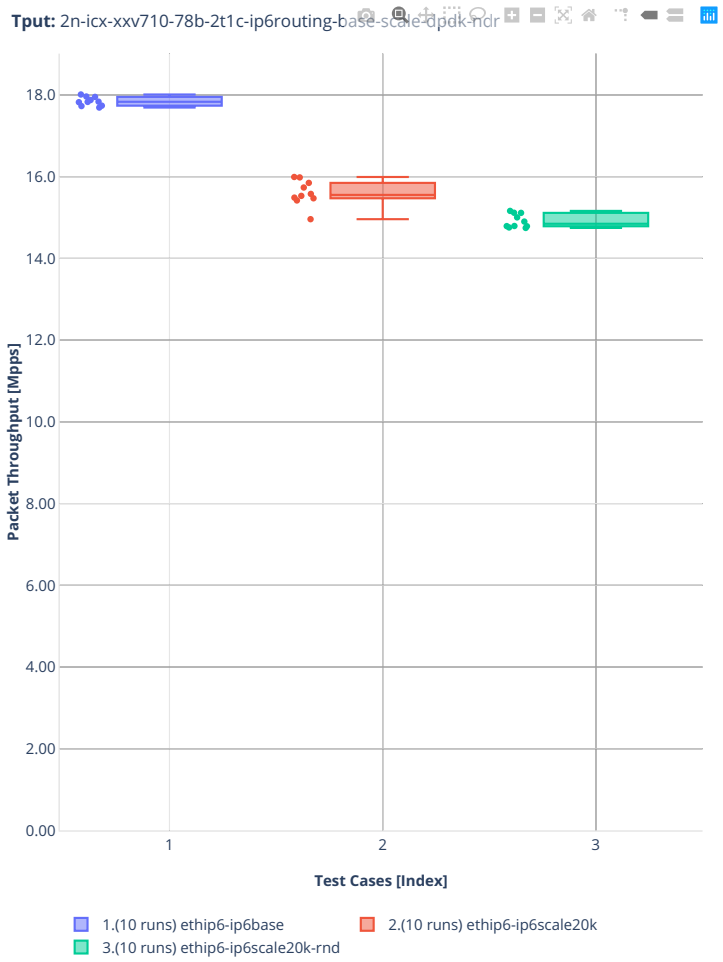
2n-icx-xxv710

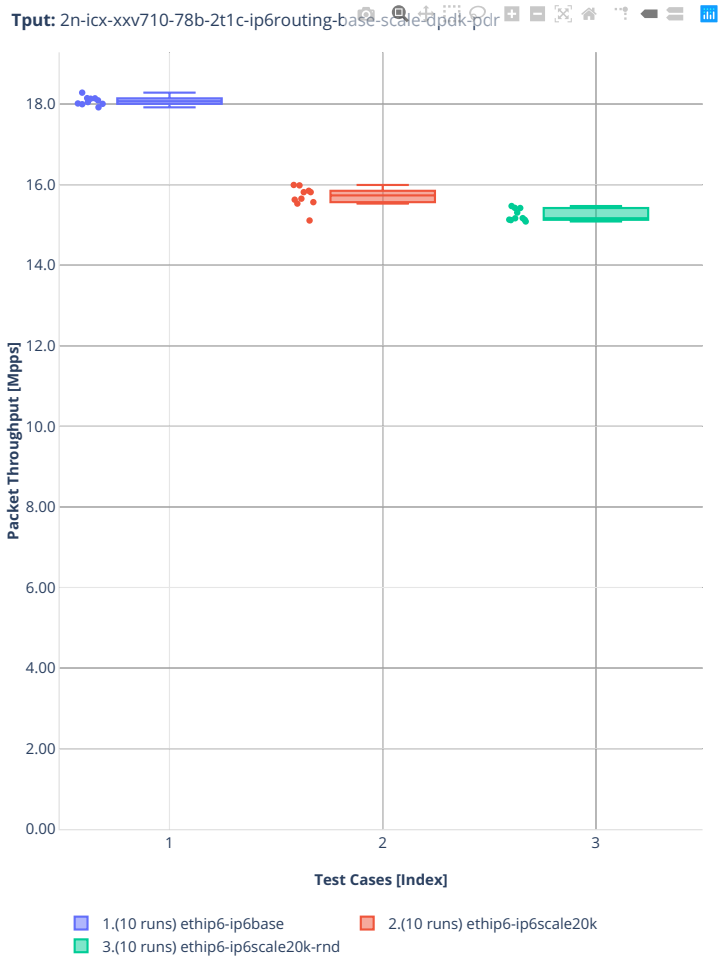
78b-2t1c-ip6routing-base-scale-avf



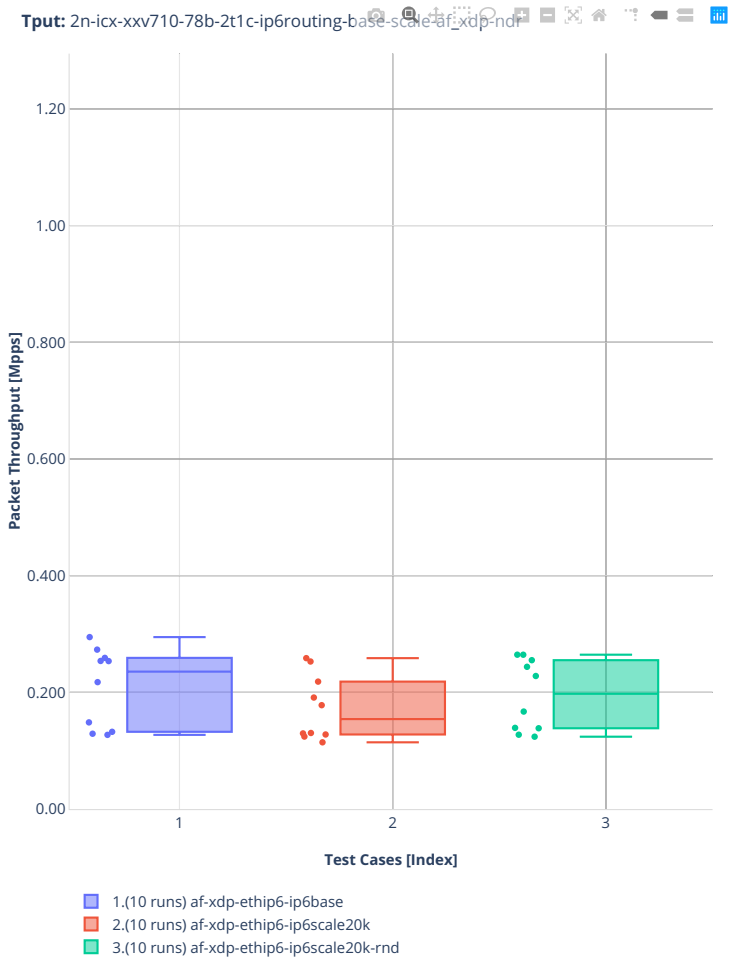

Tput: 2n-icx-xxv710-78b-2t1c-ip6routing-base-scale-avf-pdr
Packet Throughput [Mpps]
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
7
Test Cases [Index]
1.(10 runs) avf-ethip6-ip6base
2.(10 runs) avf-ethip6-ip6scale20k
3.(10 runs) avf-ethip6-ip6scale20k-rnd
4.(10 runs) avf-ethip6-ip6scale200k
5.(10 runs) avf-ethip6-ip6scale200k-rnd
6.(10 runs) avf-ethip6-ip6scale2m
7.(10 runs) avf-ethip6-ip6scale2m-rnd

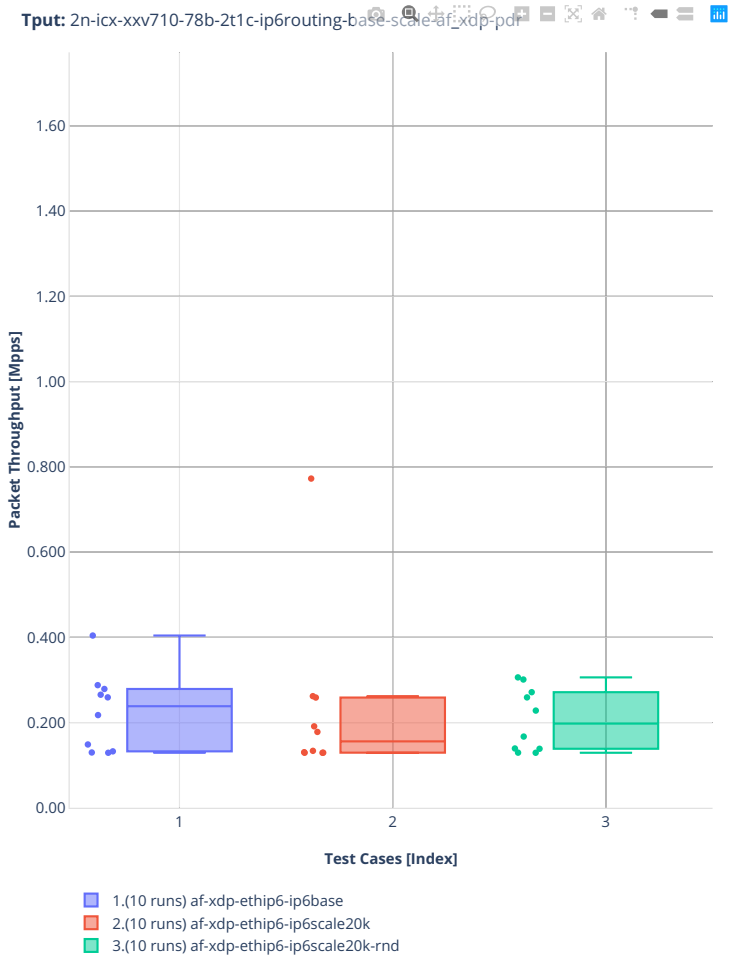
78b-2t1c-ip6routing-base-scale-dpdk




Tput: 2n-icx-xxv710-78b-2t1c-ip6routing-base-scale-dpdk-pdr
18.0
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip6-ip6base
2.(10 runs) ethip6-ip6scale20k
3.(10 runs) ethip6-ip6scale20k-rnd

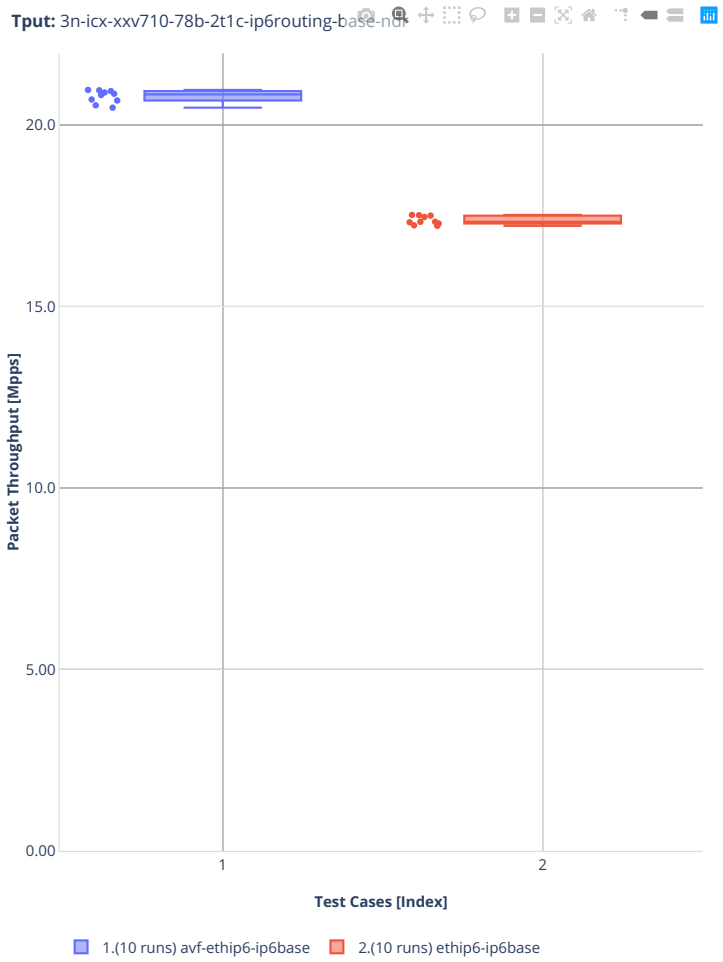
78b-2t1c-ip6routing-base-scale-af_xdp

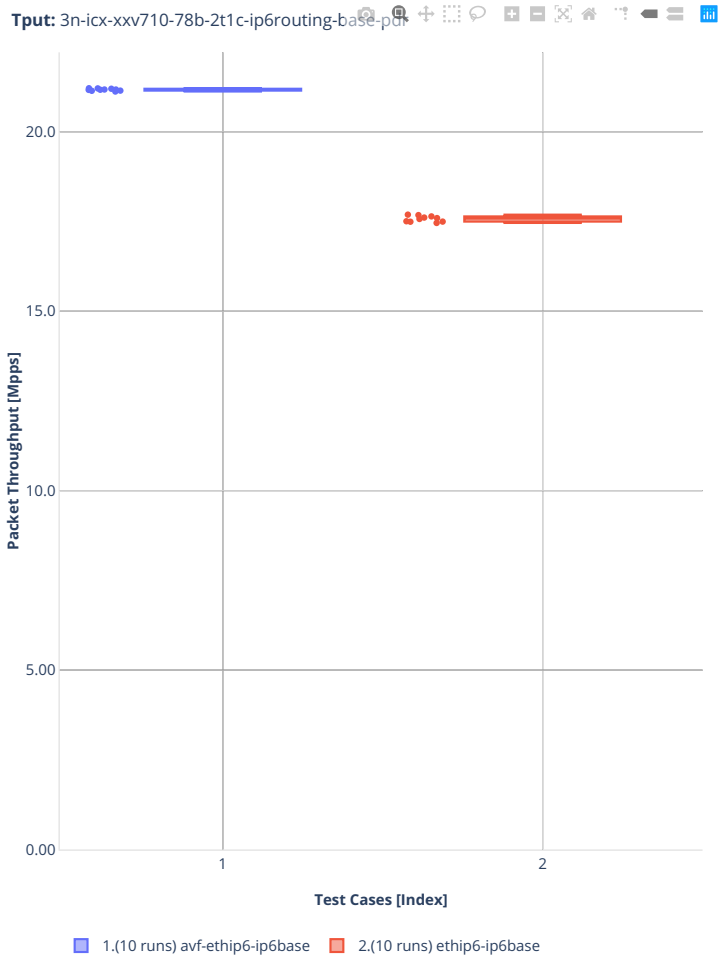



Tput: 2n-icx-xxv710-78b-2t1c-ip6routing-base-scale-af_xdp-pdr
Packet Throughput [Mpps]
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
Test Cases [Index]
1.(10 runs) af-xdp-ethip6-ip6base
2.(10 runs) af-xdp-ethip6-ip6scale20k
3.(10 runs) af-xdp-ethip6-ip6scale20k-rnd

3n-icx-xxv710

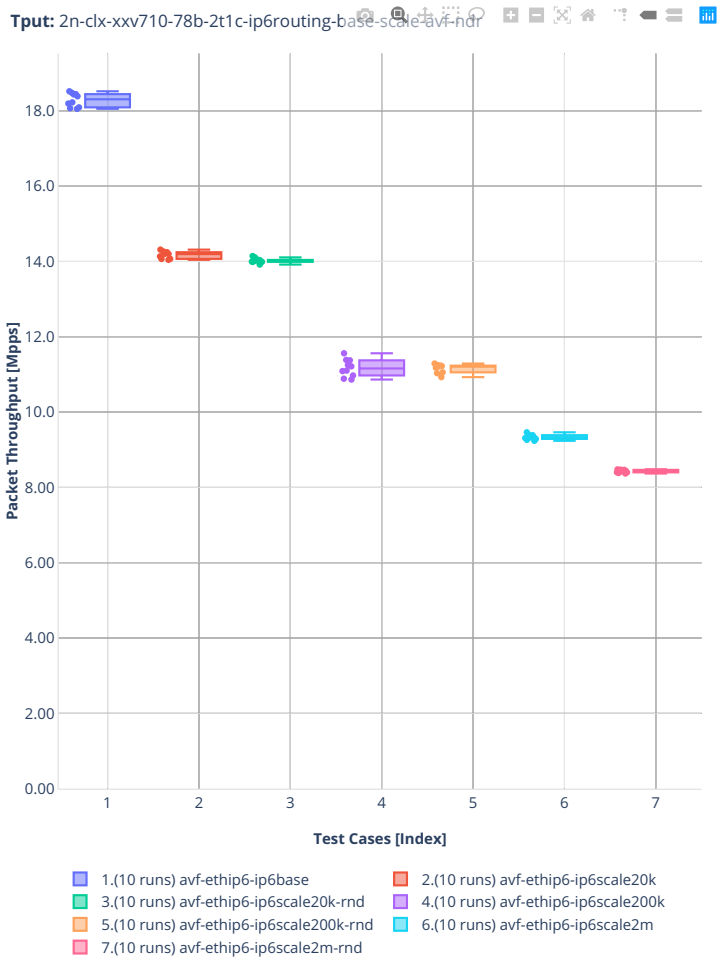
78b-2t1c-ip6routing-base

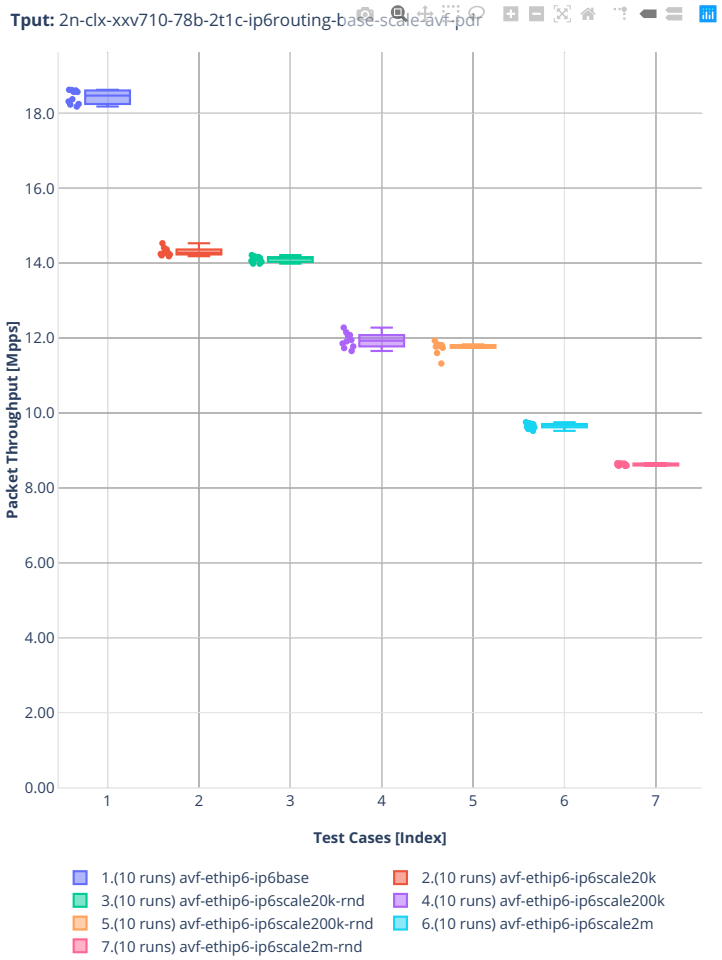



Tput: 3n-icx-xxv710-78b-2t1c-ip6routing-base-pdr
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) avf-ethip6-ip6base
2.(10 runs) ethip6-ip6base

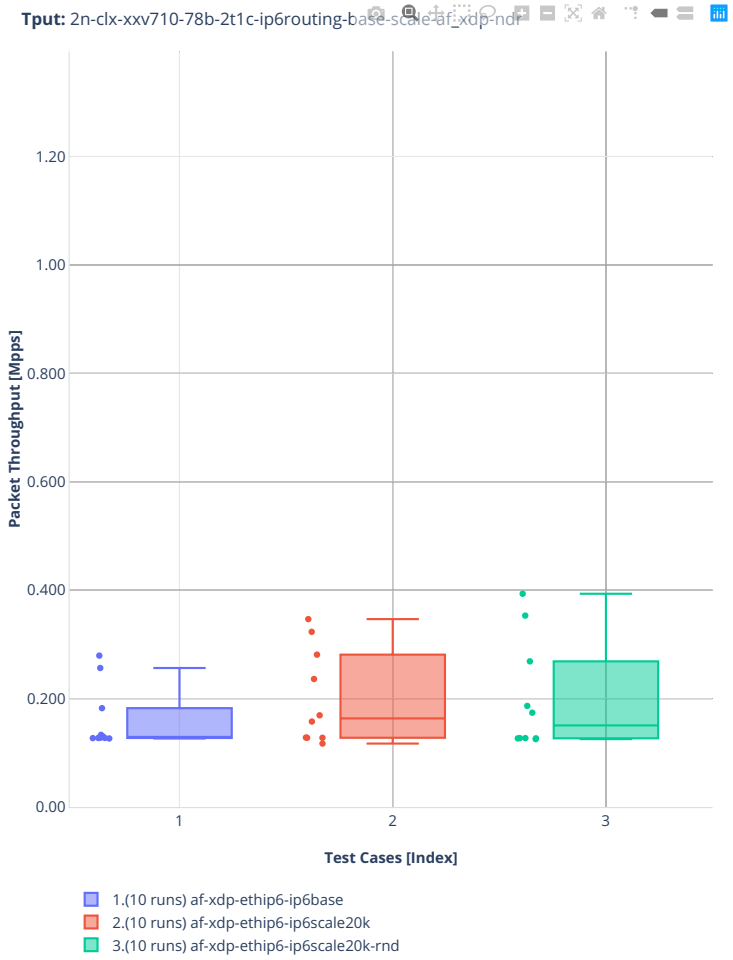
2n-clx-xxv710

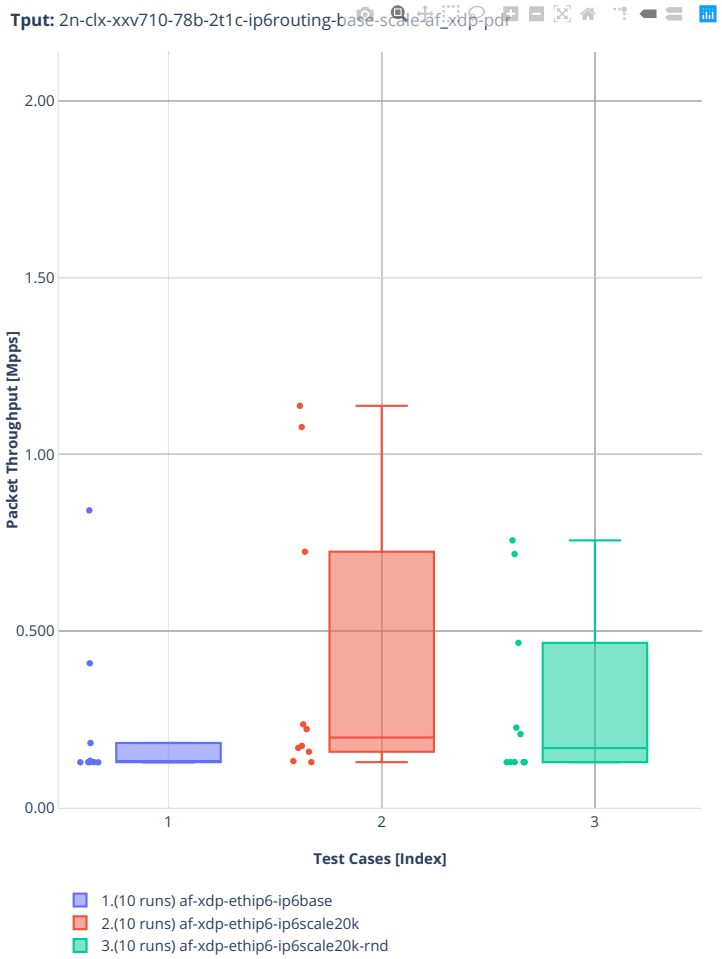
78b-2t1c-ip6routing-base-scale-avf



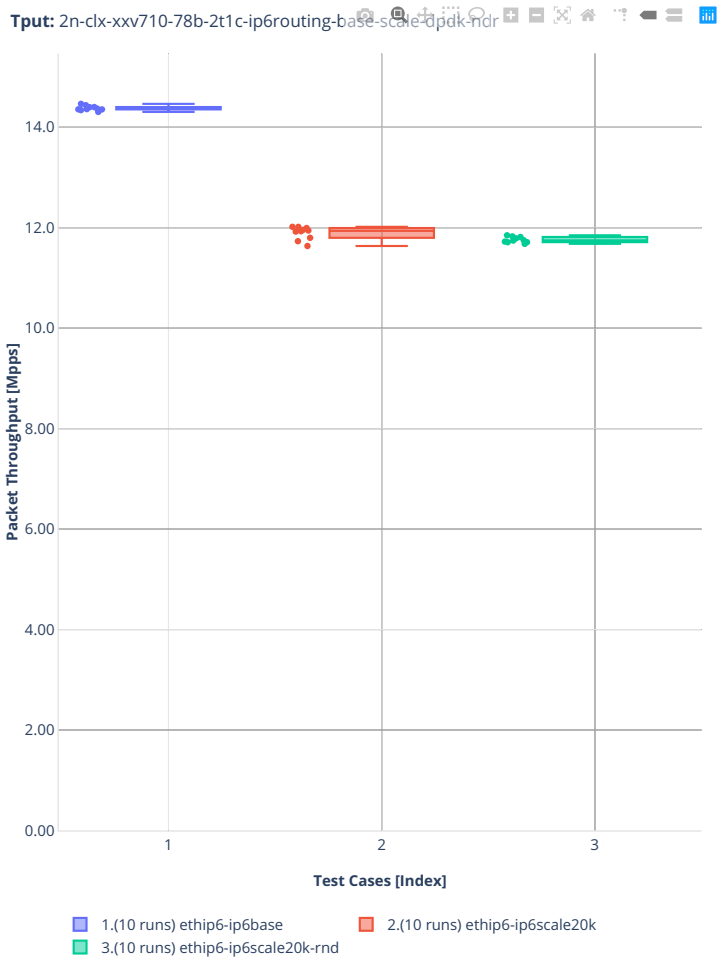

Tput: 2n-clx-xxv710-78b-2t1c-ip6routing-base-scale-avf-pdr
Packet Throughput [Mpps]
18.0
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
5
6
7
Test Cases [Index]
1.(10 runs) avf-ethip6-ip6base
2.(10 runs) avf-ethip6-ip6scale20k
3.(10 runs) avf-ethip6-ip6scale20k-rnd
4.(10 runs) avf-ethip6-ip6scale200k
5.(10 runs) avf-ethip6-ip6scale200k-rnd
6.(10 runs) avf-ethip6-ip6scale2m
7.(10 runs) avf-ethip6-ip6scale2m-rnd

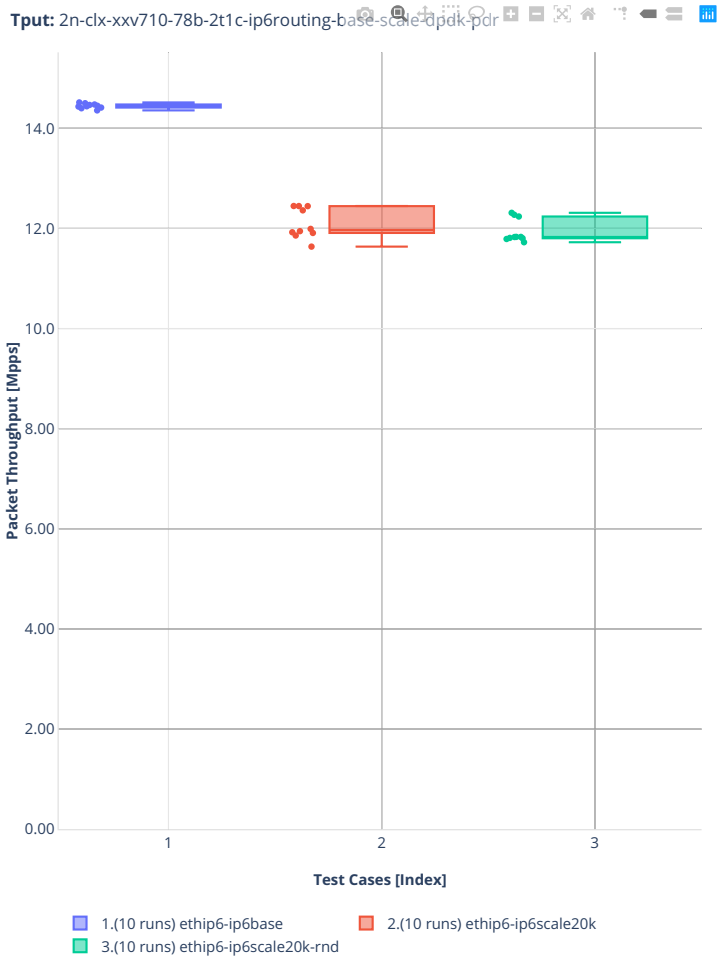
78b-2t1c-ip6routing-base-scale-af_xdp




Tput: 2n-clx-xxv710-78b-2t1c-ip6routing-base-scale-af_xdp-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) af-xdp-ethip6-ip6base
2.(10 runs) af-xdp-ethip6-ip6scale20k
3.(10 runs) af-xdp-ethip6-ip6scale20k-rnd

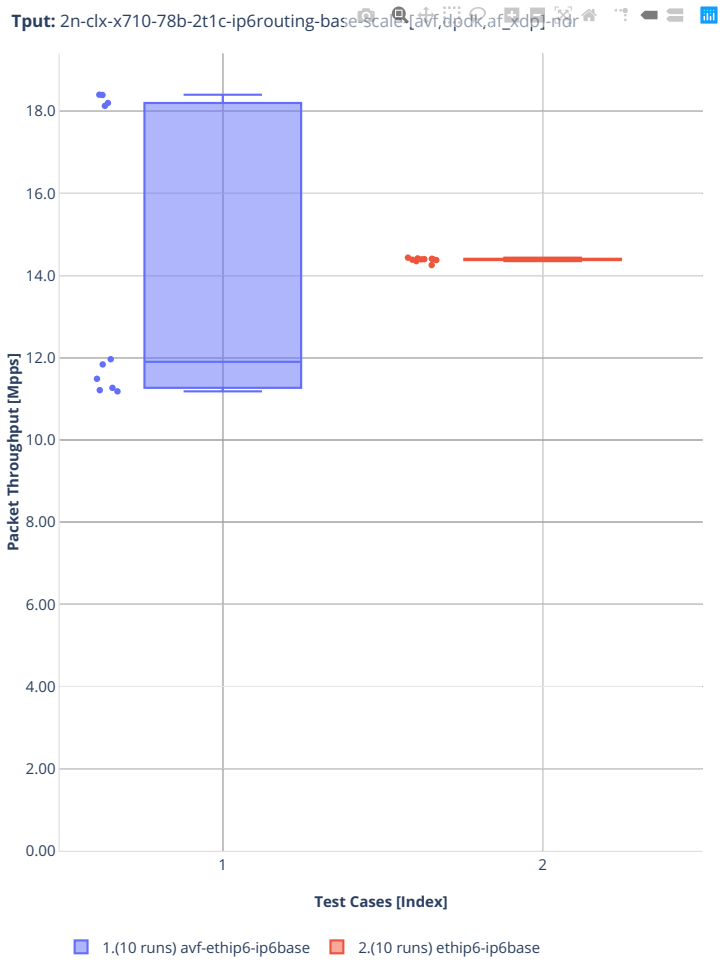
78b-2t1c-ip6routing-base-scale-dpdk

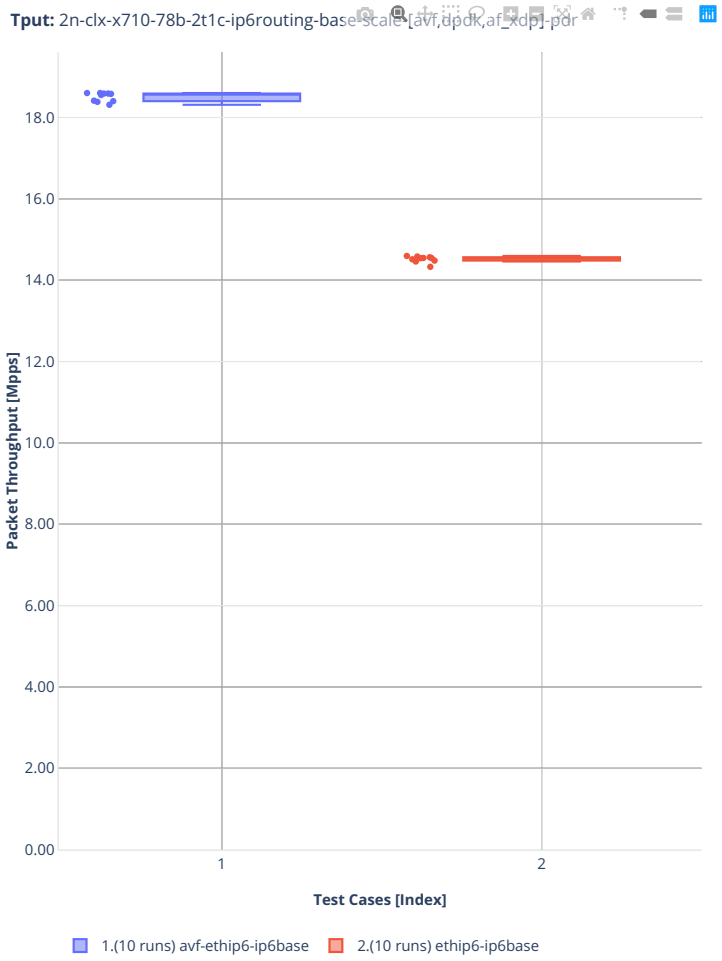



Tput: 2n-clx-xxv710-78b-2t1c-ip6routing-base-scale-dpdk-pdr
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip6-ip6base
2.(10 runs) ethip6-ip6scale20k
3.(10 runs) ethip6-ip6scale20k-rnd

2n-clx-x710

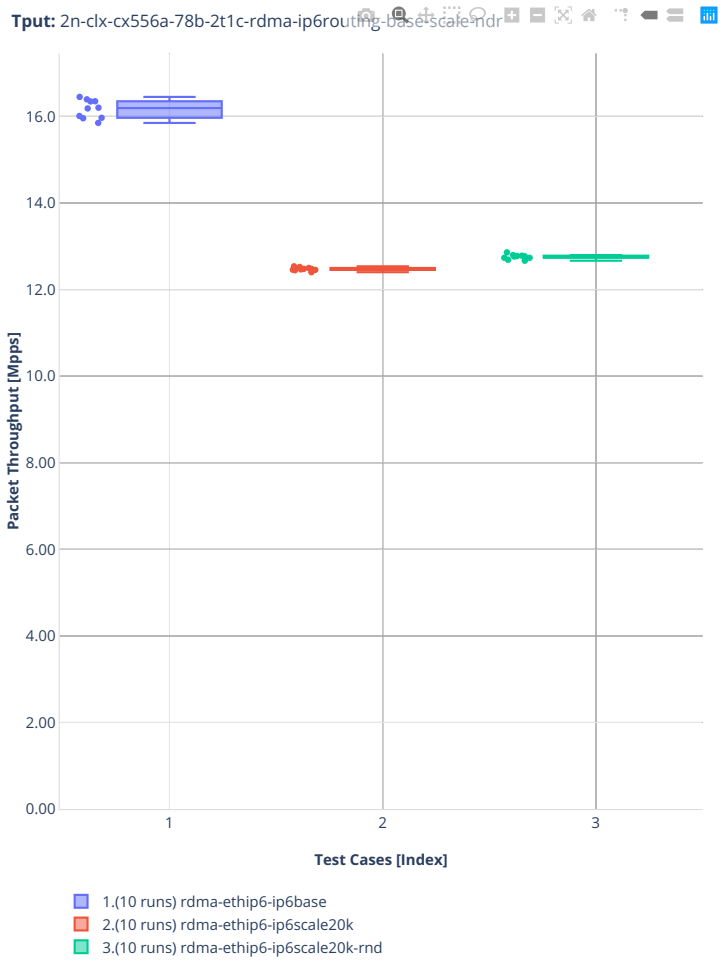
78b-2t1c-ip6routing-base-scale-[avf,dpdk]

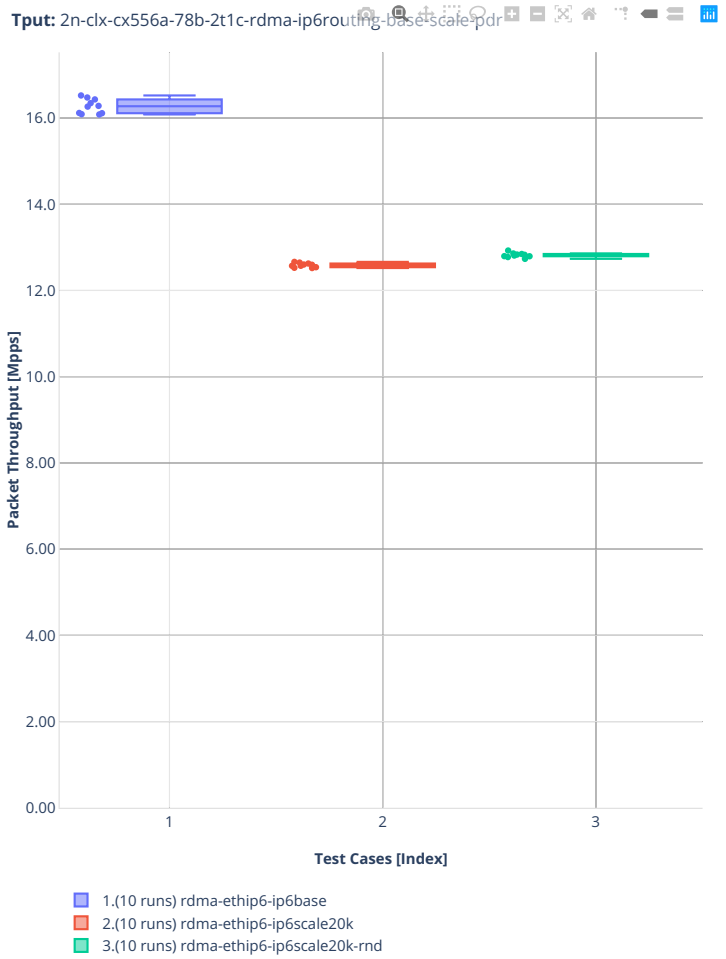



Tput: 2n-clx-x710-78b-2t1c-ip6routing-base-scale-[avf,dpdk,af_xdp]-pdr
18.0
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) avf-ethip6-ip6base
2.(10 runs) ethip6-ip6base

2n-clx-cx556a

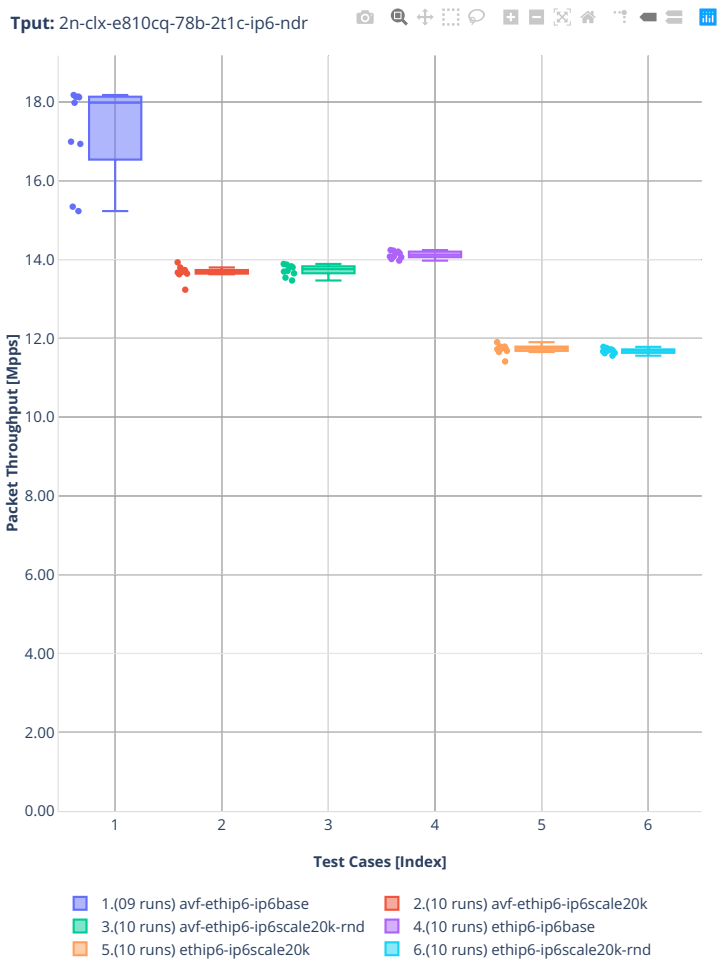
78b-2t1c-ip6routing-base-scale-rdma-core




Tput: 2n-clx-cx556a-78b-2t1c-rdma-ip6routing-base-scale-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) rdma-ethip6-ip6base
2.(10 runs) rdma-ethip6-ip6scale20k
3.(10 runs) rdma-ethip6-ip6scale20k-rnd

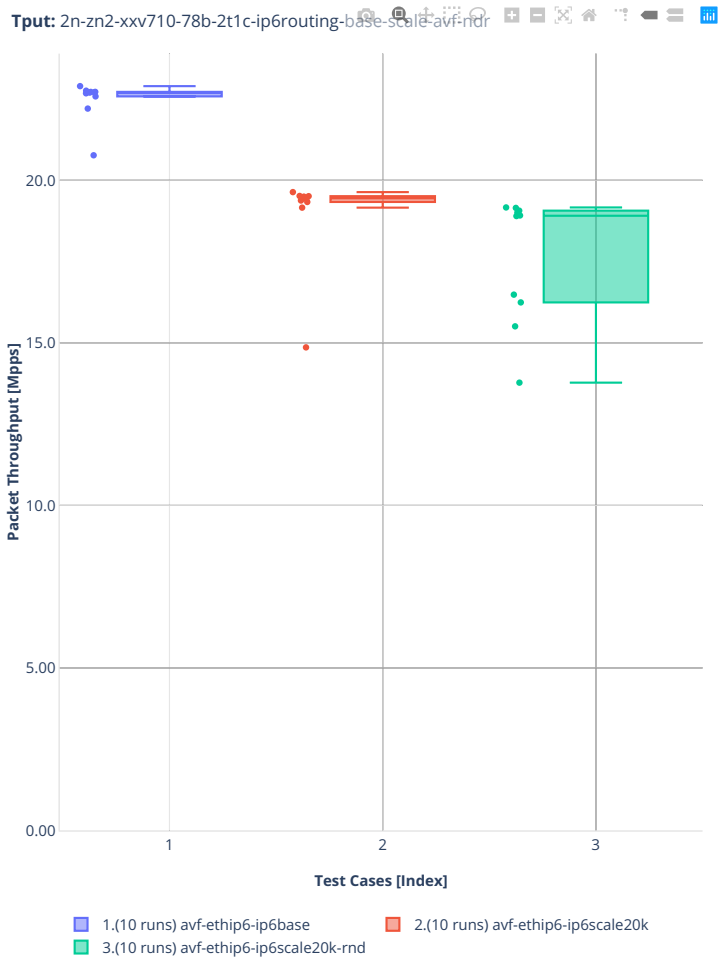
2n-clx-e810cq

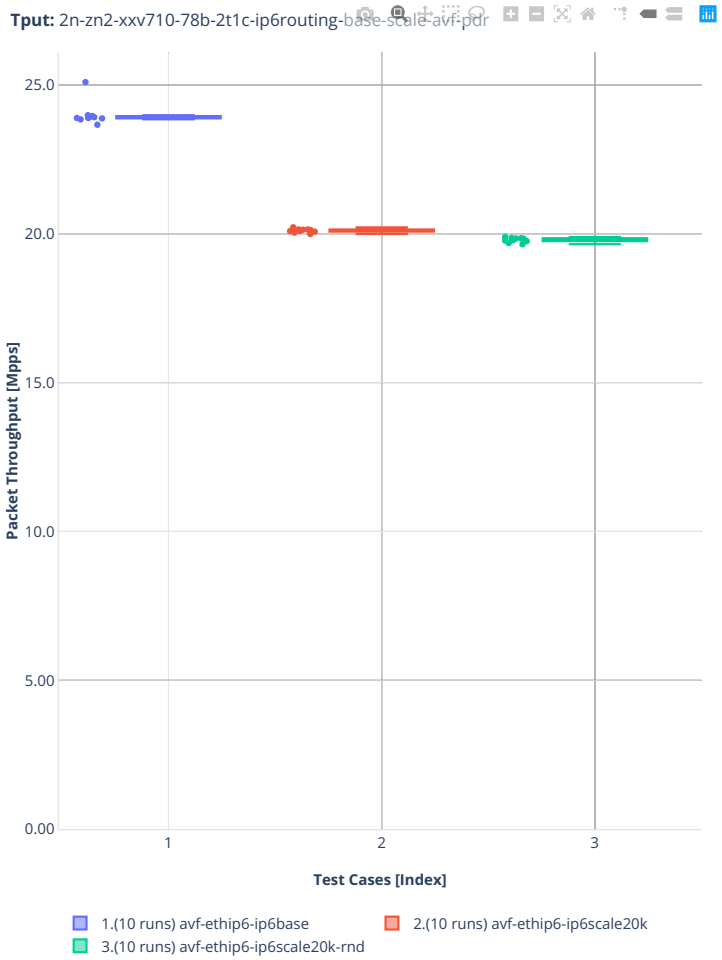
78b-2t1c-ip6routing-base-scale



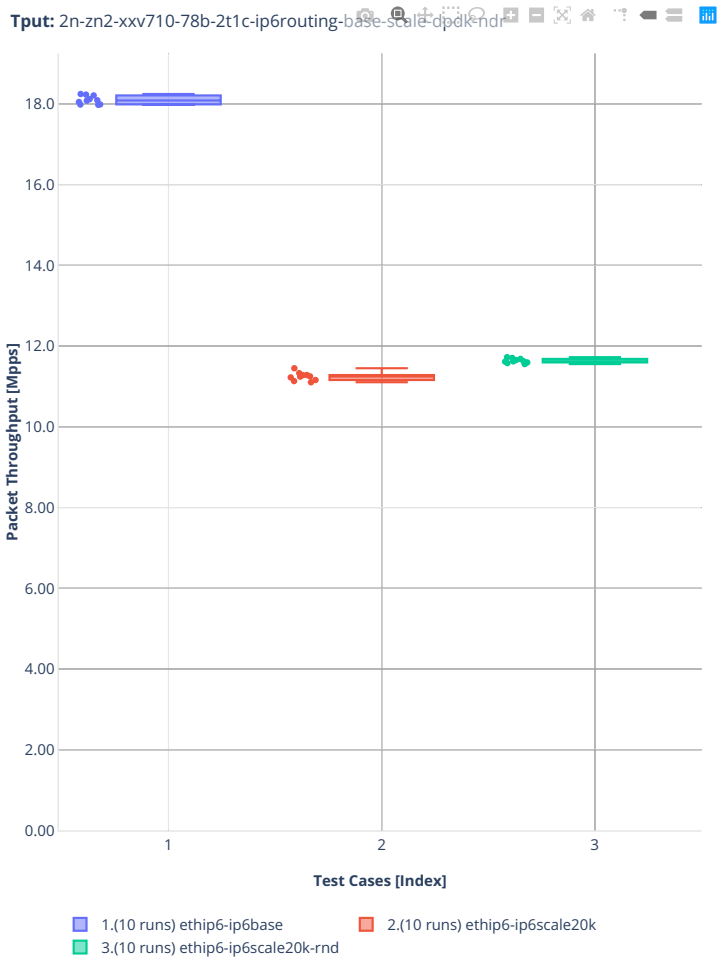
2n-zn2-xxv710

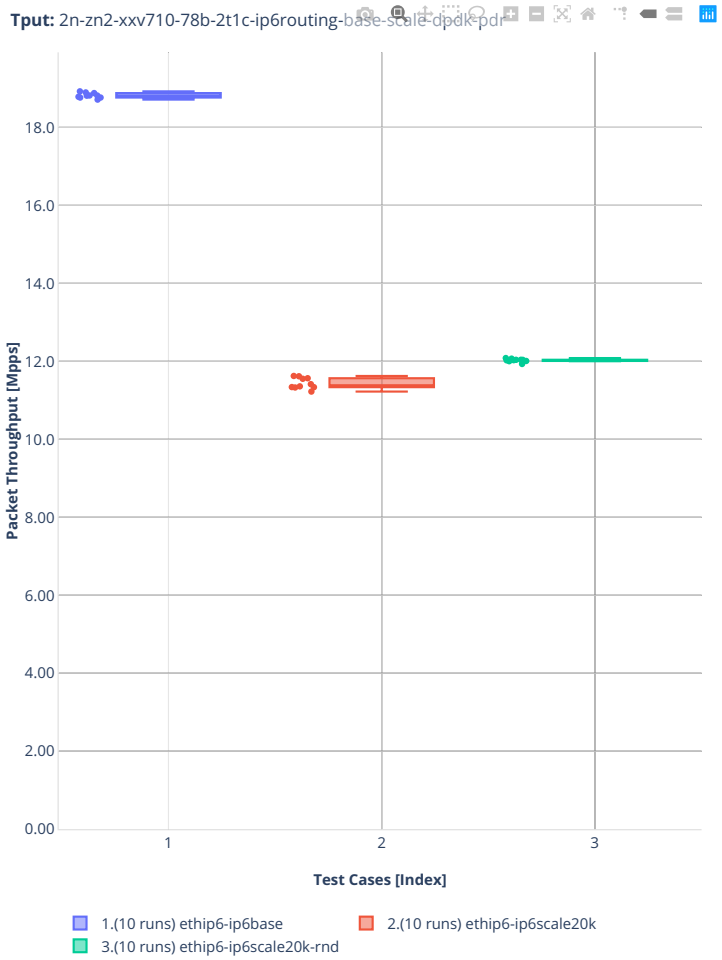
78b-2t1c-ip6routing-base-scale-avf




Tput: 2n-zn2-xxv710-78b-2t1c-ip6routing-base-scale-avf-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-ethip6-ip6base
2.(10 runs) avf-ethip6-ip6scale20k
3.(10 runs) avf-ethip6-ip6scale20k-rnd

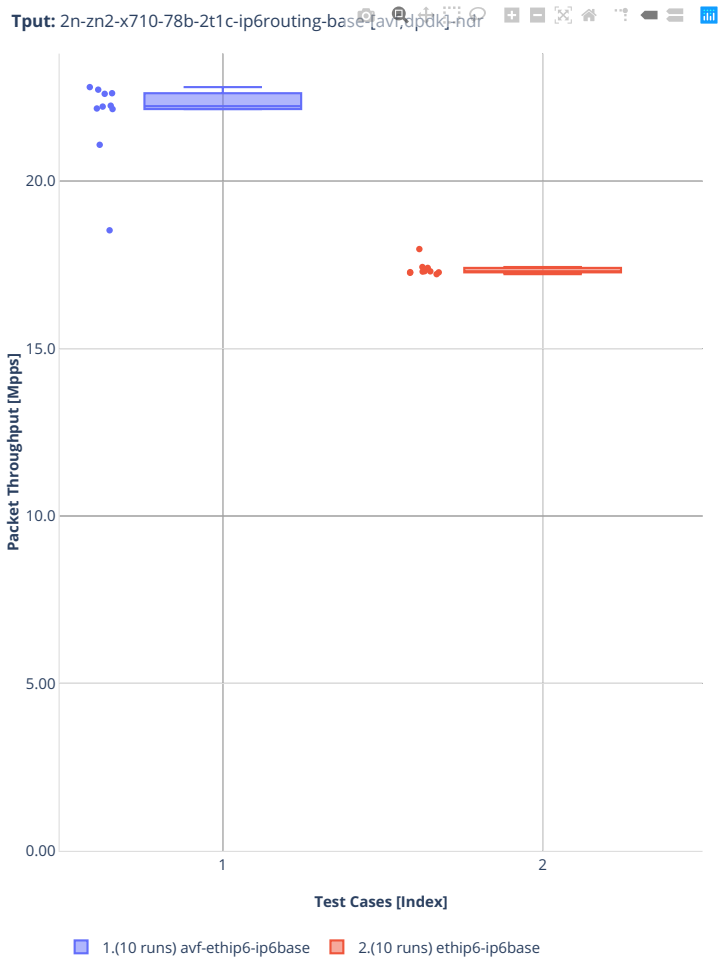
78b-2t1c-ip6routing-base-scale-dpdk

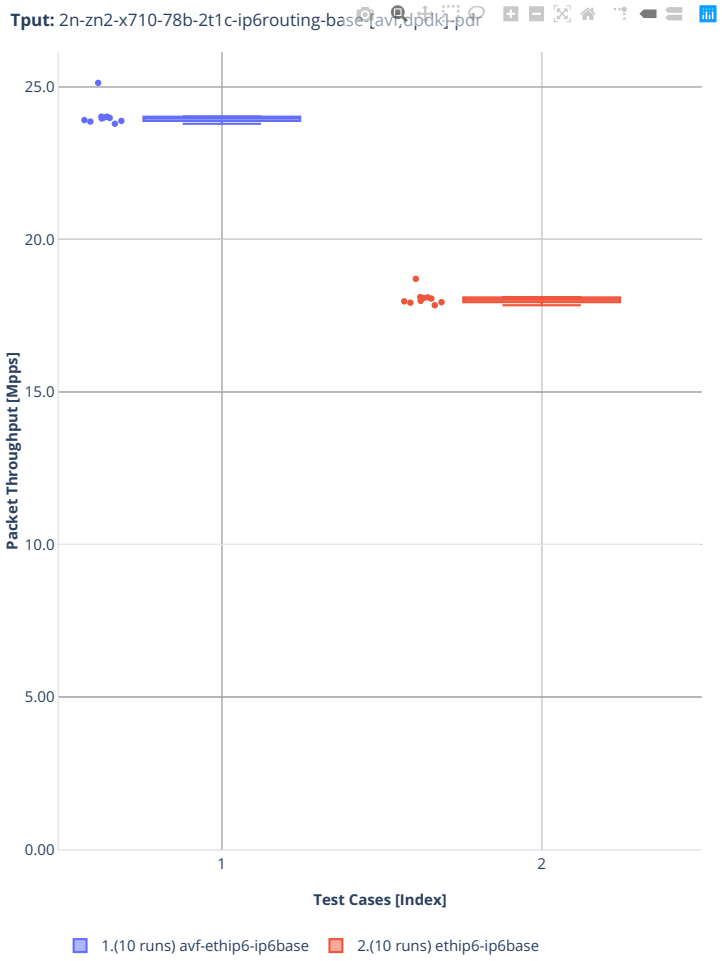



Tput: 2n-zn2-xxv710-78b-2t1c-ip6routing-base-scale-dpdk-pdr
18.0
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip6-ip6base
2.(10 runs) ethip6-ip6scale20k
3.(10 runs) ethip6-ip6scale20k-rnd

2n-zn2-x710

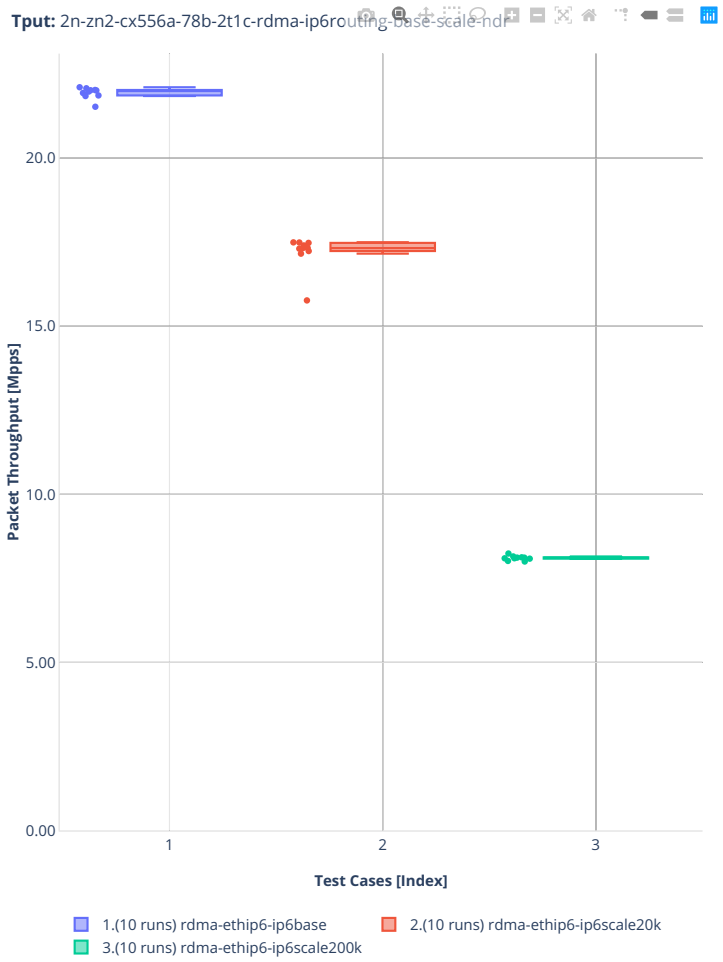
78b-2t1c-ip6routing-base-[avf,dpdk]

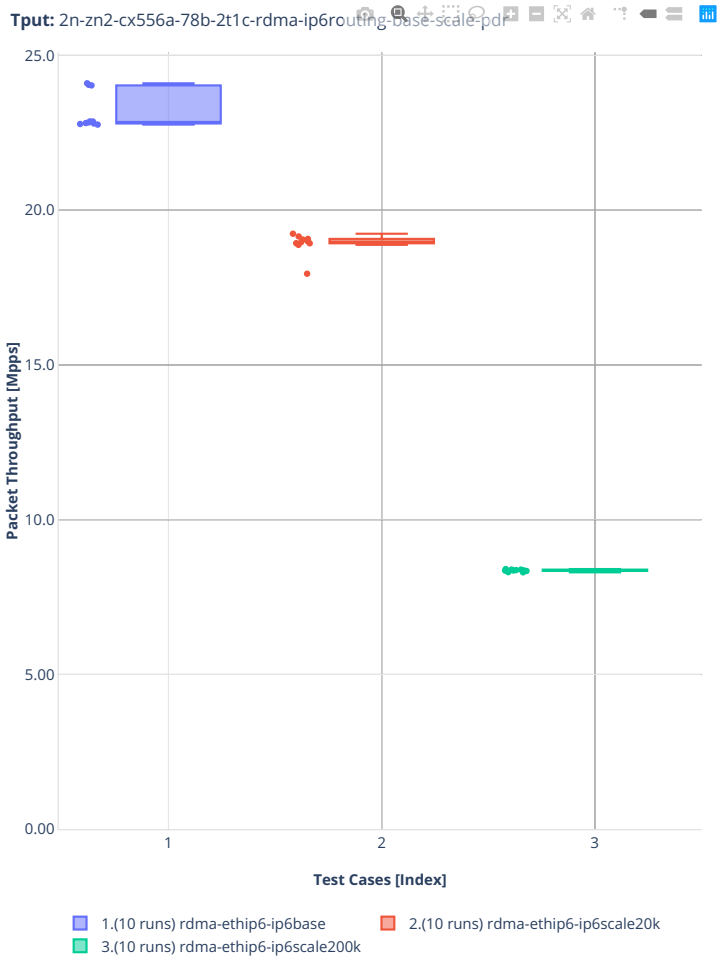



Tput: 2n-zn2-x710-78b-2t1c-ip6routing-base-[avf,dpdk]-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) avf-ethip6-ip6base
2.(10 runs) ethip6-ip6base

2n-zn2-cx556a

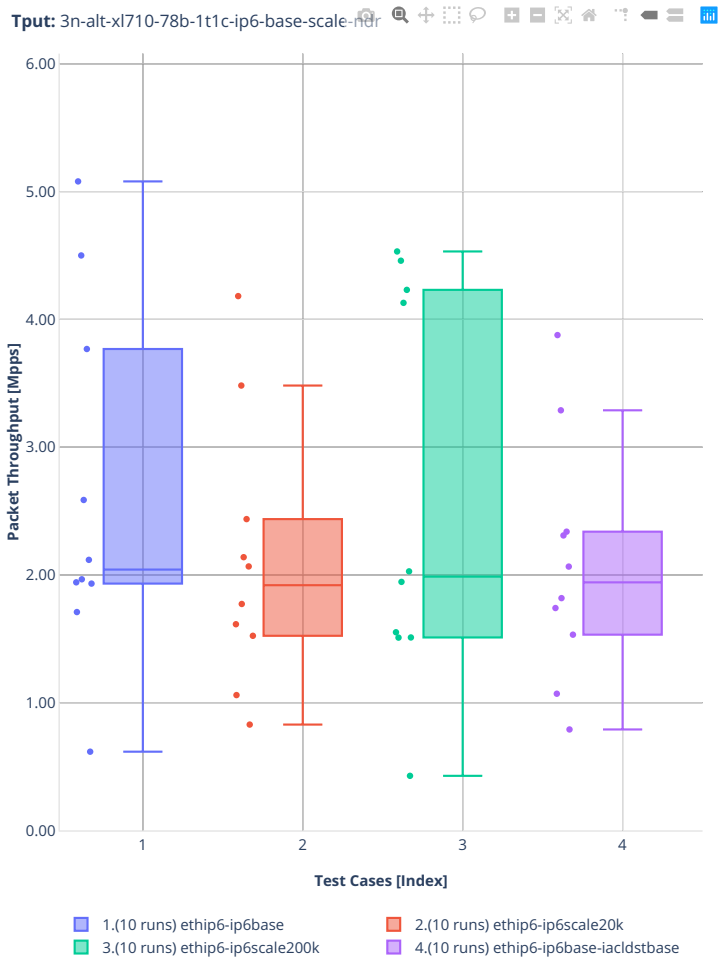
78b-2t1c-ip6routing-base-scale-rdma-core

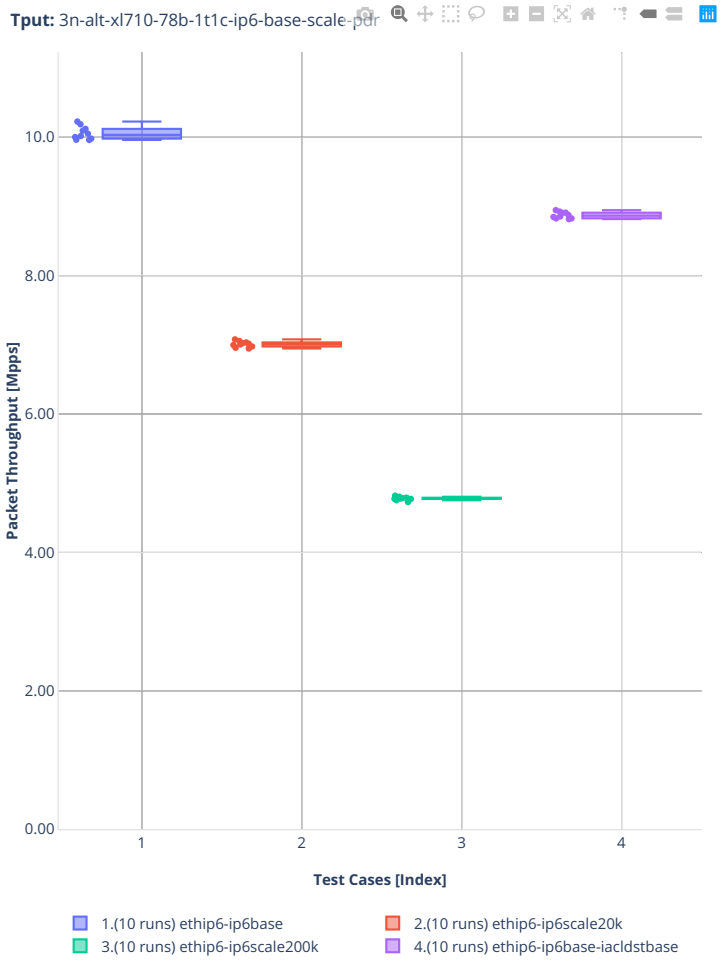



Tput: 2n-zn2-cx556a-78b-2t1c-rdma-ip6routing-base-scale-pdr
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) rdma-ethip6-ip6base
2.(10 runs) rdma-ethip6-ip6scale20k
3.(10 runs) rdma-ethip6-ip6scale200k

3n-alt-xl710

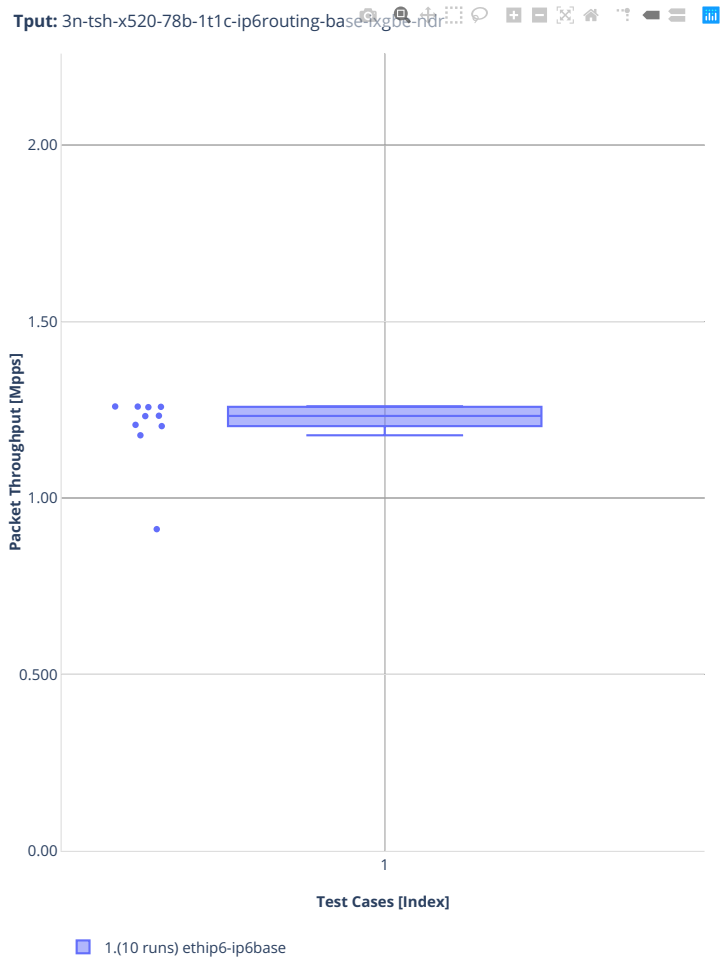
78b-1t1c-ip6routing-base-scale

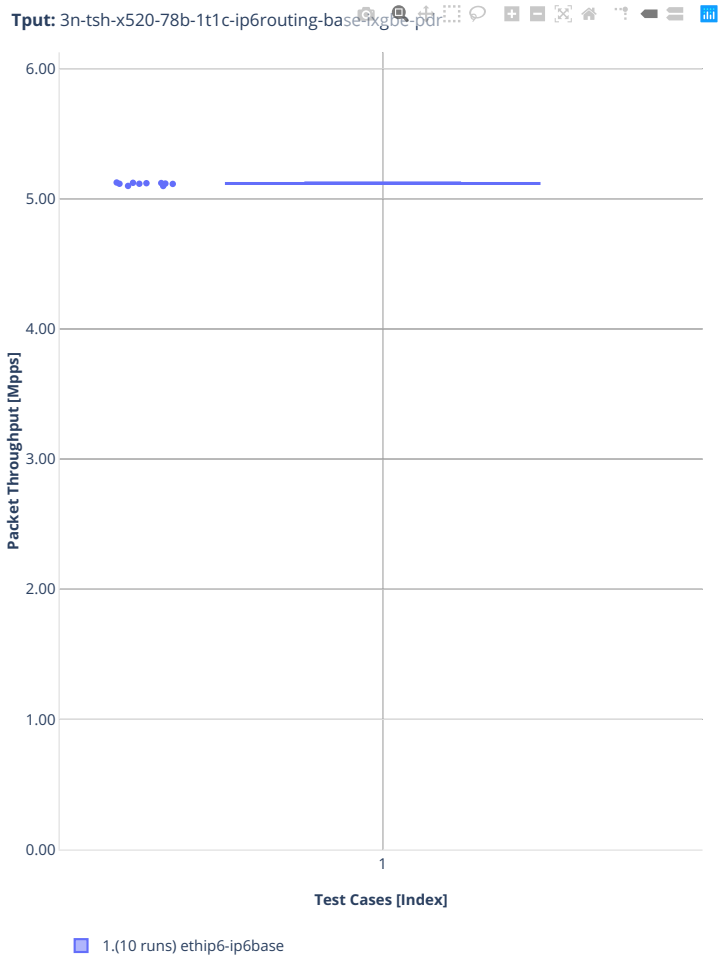



Tput: 3n-alt-xl710-78b-1t1c-ip6-base-scale-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) ethip6-ip6base
2.(10 runs) ethip6-ip6scale20k
3.(10 runs) ethip6-ip6scale200k
4.(10 runs) ethip6-ip6base-iacldstbase

3n-tsh-x520

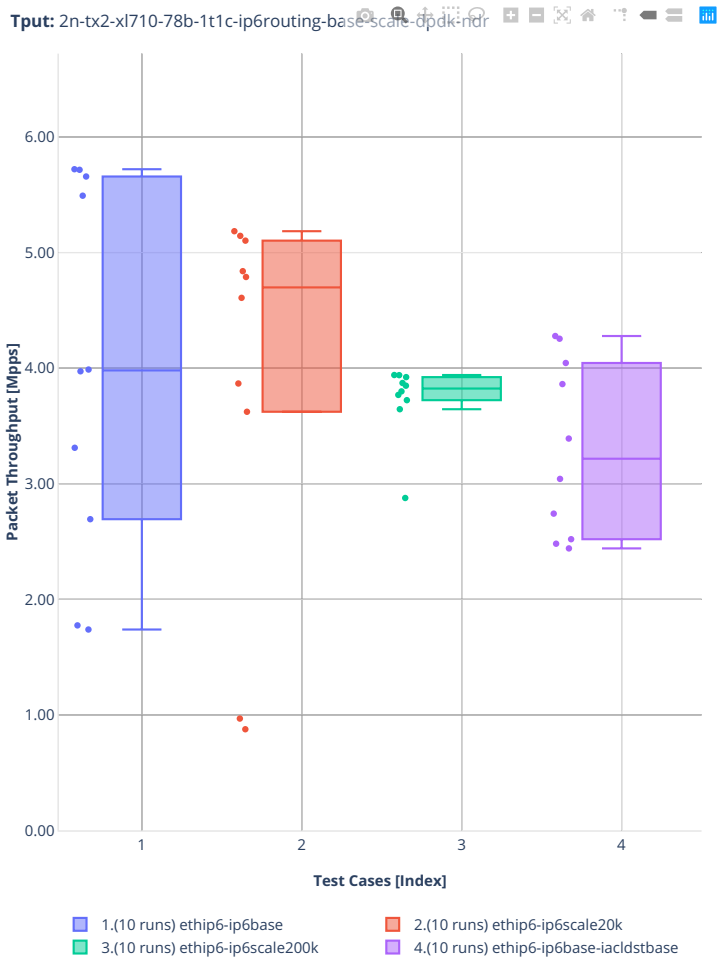
78b-1t1c-ip6routing-base-ixgbe

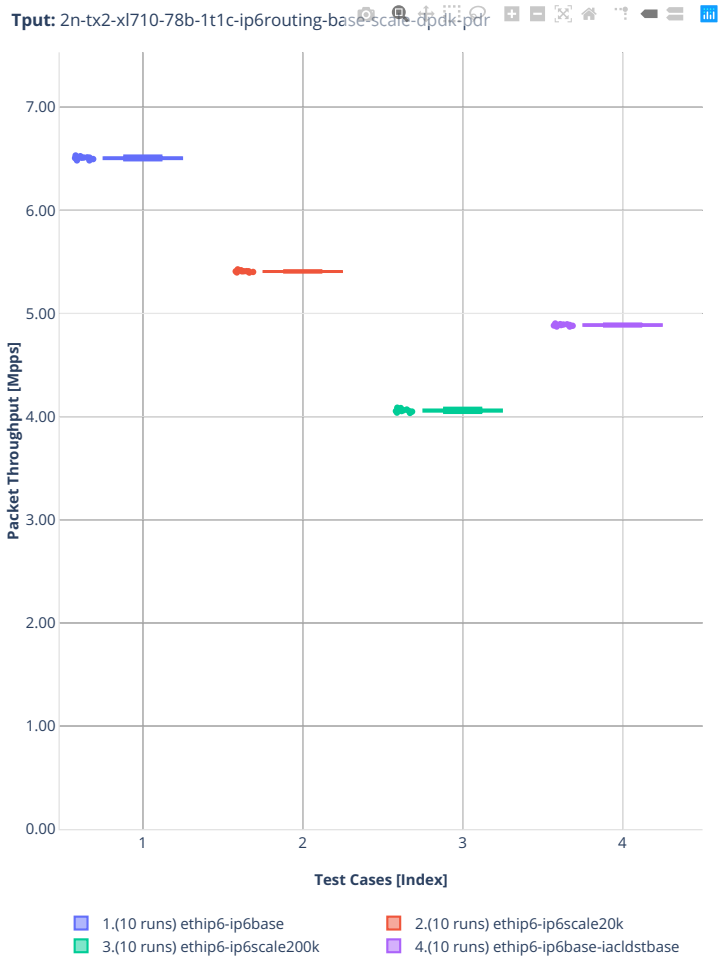



Tput: 3n-tsh-x520-78b-1t1c-ip6routing-base-ixgbe-pdr
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
1.(10 runs) ethip6-ip6base

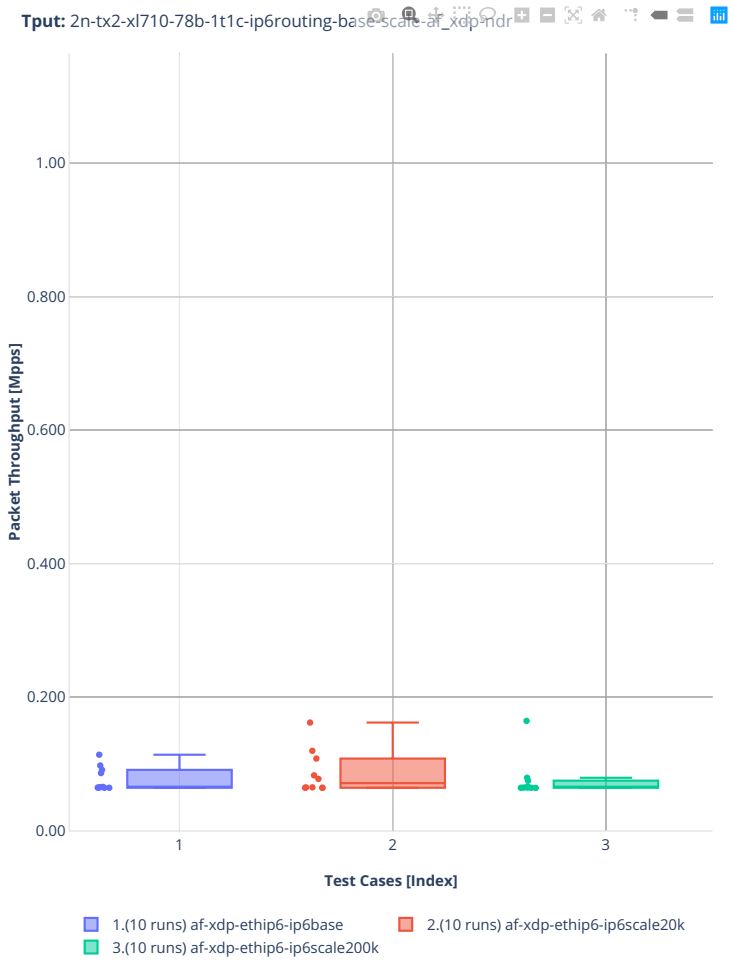
2n-tx2-xl710

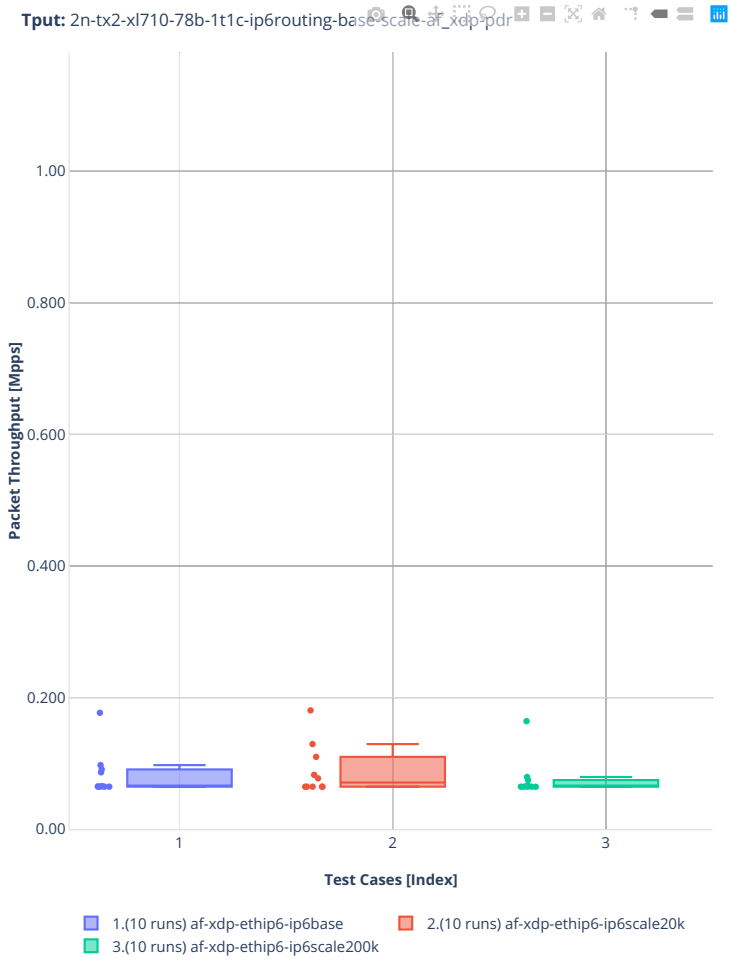
78b-1t1c-ip6routing-base-scale-dpdk




Tput: 2n-tx2-xl710-78b-1t1c-ip6routing-base-scale-dpdk-pdr
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) ethip6-ip6base
2.(10 runs) ethip6-ip6scale20k
3.(10 runs) ethip6-ip6scale200k
4.(10 runs) ethip6-ip6base-iacldstbase

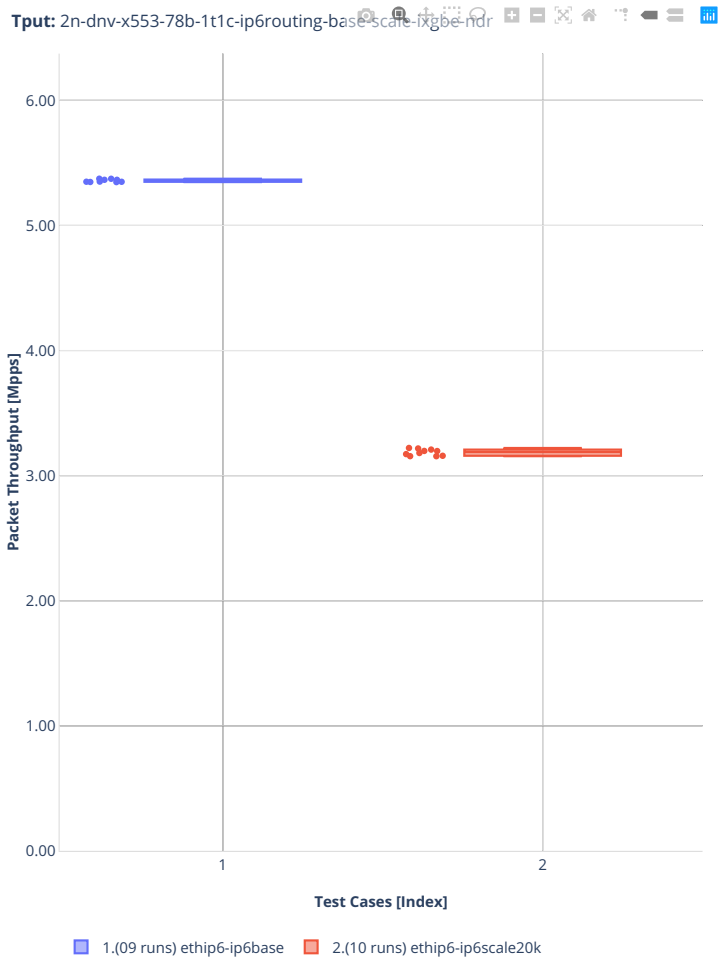
78b-1t1c-ip6routing-base-scale-af-xdp

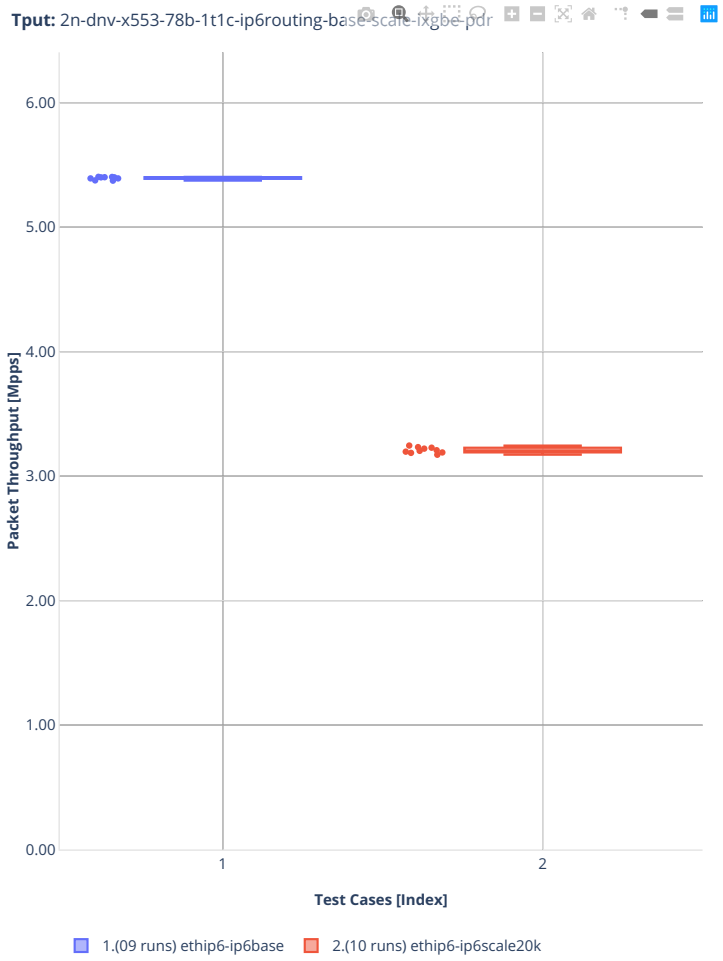



Tput: 2n-tx2-xl710-78b-1t1c-ip6routing-base-scale-af_xdp-pdr
1.00
0.800
0.600
0.400
0.200
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) af-xdp-ethip6-ip6base
2.(10 runs) af-xdp-ethip6-ip6scale20k
3.(10 runs) af-xdp-ethip6-ip6scale200k

2n-dnv-x553

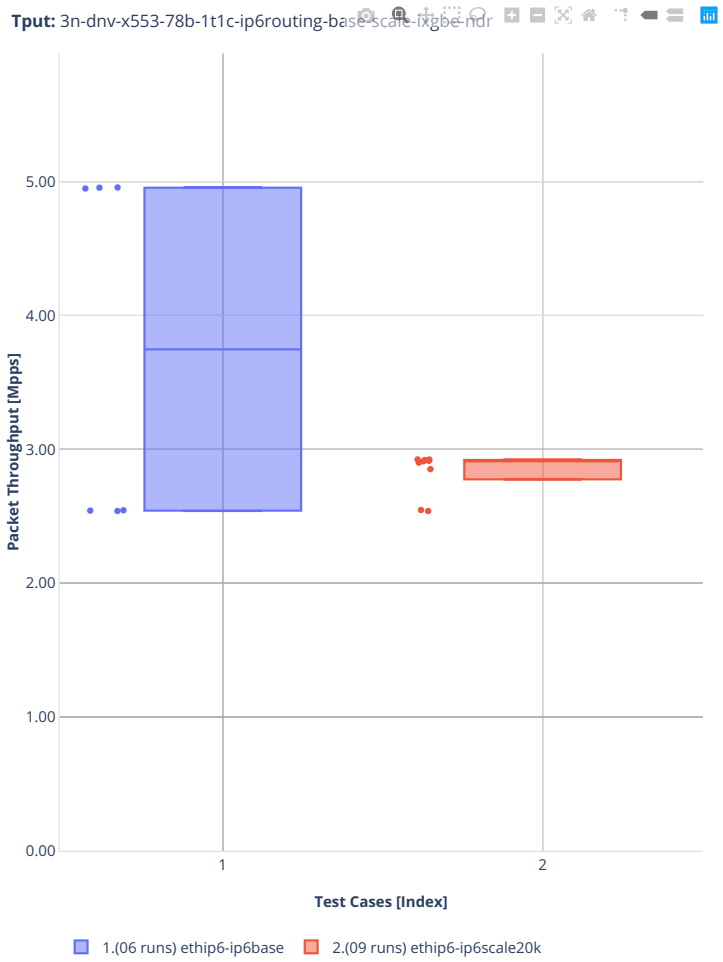
78b-1t1c-ip6routing-base-scale-ixgbe

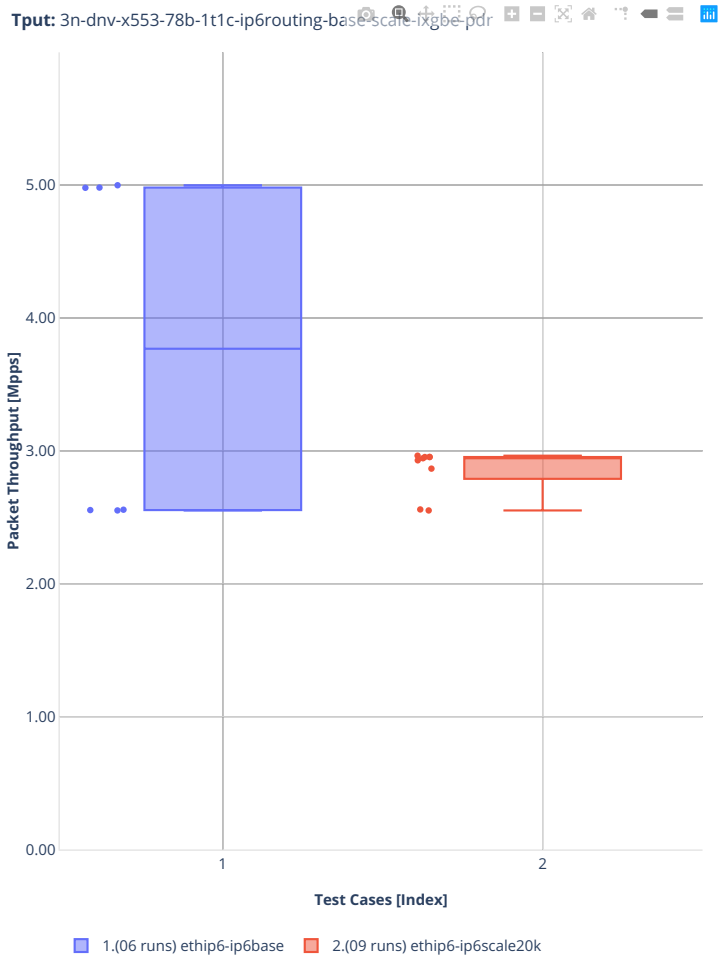



Tput: 2n-dnv-x553-78b-1t1c-ip6routing-base-scale-ixgbe-pdr
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(09 runs) ethip6-ip6base
2.(10 runs) ethip6-ip6scale20k

3n-dnv-x553

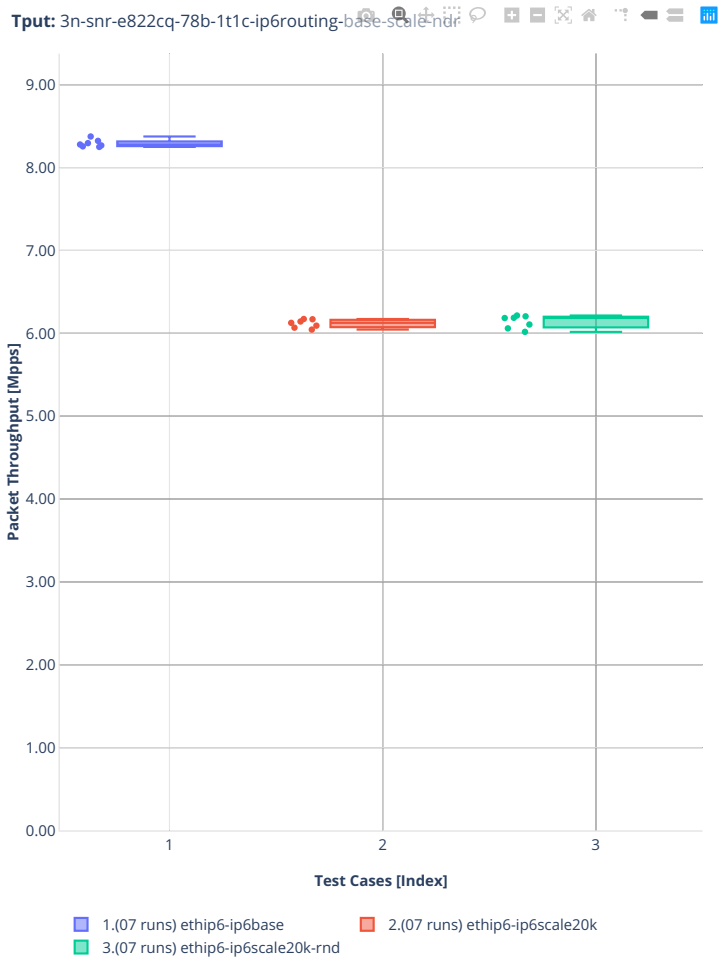
78b-1t1c-ip6routing-base-scale-ixgbe

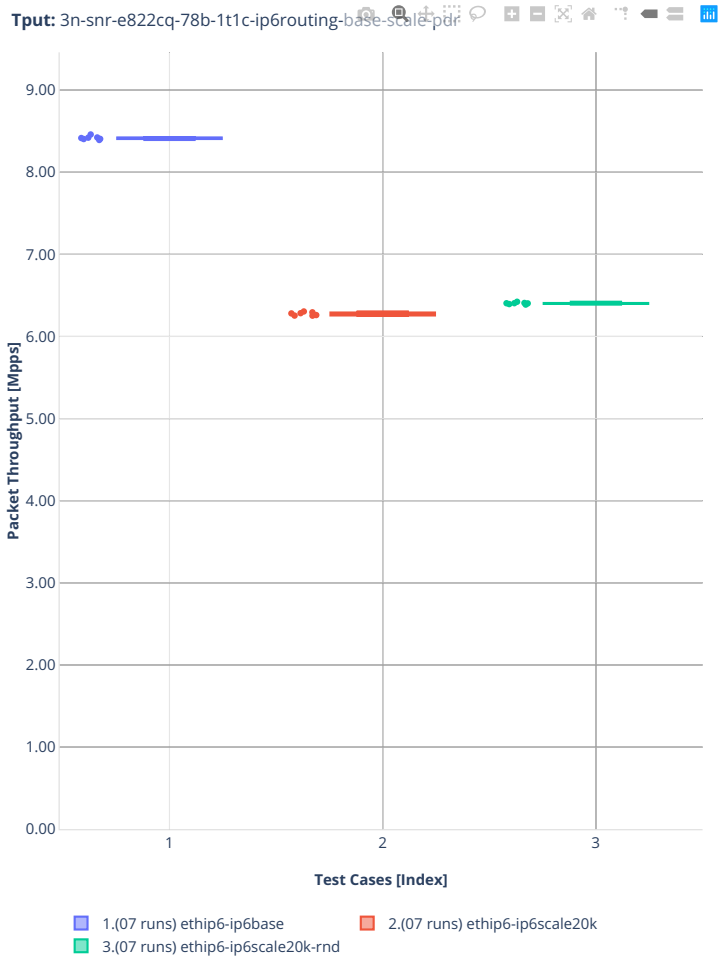



Tput: 3n-dnv-x553-78b-1t1c-ip6routing-base-scale-ixgbe-pdr
Packet Throughput [Mpps]
5.00
4.00
3.00
2.00
1.00
0.00
1
2
Test Cases [Index]
1.(06 runs) ethip6-ip6base
2.(09 runs) ethip6-ip6scale20k

3n-snr-e822cq

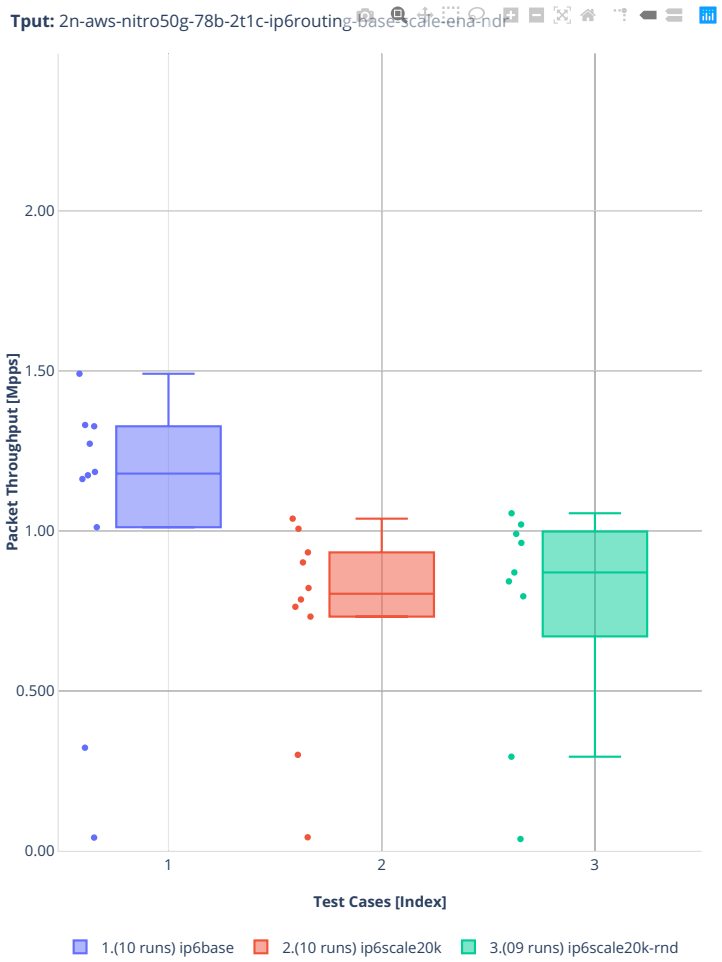
78b-1t1c-ip6routing-base-scale

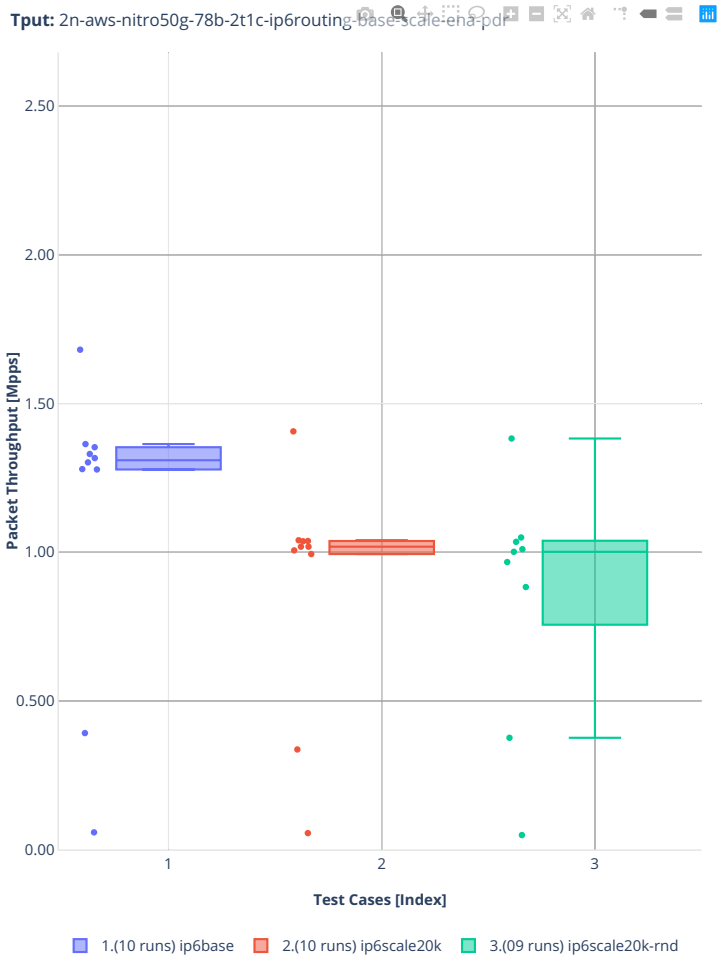



Tput: 3n-snr-e822cq-78b-1t1c-ip6routing-base-scale-pdr
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(07 runs) ethip6-ip6base
2.(07 runs) ethip6-ip6scale20k
3.(07 runs) ethip6-ip6scale20k-rnd

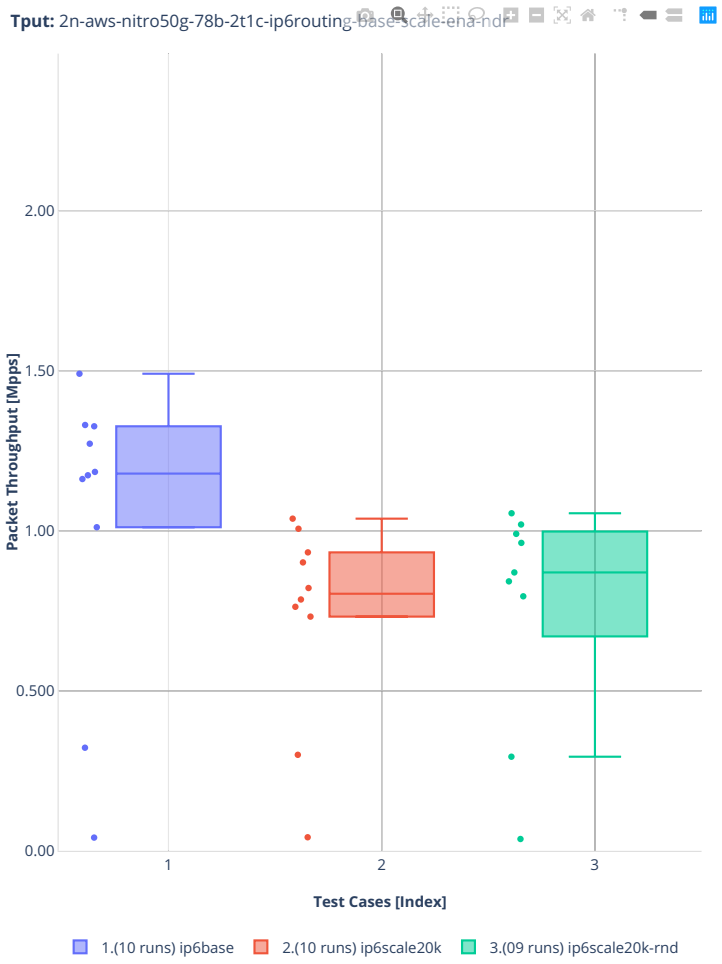
2n-aws-nitro50g

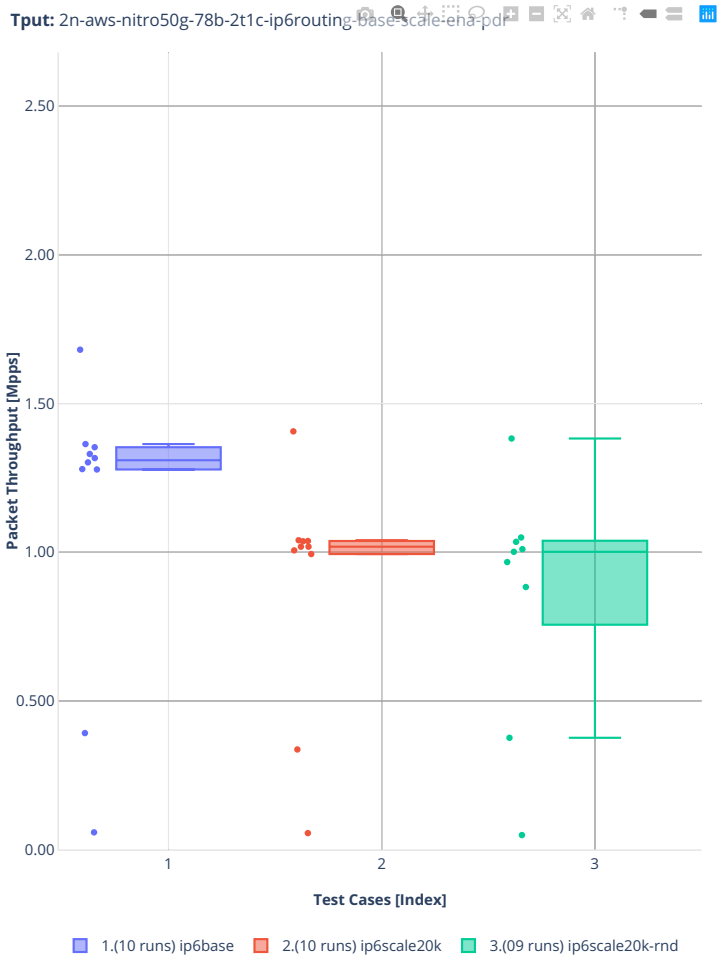
78b-2t1c-ip6routing-base-scale-ena



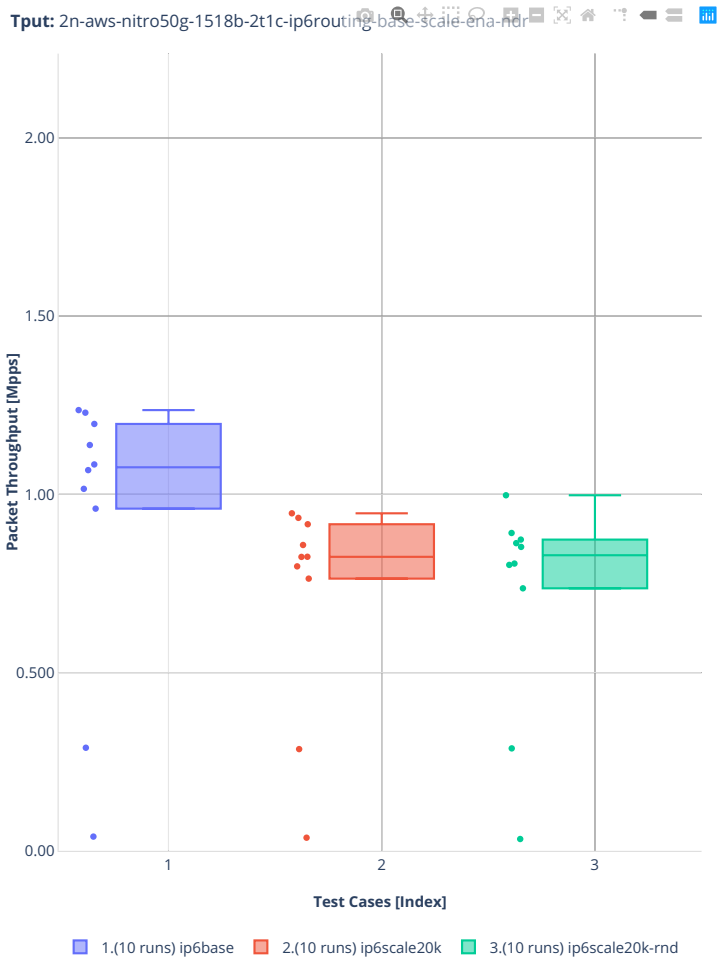

Tput: 2n-aws-nitro50g-78b-2t1c-ip6routing-base-scale-ena-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip6base
2.(10 runs) ip6scale20k
3.(09 runs) ip6scale20k-rnd

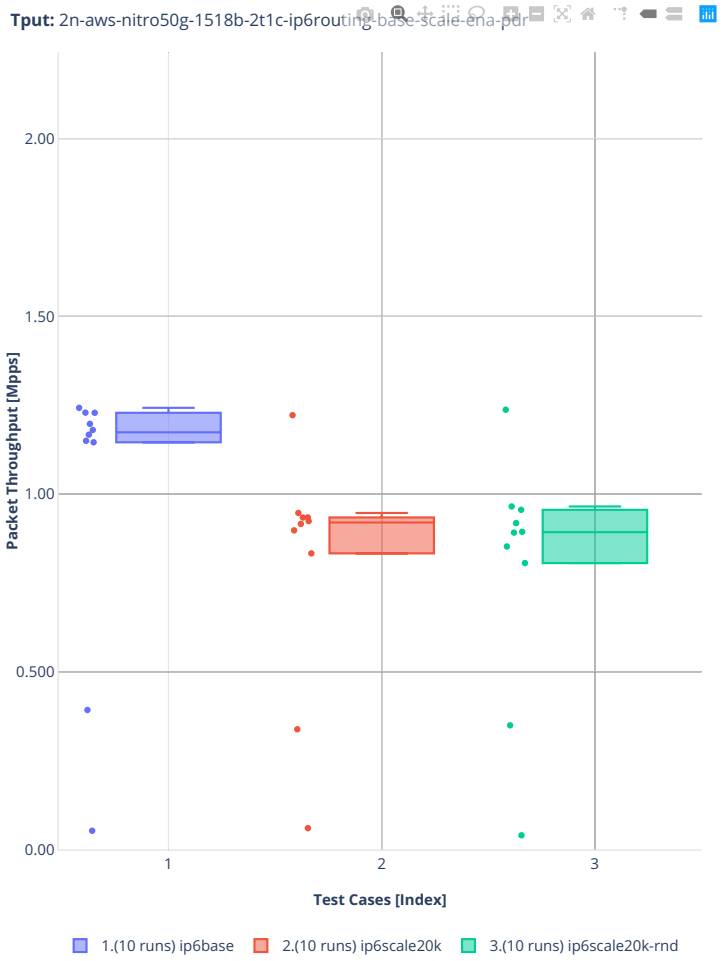
78b-4t2c-ip6routing-base-scale-ena



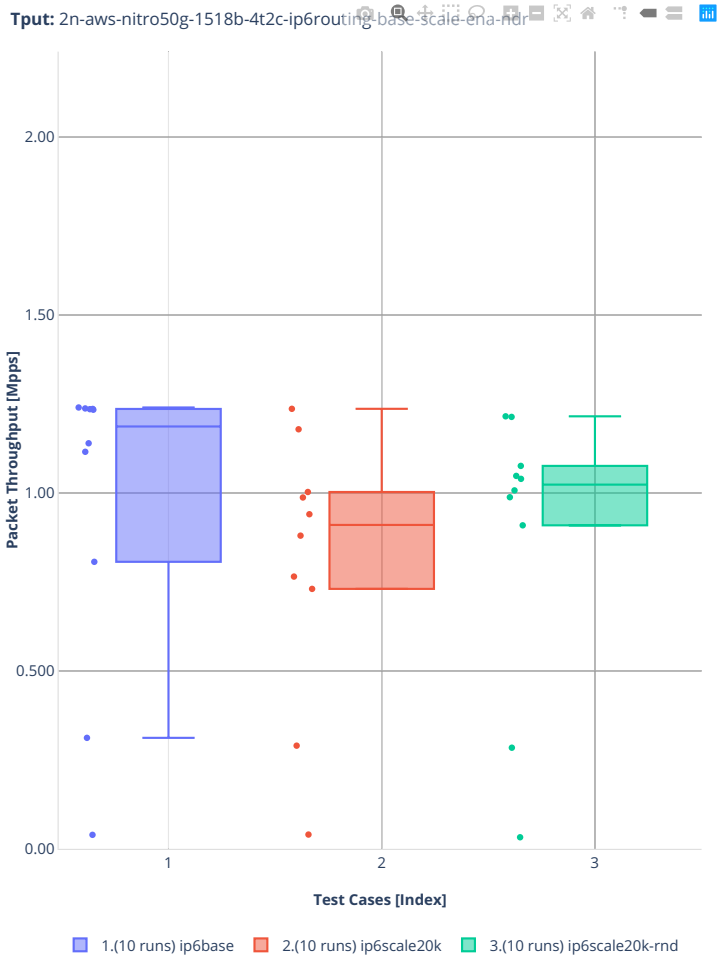

Tput: 2n-aws-nitro50g-78b-2t1c-ip6routing-base-scale-ena-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip6base
2.(10 runs) ip6scale20k
3.(09 runs) ip6scale20k-rnd

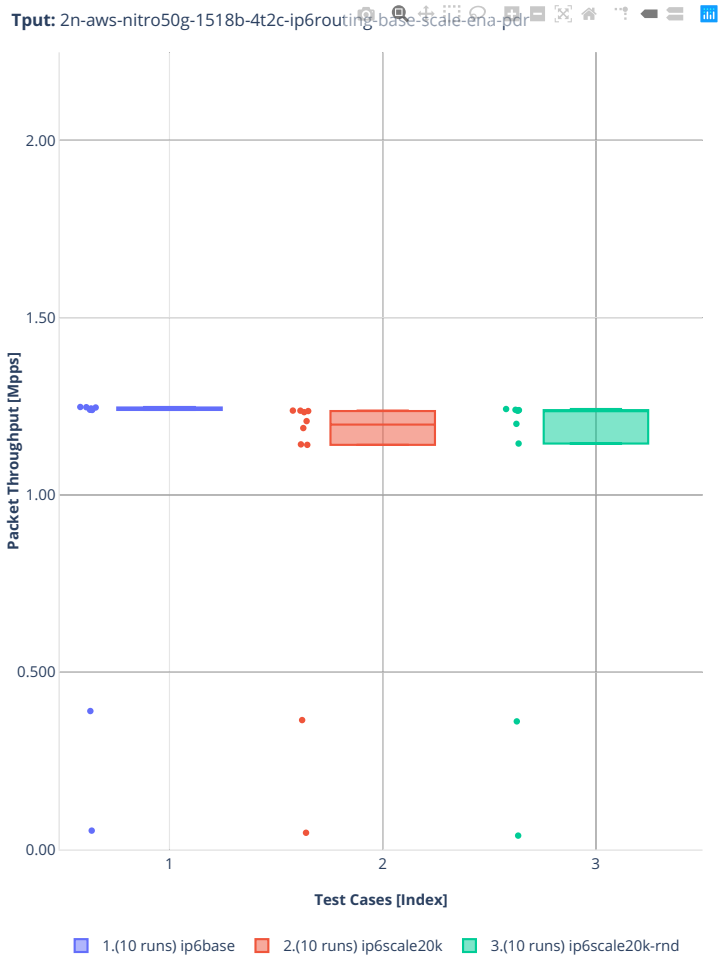
1518b-2t1c-ip6routing-base-scale-ena




Tput: 2n-aws-nitro50g-1518b-2t1c-ip6routing-base-scale-ena-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip6base
2.(10 runs) ip6scale20k
3.(10 runs) ip6scale20k-rnd

1518b-4t2c-ip6routing-base-scale-ena




Tput: 2n-aws-nitro50g-1518b-4t2c-ip6routing-base-scale-ena-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ip6base
2.(10 runs) ip6scale20k
3.(10 runs) ip6scale20k-rnd

2.3.4 SRv6 Routing

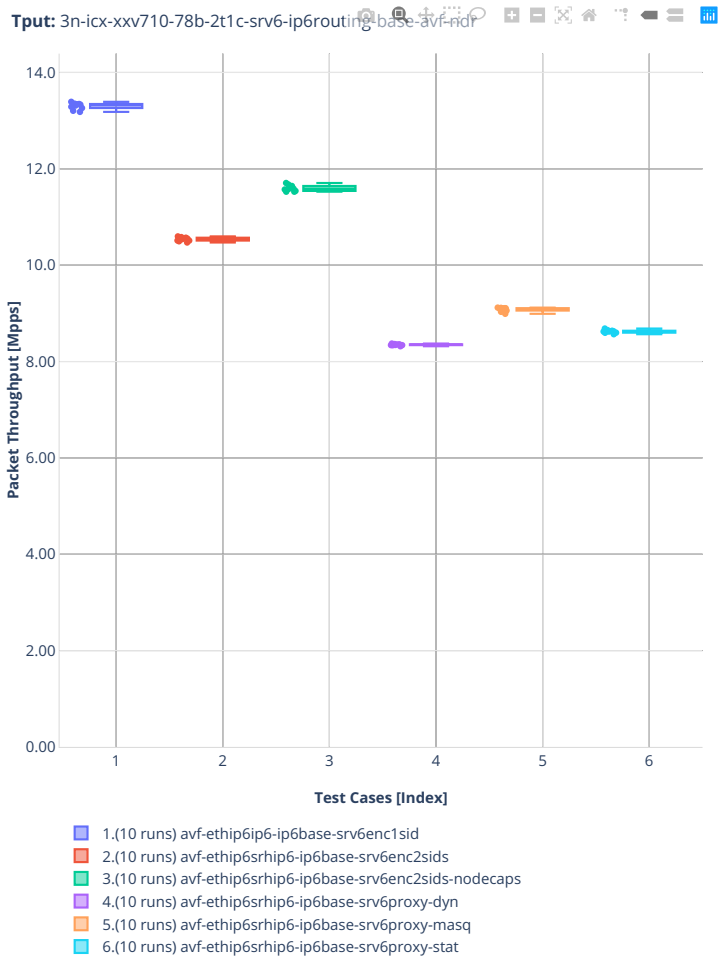
Following sections include summary graphs of VPP Phy-to-Phy performance with SRv6, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

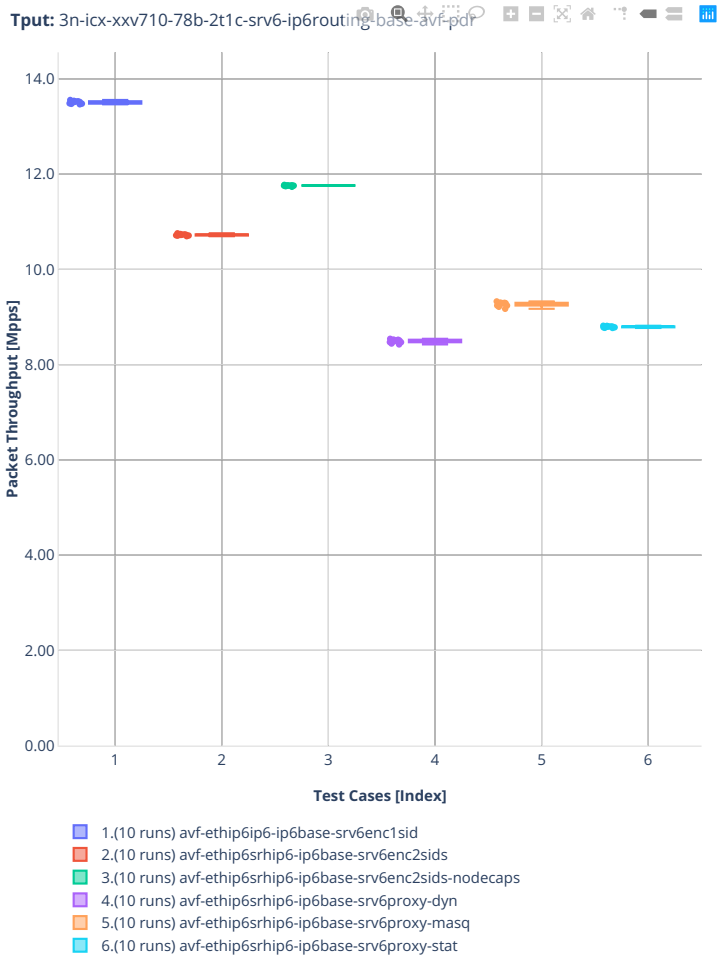
CSIT source code for the test cases used for plots can be found in CSIT git repository[117].

[117] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls2210

3n-icx-xxv710

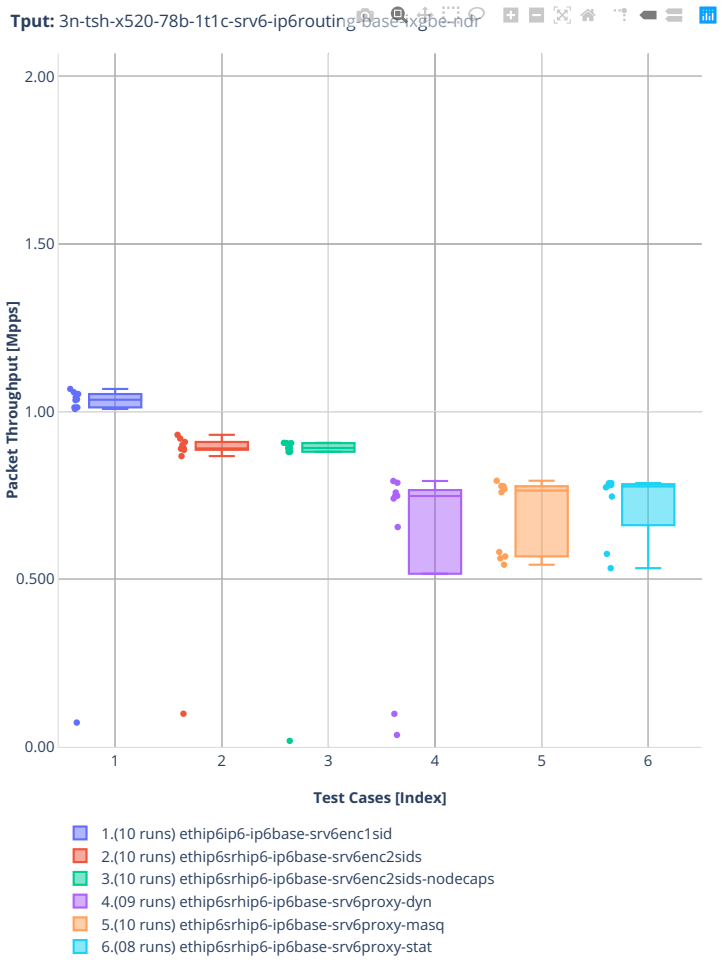
78b-2t1c-srv6-ip6routing-base-avf

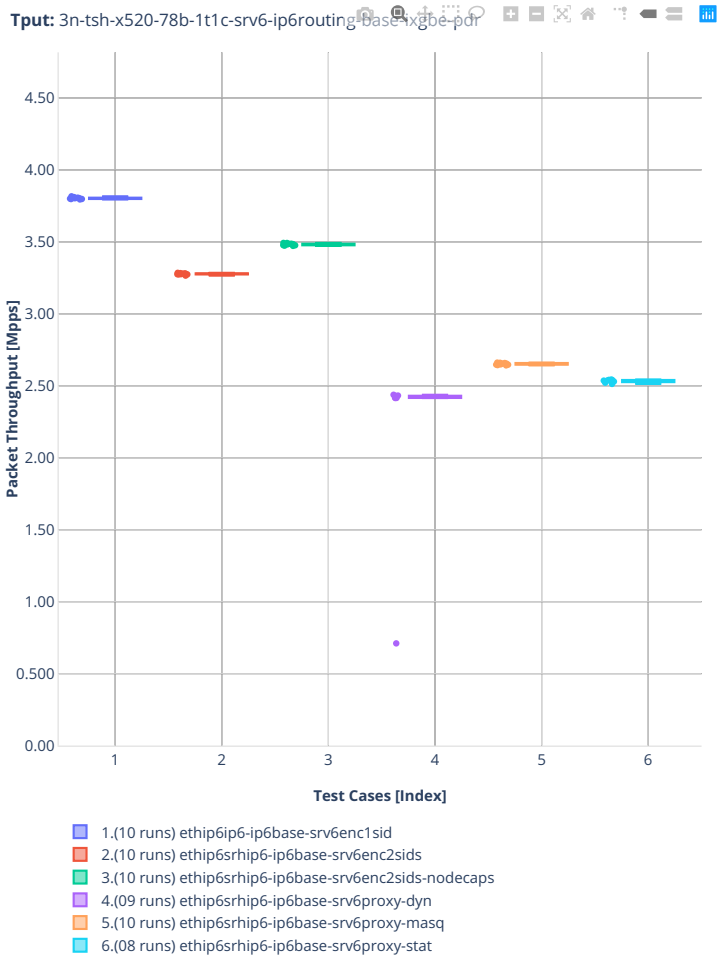



Tput: 3n-icx-xxv710-78b-2t1c-srv6-ip6routing-base-avf-pdr
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip6ip6-ip6base-srv6enc1sid
2.(10 runs) avf-ethip6srhip6-ip6base-srv6enc2sids
3.(10 runs) avf-ethip6srhip6-ip6base-srv6enc2sids-nodecaps
4.(10 runs) avf-ethip6srhip6-ip6base-srv6proxy-dyn
5.(10 runs) avf-ethip6srhip6-ip6base-srv6proxy-masq
6.(10 runs) avf-ethip6srhip6-ip6base-srv6proxy-stat

3n-tsh-x520

78b-1t1c-srv6-ip6routing-base-ixgbe




Tput: 3n-tsh-x520-78b-1t1c-srv6-ip6routing-base-ixgbe-pdr
Packet Throughput [Mpps]
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.500
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip6ip6-ip6base-srv6enc1sid
2.(10 runs) ethip6srhip6-ip6base-srv6enc2sids
3.(10 runs) ethip6srhip6-ip6base-srv6enc2sids-nodecaps
4.(09 runs) ethip6srhip6-ip6base-srv6proxy-dyn
5.(10 runs) ethip6srhip6-ip6base-srv6proxy-masq
6.(08 runs) ethip6srhip6-ip6base-srv6proxy-stat

2.3.5 IPv4 Tunnels

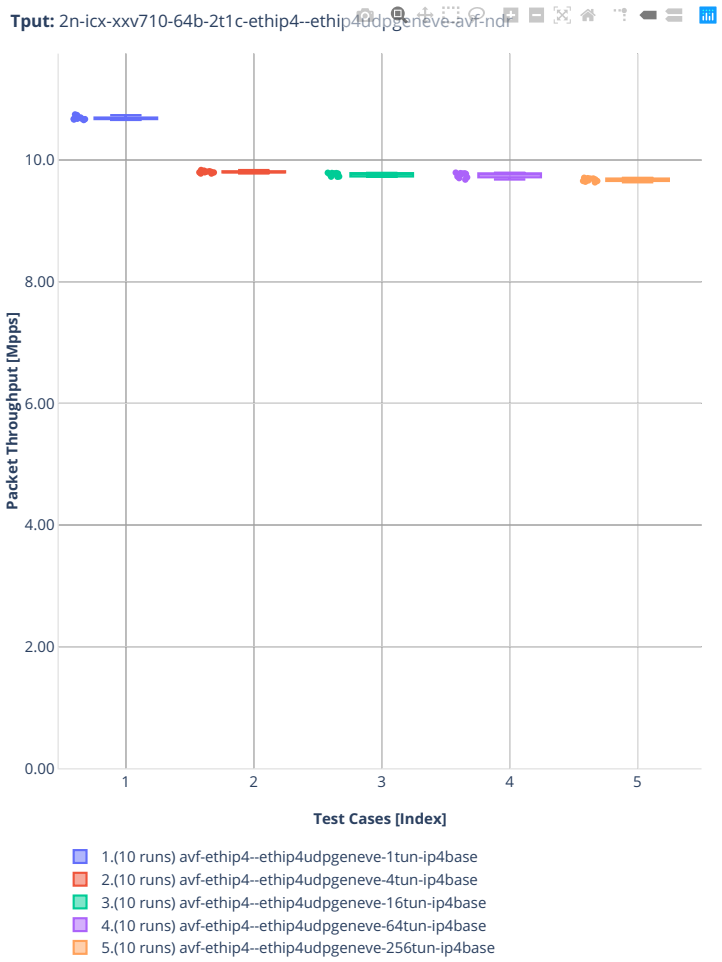
Following sections include summary graphs of VPP Phy-to-Phy performance with IPv4 Overlay Tunnels, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

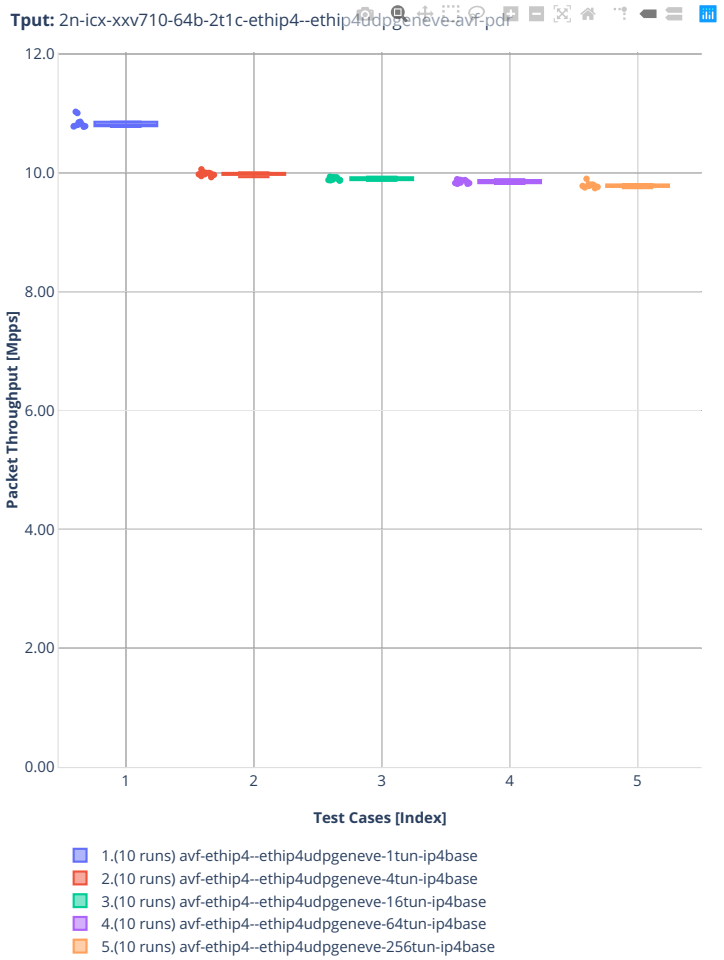
CSIT source code for the test cases used for plots can be found in CSIT git repository[118].

[118] https://git.fd.io/csit/tree/tests/vpp/perf/ip4_tunnels?h=rls2210

2n-icx-xxv710

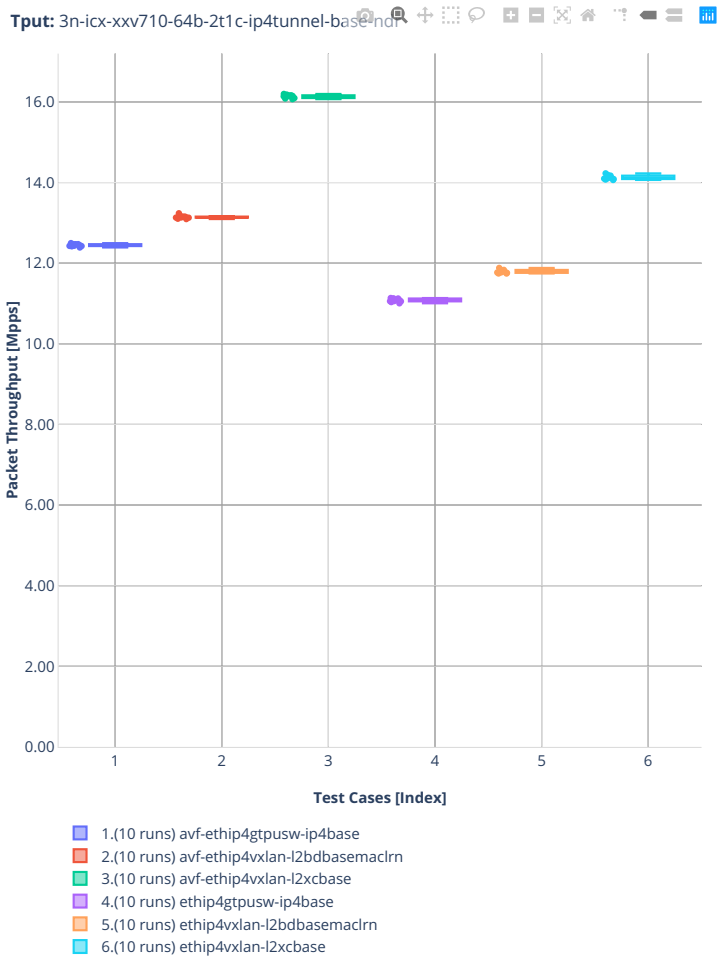
64b-2t1c-ethip4–ethip4udpgeneve-avf

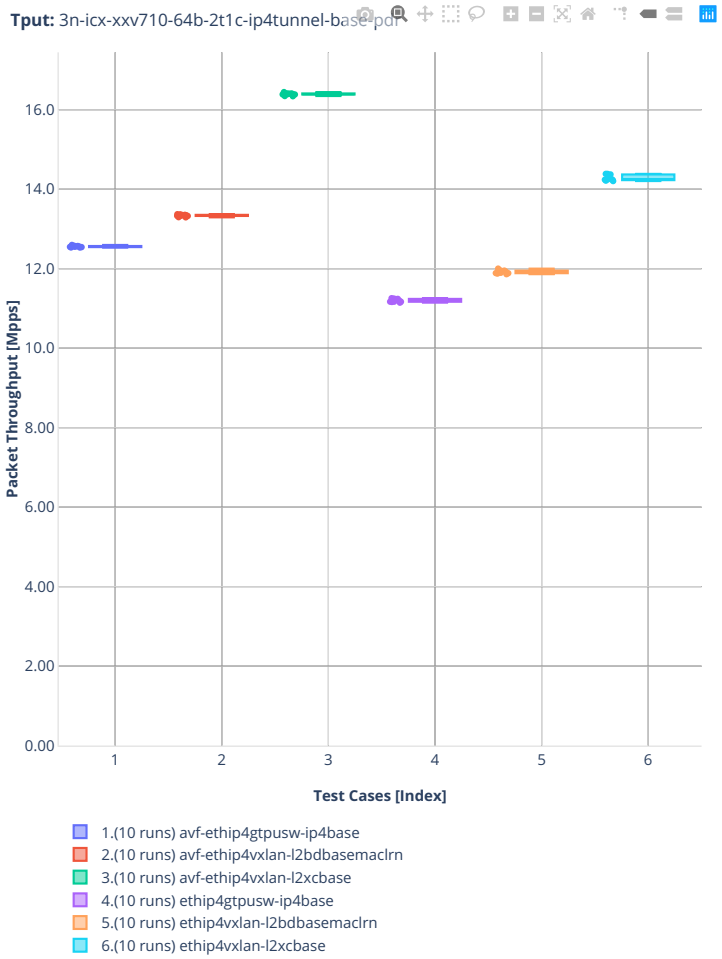



Tput: 2n-icx-xxv710-64b-2t1c-ethip4--ethip4udpgeneve-avf-pdr
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) avf-ethip4--ethip4udpgeneve-1tun-ip4base
2.(10 runs) avf-ethip4--ethip4udpgeneve-4tun-ip4base
3.(10 runs) avf-ethip4--ethip4udpgeneve-16tun-ip4base
4.(10 runs) avf-ethip4--ethip4udpgeneve-64tun-ip4base
5.(10 runs) avf-ethip4--ethip4udpgeneve-256tun-ip4base

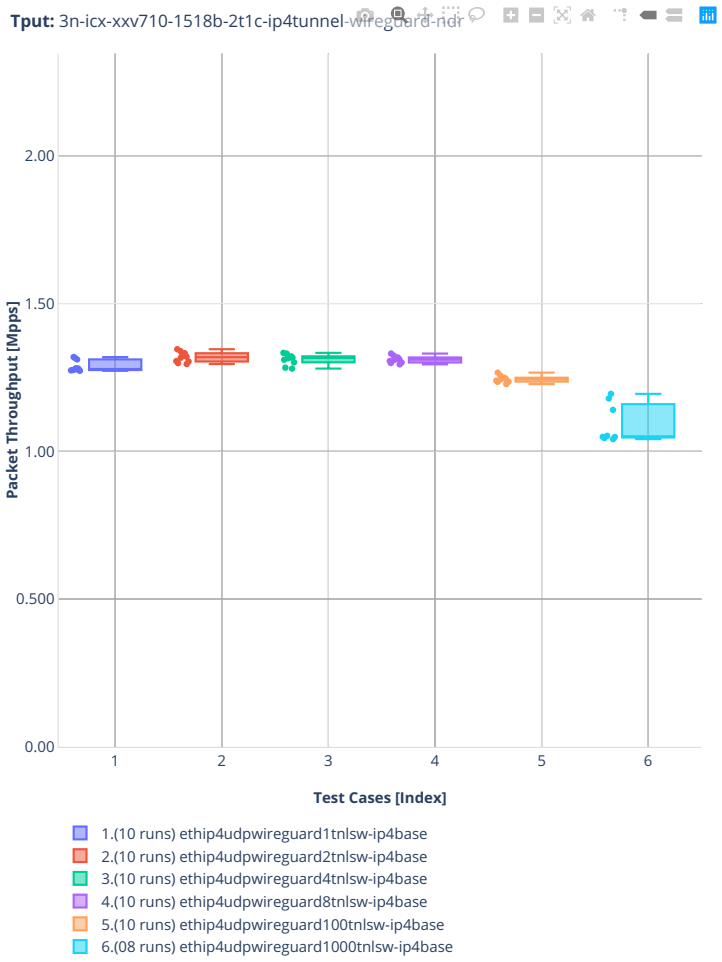
3n-icx-xxv710

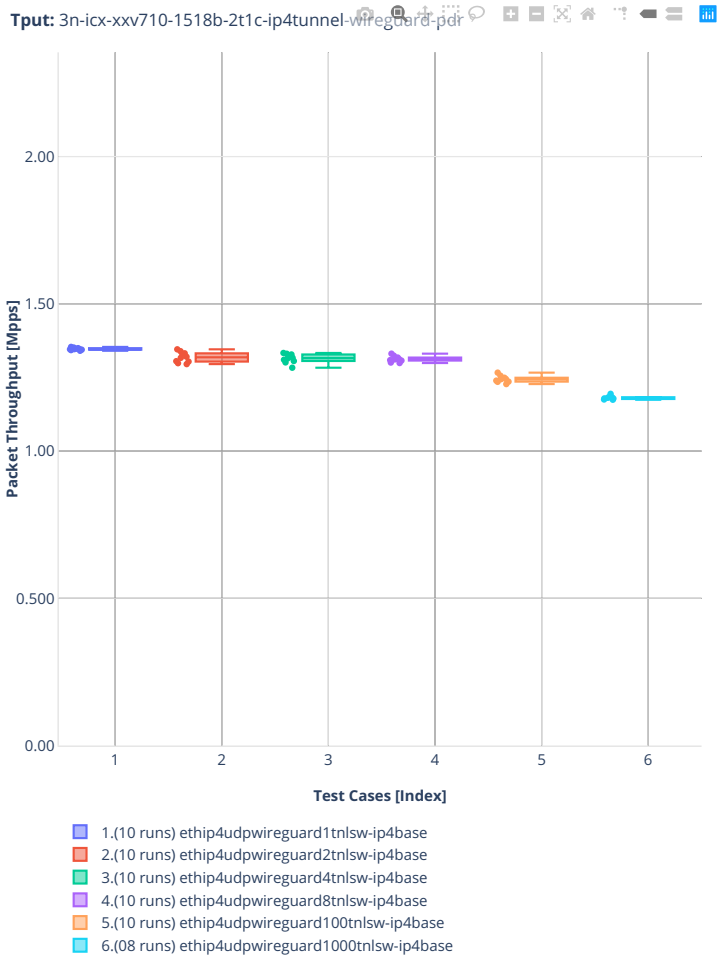
64b-2t1c-ip4tunnel-base



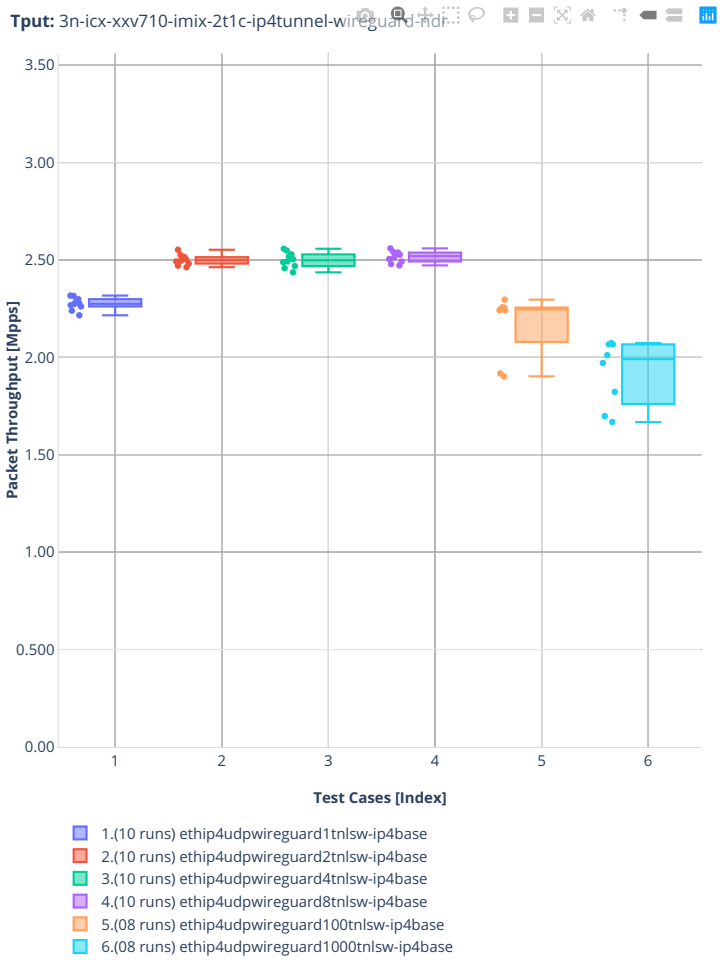

Tput: 3n-icx-xxv710-64b-2t1c-ip4tunnel-base-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4gtpusw-ip4base
2.(10 runs) avf-ethip4vxlan-l2bdbasemaclrn
3.(10 runs) avf-ethip4vxlan-l2xcbase
4.(10 runs) ethip4gtpusw-ip4base
5.(10 runs) ethip4vxlan-l2bdbasemaclrn
6.(10 runs) ethip4vxlan-l2xcbase

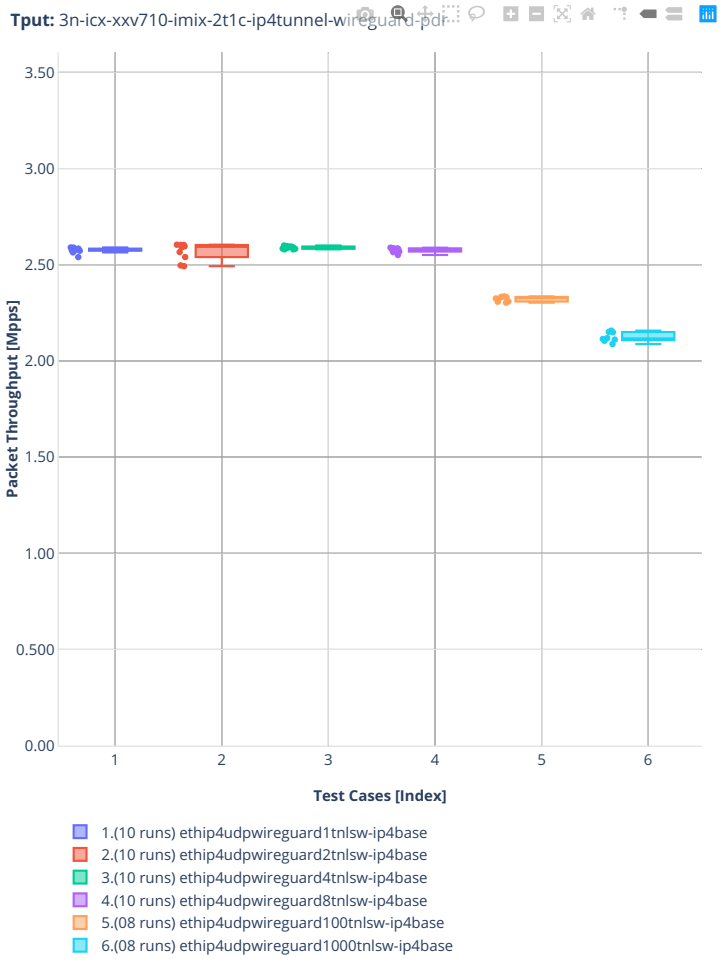
1518b-2t1c-ip4tunnel-wireguard




Tput: 3n-icx-xxv710-1518b-2t1c-ip4tunnel-wireguard-pdr
Packet Throughput [Mpps]
2.00
1.50
1.00
0.500
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4udpwireguard1tnlsw-ip4base
2.(10 runs) ethip4udpwireguard2tnlsw-ip4base
3.(10 runs) ethip4udpwireguard4tnlsw-ip4base
4.(10 runs) ethip4udpwireguard8tnlsw-ip4base
5.(10 runs) ethip4udpwireguard100tnlsw-ip4base
6.(08 runs) ethip4udpwireguard1000tnlsw-ip4base

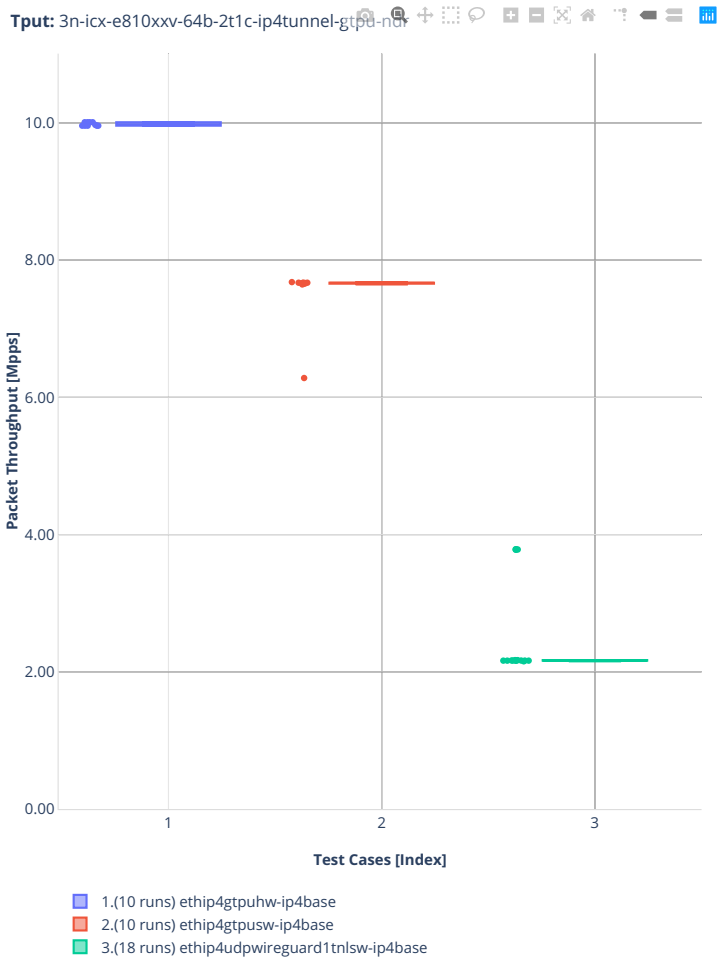
imix-2t1c-ip4tunnel-wireguard

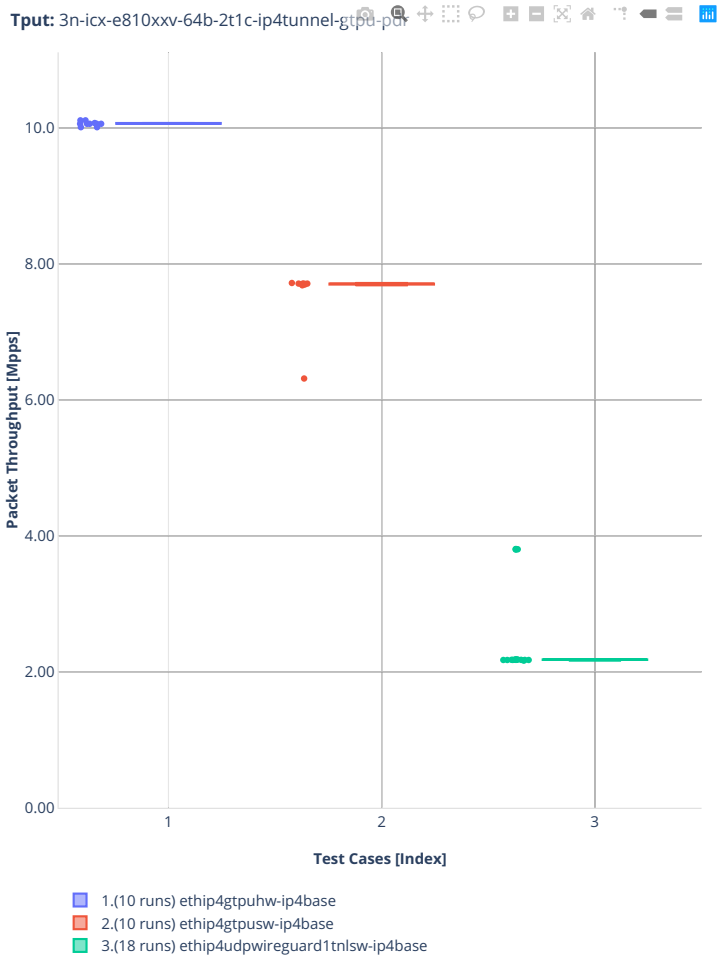



Tput: 3n-icx-xxv710-imix-2t1c-ip4tunnel-wireguard-pdr
Packet Throughput [Mpps]
3.50
3.00
2.50
2.00
1.50
1.00
0.500
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4udpwireguard1tnlsw-ip4base
2.(10 runs) ethip4udpwireguard2tnlsw-ip4base
3.(10 runs) ethip4udpwireguard4tnlsw-ip4base
4.(10 runs) ethip4udpwireguard8tnlsw-ip4base
5.(08 runs) ethip4udpwireguard100tnlsw-ip4base
6.(08 runs) ethip4udpwireguard1000tnlsw-ip4base

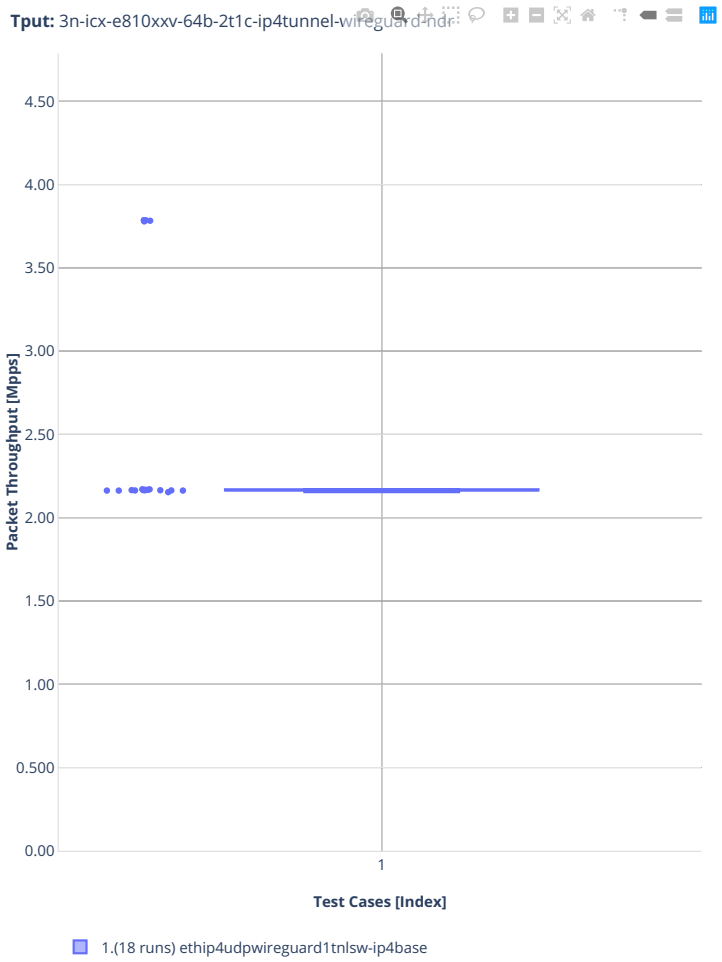
3n-icx-e810xxv

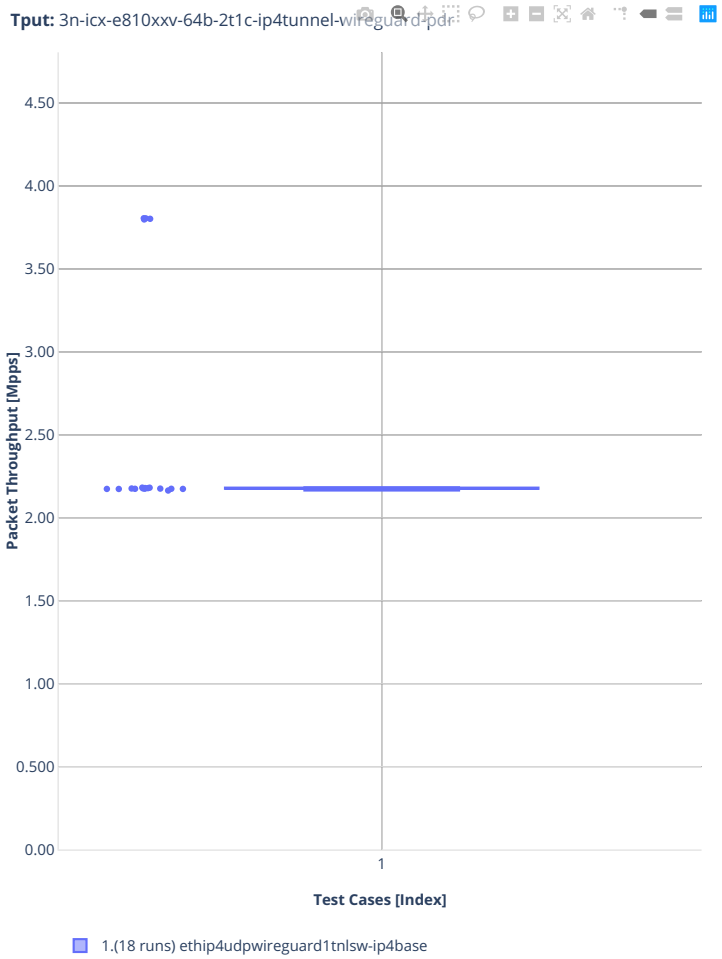
64b-2t1c-ip4tunnel-gtpu




Tput: 3n-icx-e810xxv-64b-2t1c-ip4tunnel-gtpu-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4gtpuhw-ip4base
2.(10 runs) ethip4gtpusw-ip4base
3.(18 runs) ethip4udpwireguard1tnlsw-ip4base

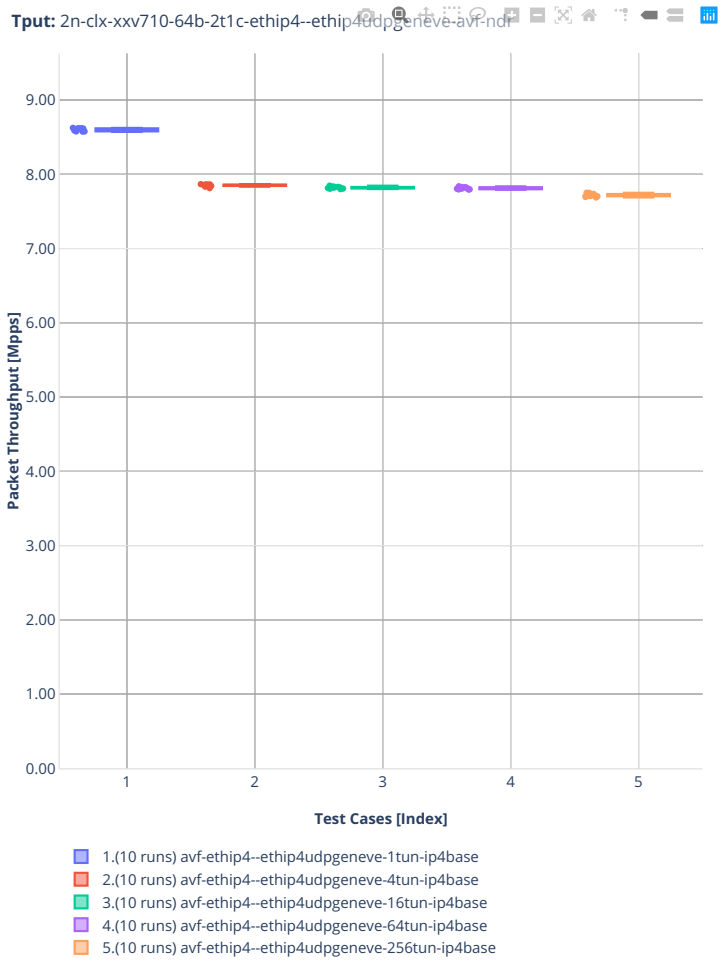
64b-2t1c-ip4tunnel-wireguard

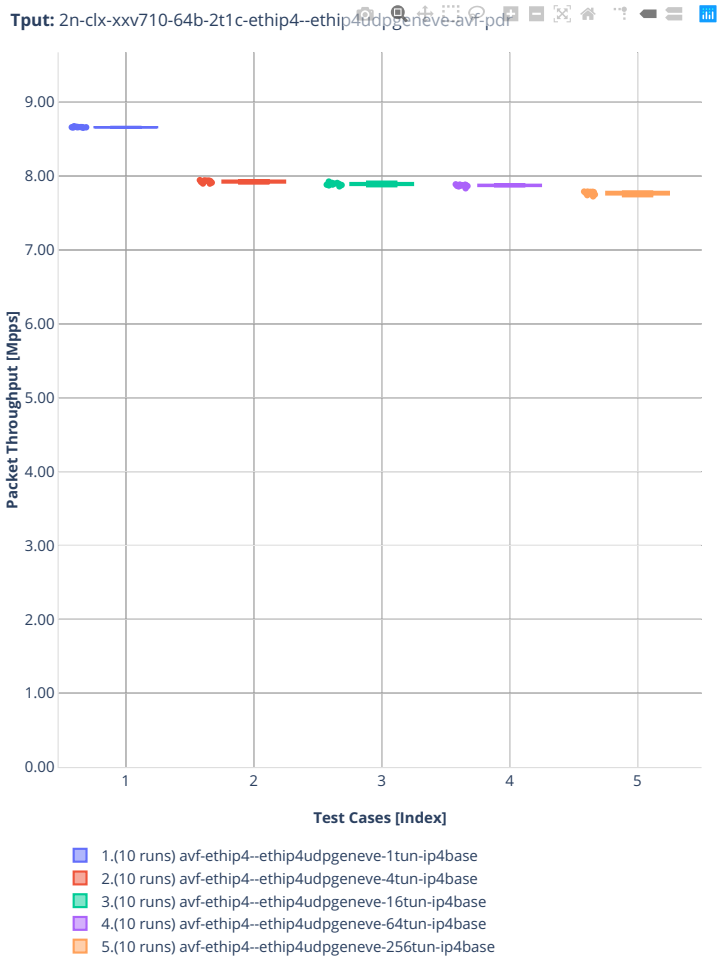



Tput: 3n-icx-e810xxv-64b-2t1c-ip4tunnel-wireguard-pdr
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
1.(18 runs) ethip4udpwireguard1tnlsw-ip4base

2n-clx-xxv710

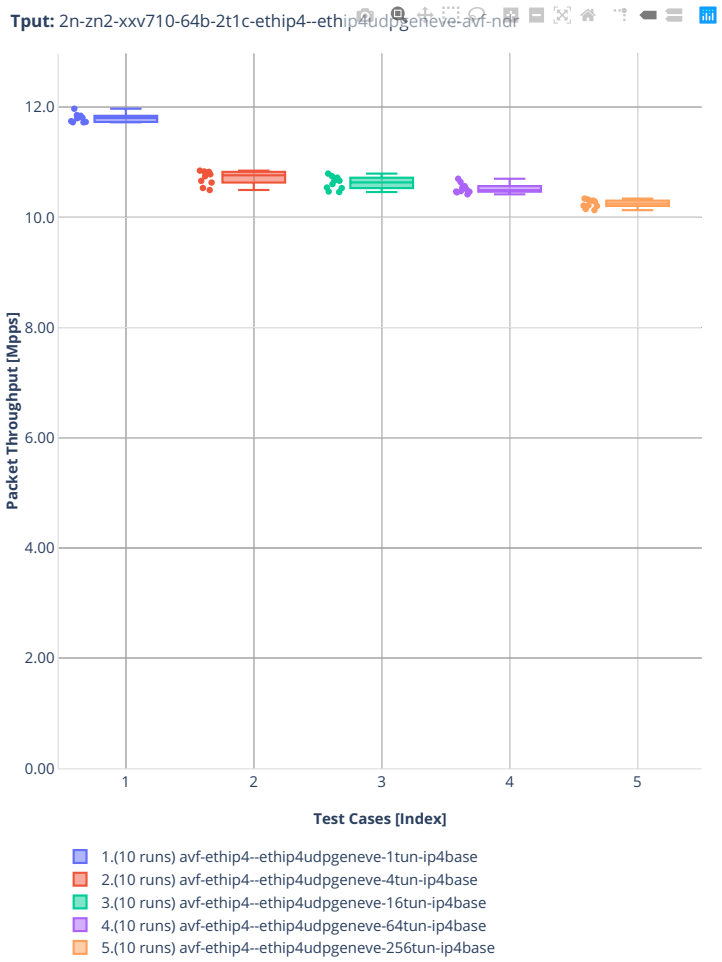
64b-2t1c-ethip4–ethip4udpgeneve-avf

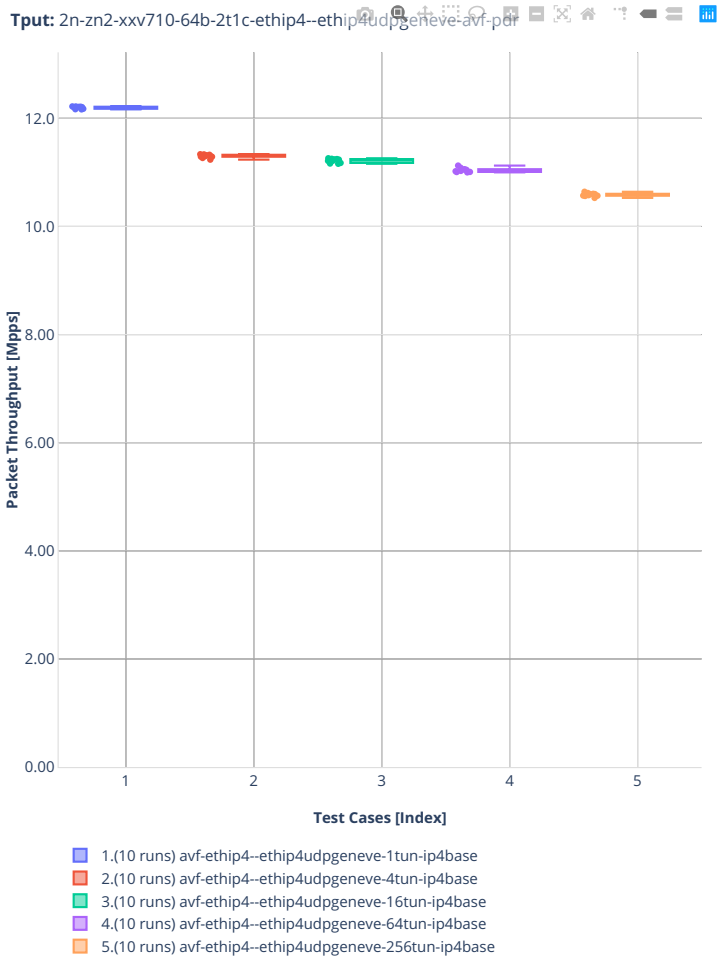



Tput: 2n-clx-xxv710-64b-2t1c-ethip4--ethip4udpgeneve-avf-pdr
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) avf-ethip4--ethip4udpgeneve-1tun-ip4base
2.(10 runs) avf-ethip4--ethip4udpgeneve-4tun-ip4base
3.(10 runs) avf-ethip4--ethip4udpgeneve-16tun-ip4base
4.(10 runs) avf-ethip4--ethip4udpgeneve-64tun-ip4base
5.(10 runs) avf-ethip4--ethip4udpgeneve-256tun-ip4base

2n-zn2-xxv710

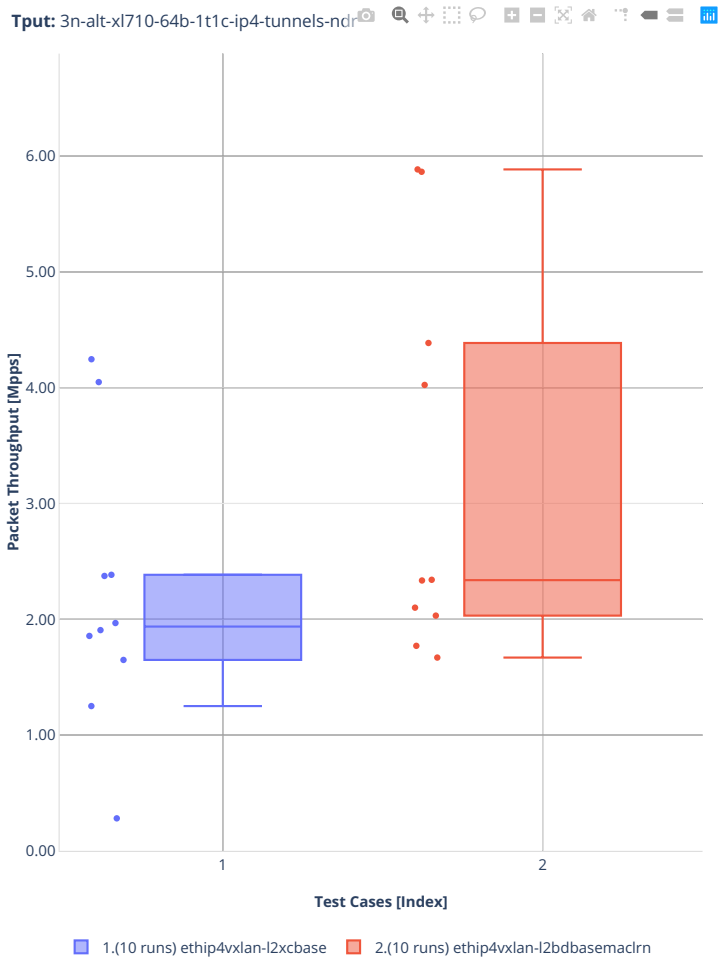
64b-2t1c-ethip4–ethip4udpgeneve-avf

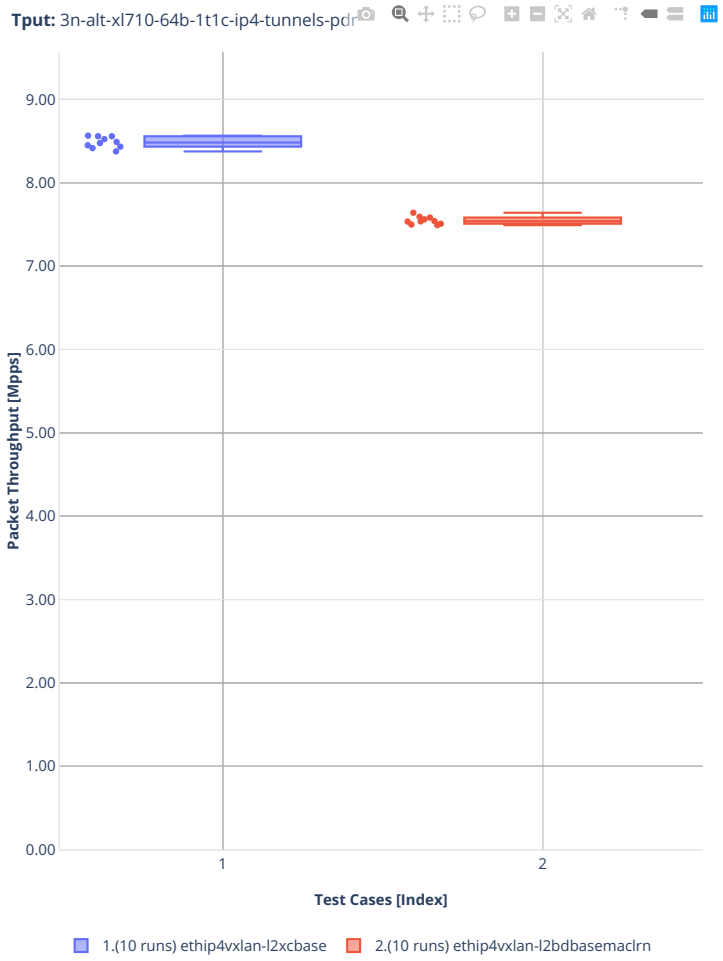



Tput: 2n-zn2-xxv710-64b-2t1c-ethip4--ethip4udpgeneve-avf-pdr
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
Test Cases [Index]
1.(10 runs) avf-ethip4--ethip4udpgeneve-1tun-ip4base
2.(10 runs) avf-ethip4--ethip4udpgeneve-4tun-ip4base
3.(10 runs) avf-ethip4--ethip4udpgeneve-16tun-ip4base
4.(10 runs) avf-ethip4--ethip4udpgeneve-64tun-ip4base
5.(10 runs) avf-ethip4--ethip4udpgeneve-256tun-ip4base

3n-alt-xl710

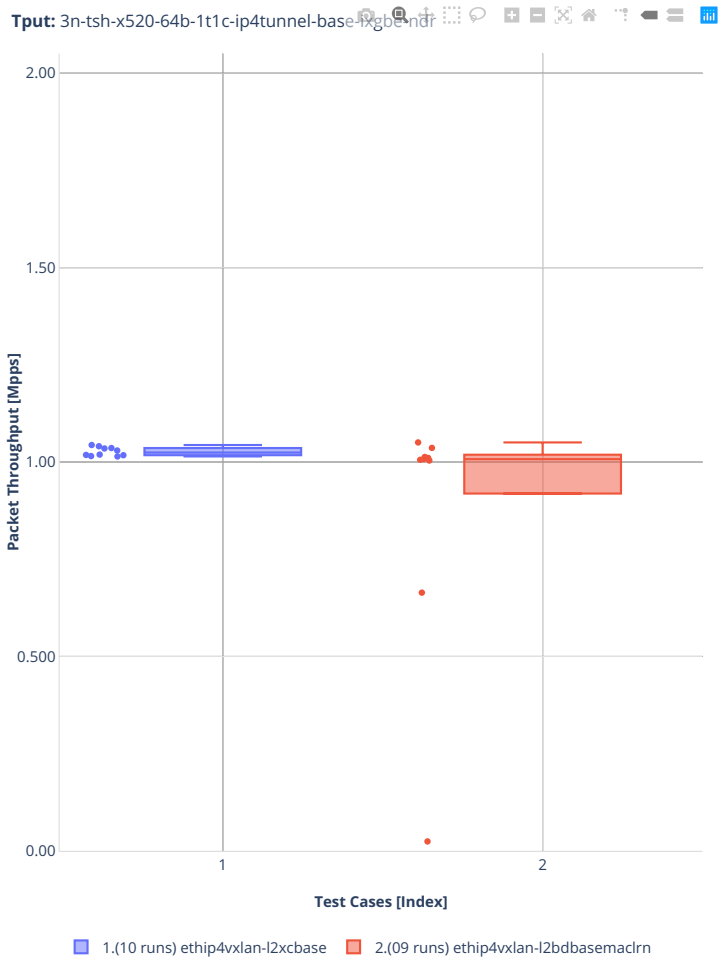
64b-1t1c-ip4tunnel-base

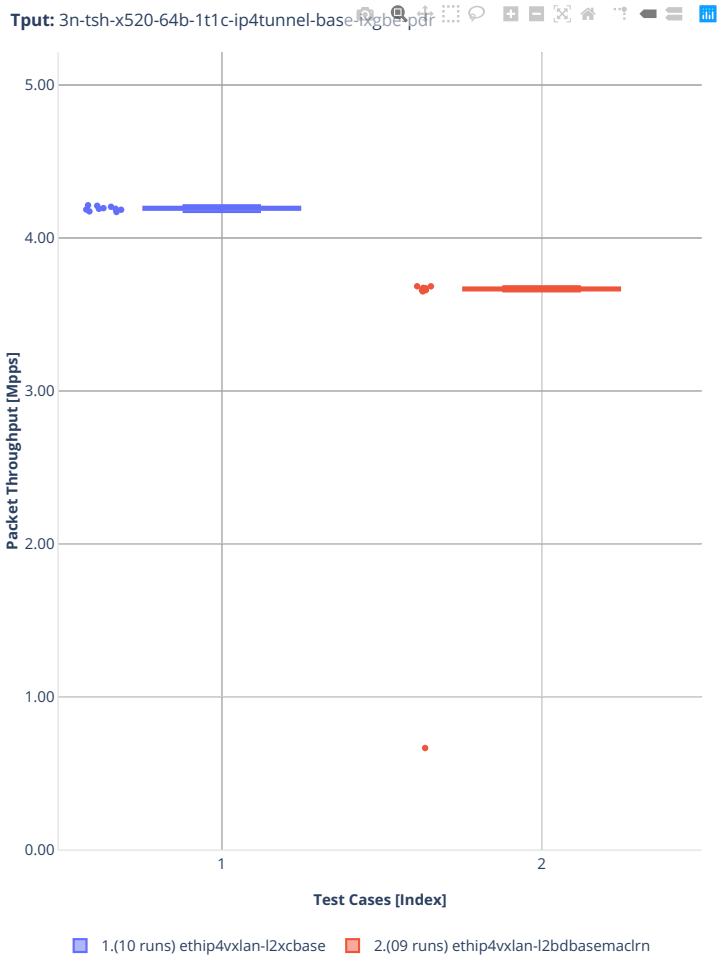



Tput: 3n-alt-xl710-64b-1t1c-ip4-tunnels-pdr
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) ethip4vxlan-l2xcbase
2.(10 runs) ethip4vxlan-l2bdbasemaclrn

3n-tsh-x520

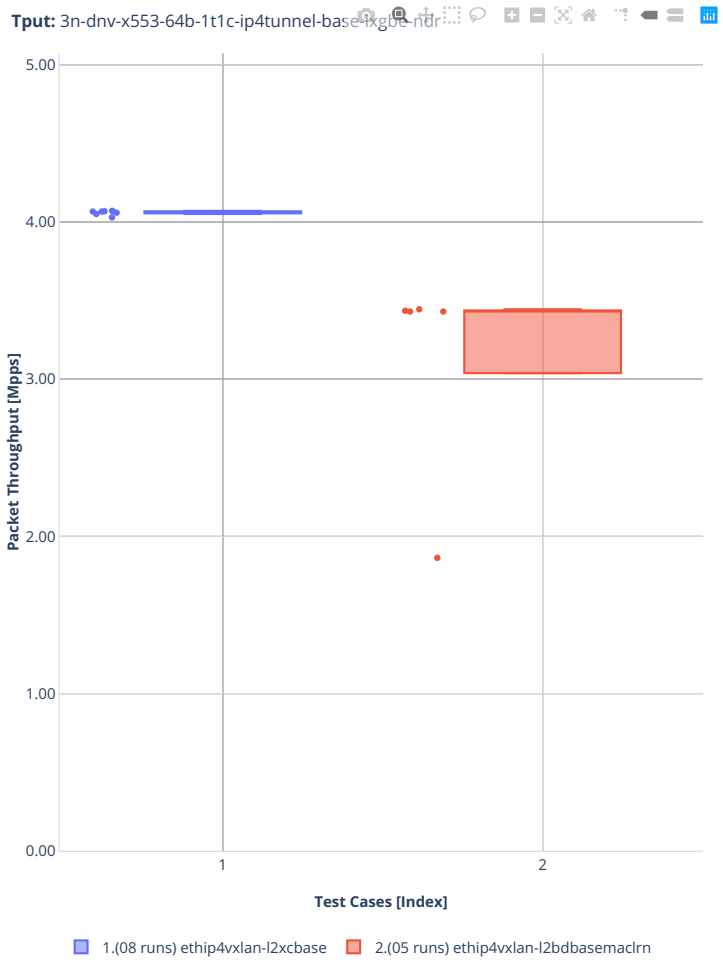
64b-1t1c-ip4tunnel-base-ixgbe

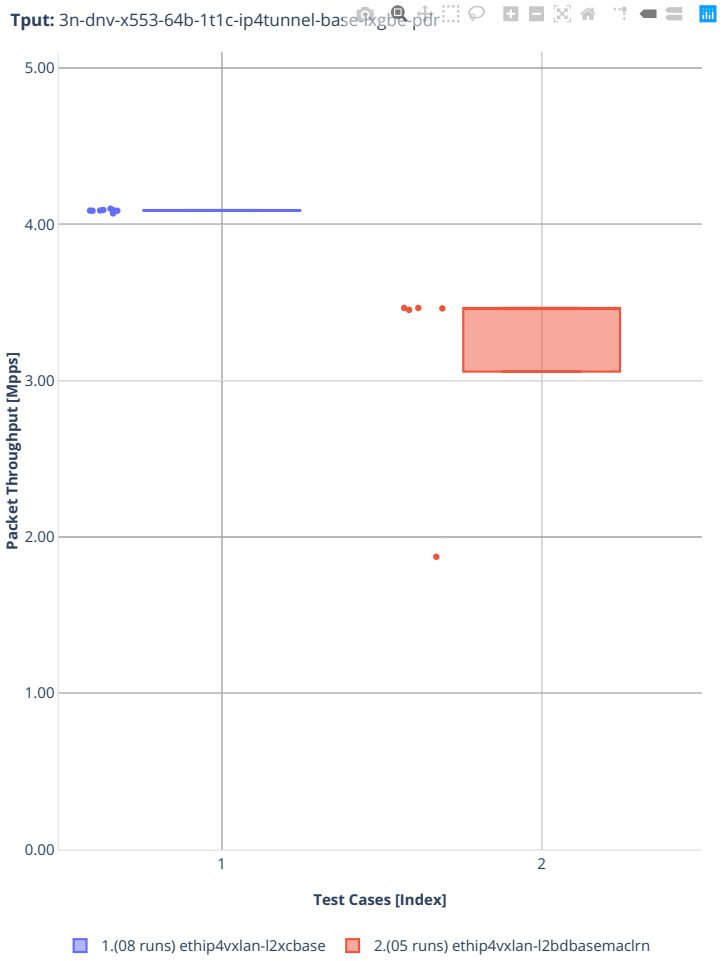



Tput: 3n-tsh-x520-64b-1t1c-ip4tunnel-base-ixgbe-pdr
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) ethip4vxlan-l2xcbase
2.(09 runs) ethip4vxlan-l2bdbasemaclrn

3n-dnv-x553

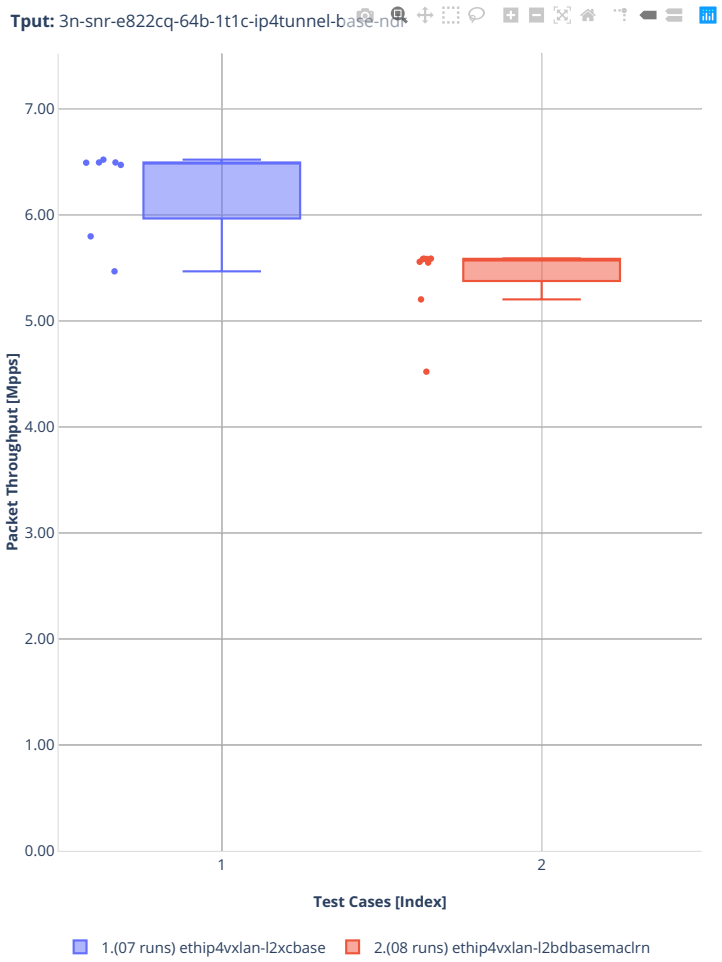
64b-1t1c-ip4tunnel-base-ixgbe

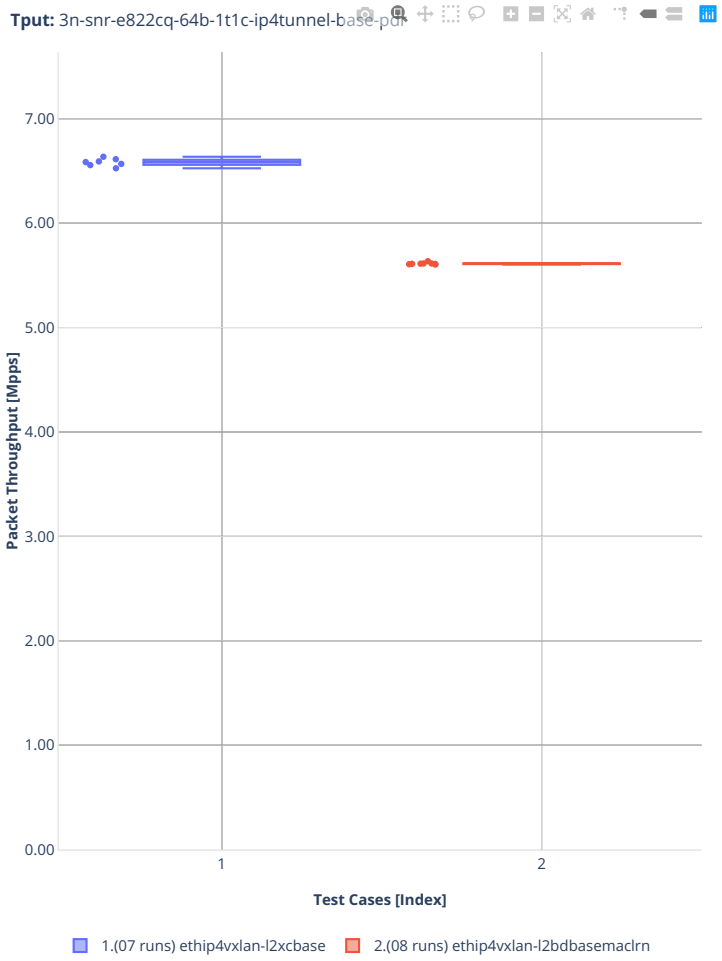



Tput: 3n-dnv-x553-64b-1t1c-ip4tunnel-base-ixgbe-pdr
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(08 runs) ethip4vxlan-l2xcbase
2.(05 runs) ethip4vxlan-l2bdbasemaclrn

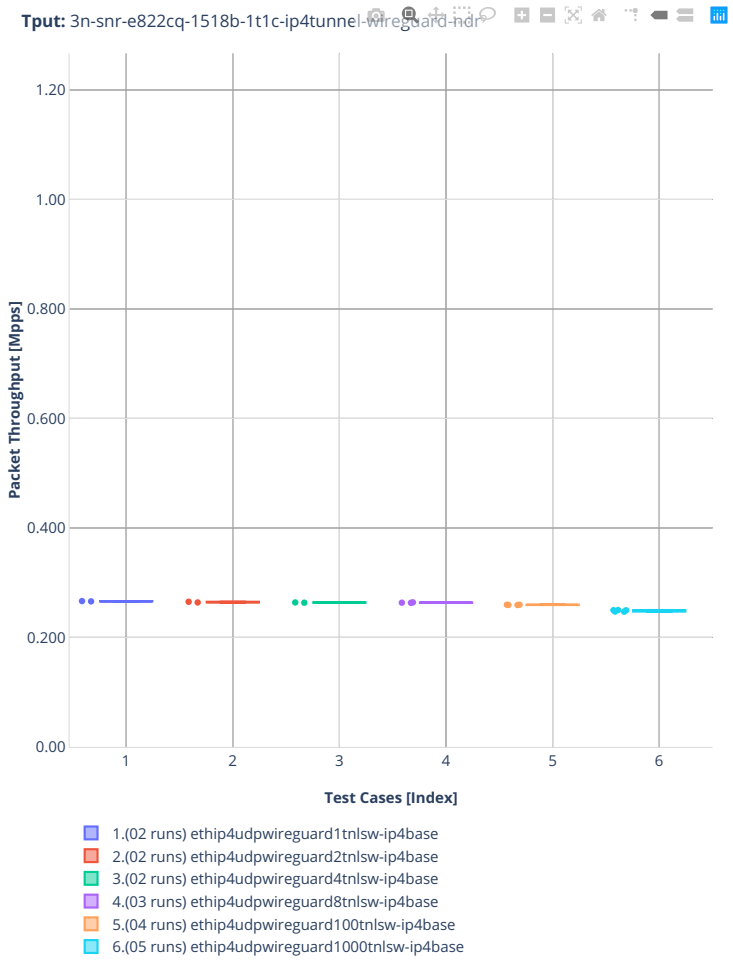
3n-snr-e822cq

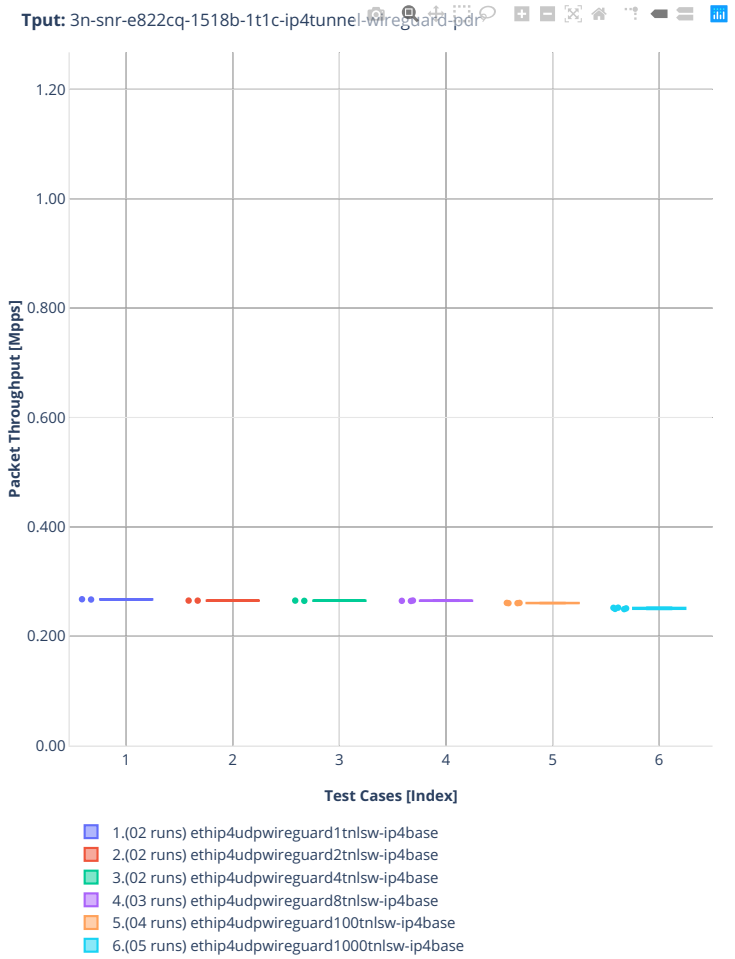
64b-1t1c-ip4tunnel-base



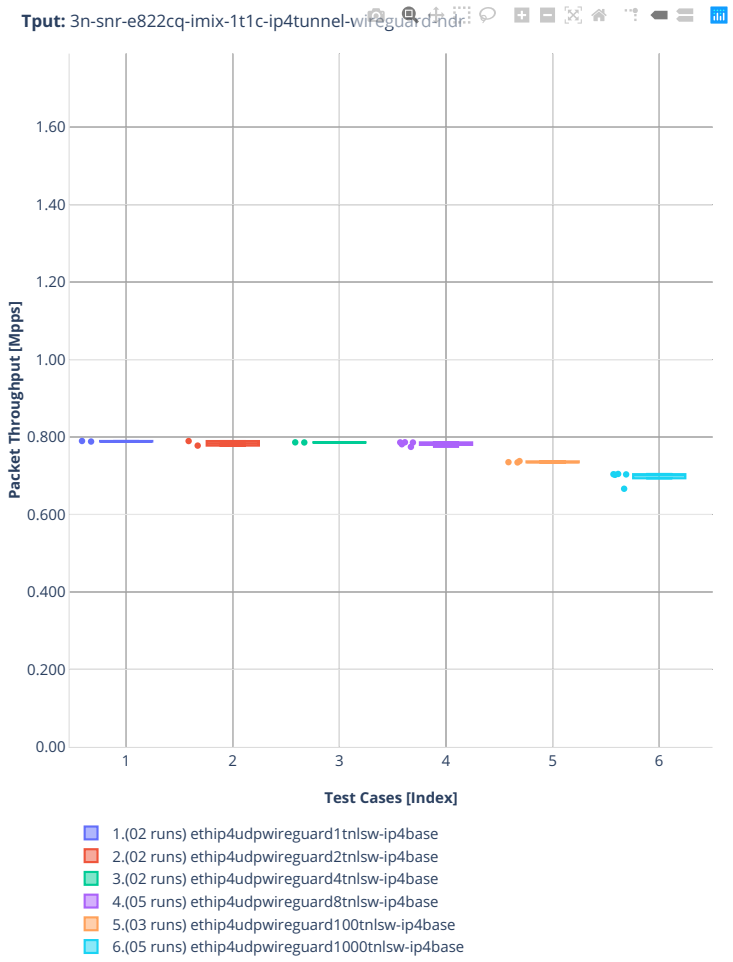

Tput: 3n-snr-e822cq-64b-1t1c-ip4tunnel-base-pdr
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(07 runs) ethip4vxlan-l2xcbase
2.(08 runs) ethip4vxlan-l2bdbasemaclrn

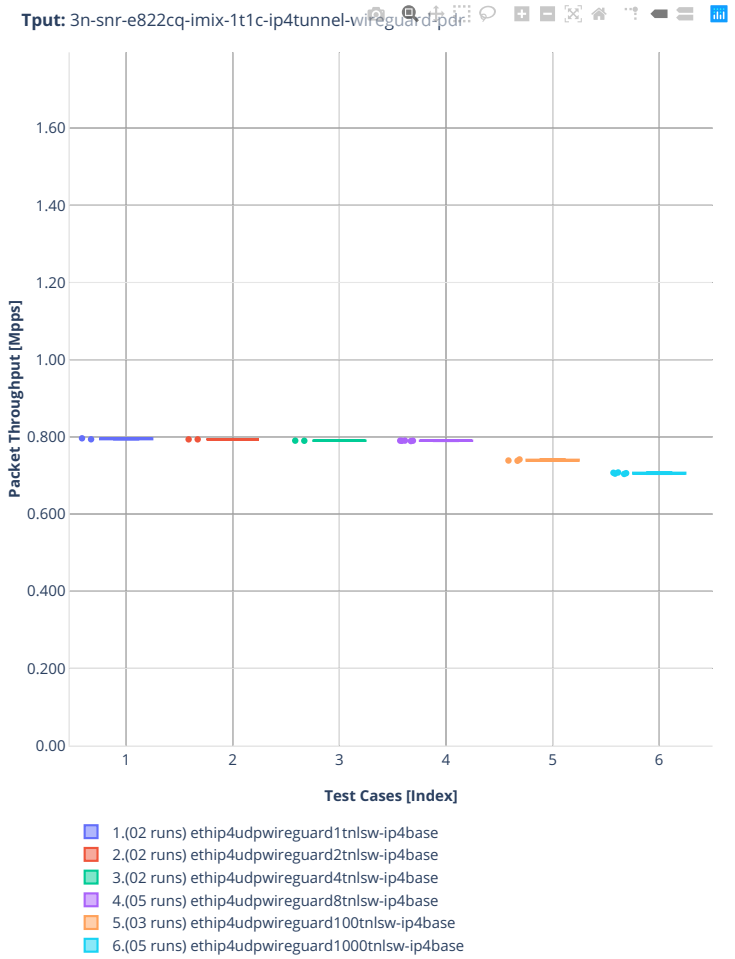
1518b-1t1c-ip4tunnel-wireguard




Tput: 3n-snr-e822cq-1518b-1t1c-ip4tunnel-wireguard-pdr
Packet Throughput [Mpps]
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(02 runs) ethip4udpwireguard1tnlsw-ip4base
2.(02 runs) ethip4udpwireguard2tnlsw-ip4base
3.(02 runs) ethip4udpwireguard4tnlsw-ip4base
4.(03 runs) ethip4udpwireguard8tnlsw-ip4base
5.(04 runs) ethip4udpwireguard100tnlsw-ip4base
6.(05 runs) ethip4udpwireguard1000tnlsw-ip4base

imix-1t1c-ip4tunnel-wireguard




Tput: 3n-snr-e822cq-imix-1t1c-ip4tunnel-wireguard-pdr
Packet Throughput [Mpps]
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(02 runs) ethip4udpwireguard1tnlsw-ip4base
2.(02 runs) ethip4udpwireguard2tnlsw-ip4base
3.(02 runs) ethip4udpwireguard4tnlsw-ip4base
4.(05 runs) ethip4udpwireguard8tnlsw-ip4base
5.(03 runs) ethip4udpwireguard100tnlsw-ip4base
6.(05 runs) ethip4udpwireguard1000tnlsw-ip4base

2.3.6 NAT44 IPv4 Routing

Following sections include summary graphs of VPP Phy-to-Phy performance with IPv4 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

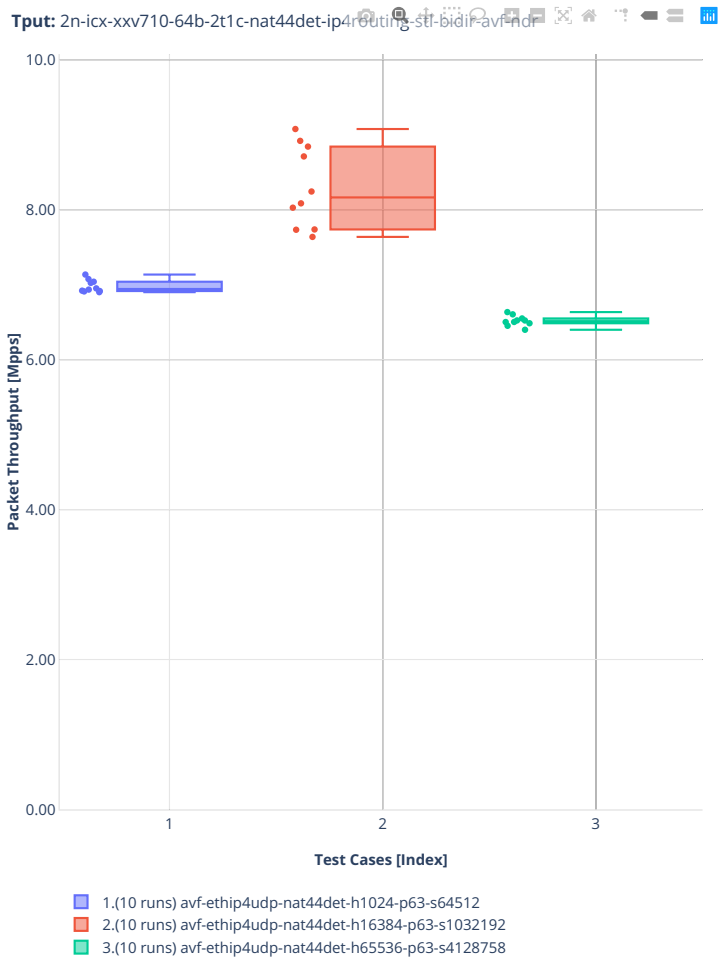
CSIT source code for the test cases used for plots can be found in CSIT git repository[119].

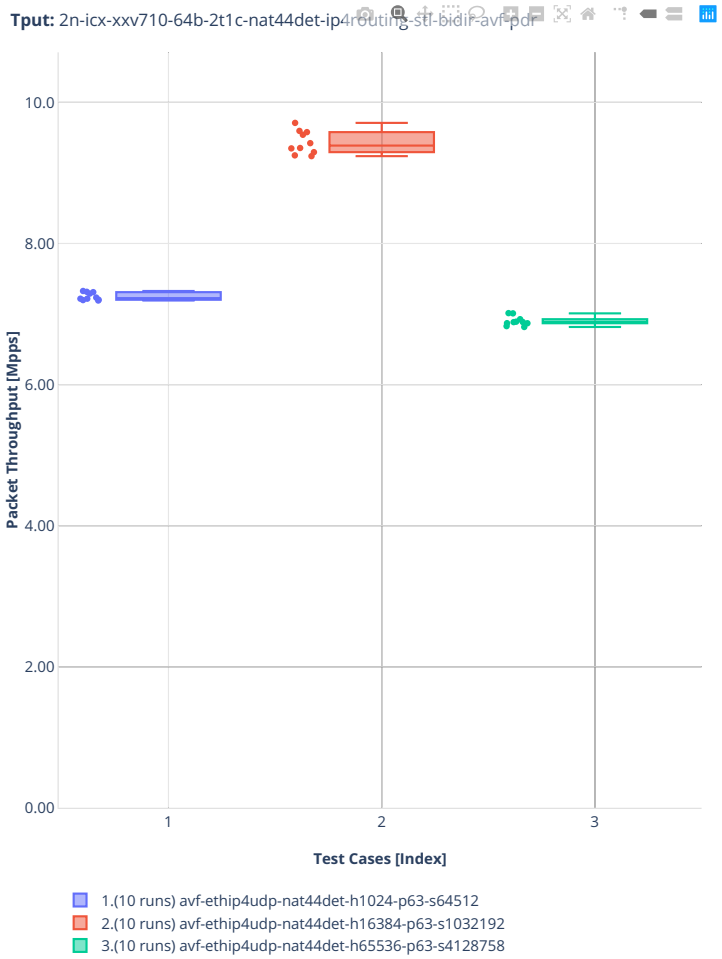
[119] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls2210

Det BiDir

2n-icx-xxv710

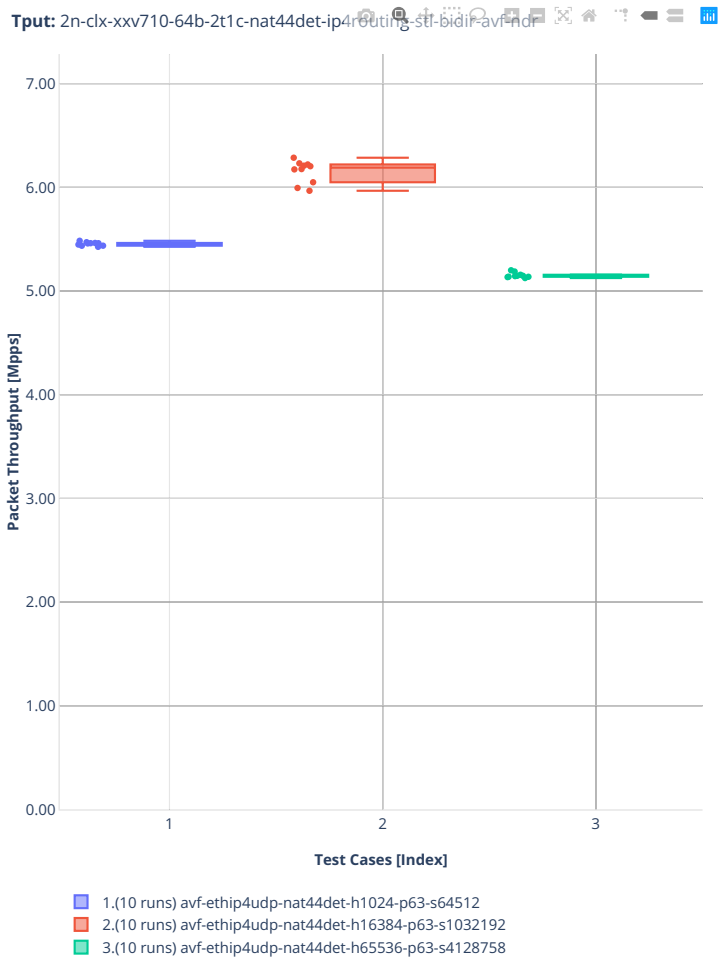
64b-nat44det-ip4routing-stl-bidir-avf

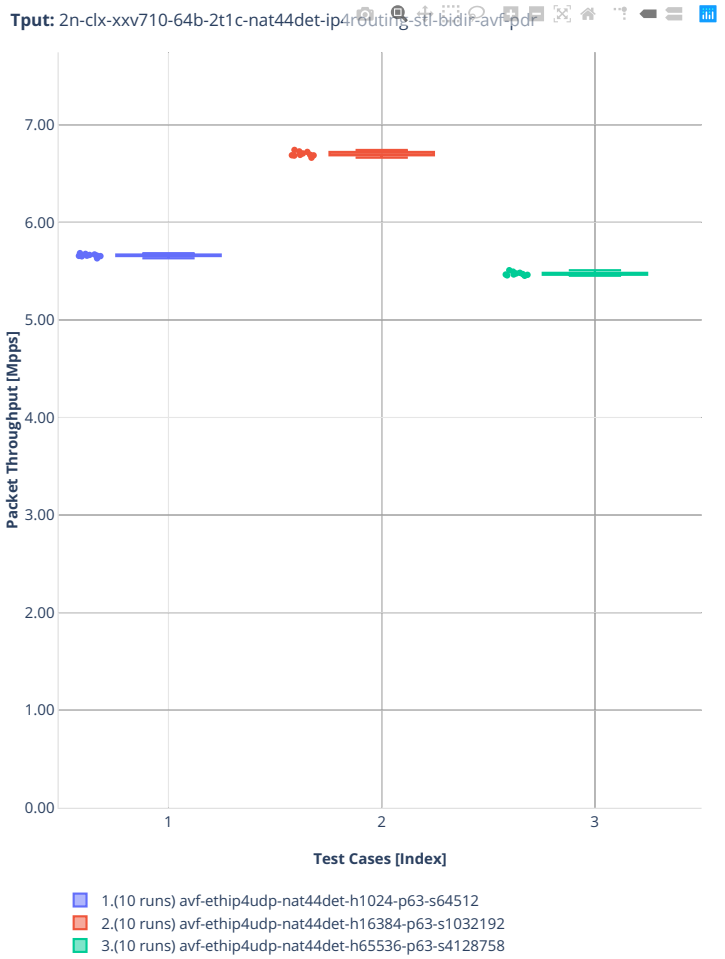



Tput: 2n-icx-xxv710-64b-2t1c-nat44det-ip4routing-stl-bidir-avf-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-ethip4udp-nat44det-h1024-p63-s64512
2.(10 runs) avf-ethip4udp-nat44det-h16384-p63-s1032192
3.(10 runs) avf-ethip4udp-nat44det-h65536-p63-s4128758

2n-clx-xxv710

64b-nat44det-ip4routing-stl-bidir-avf

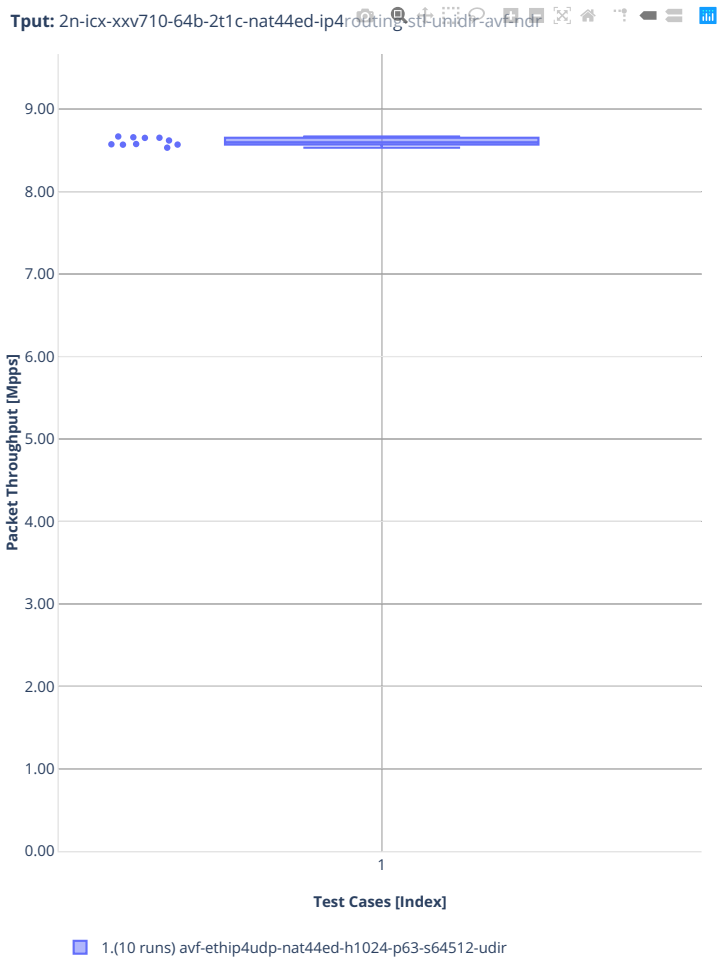


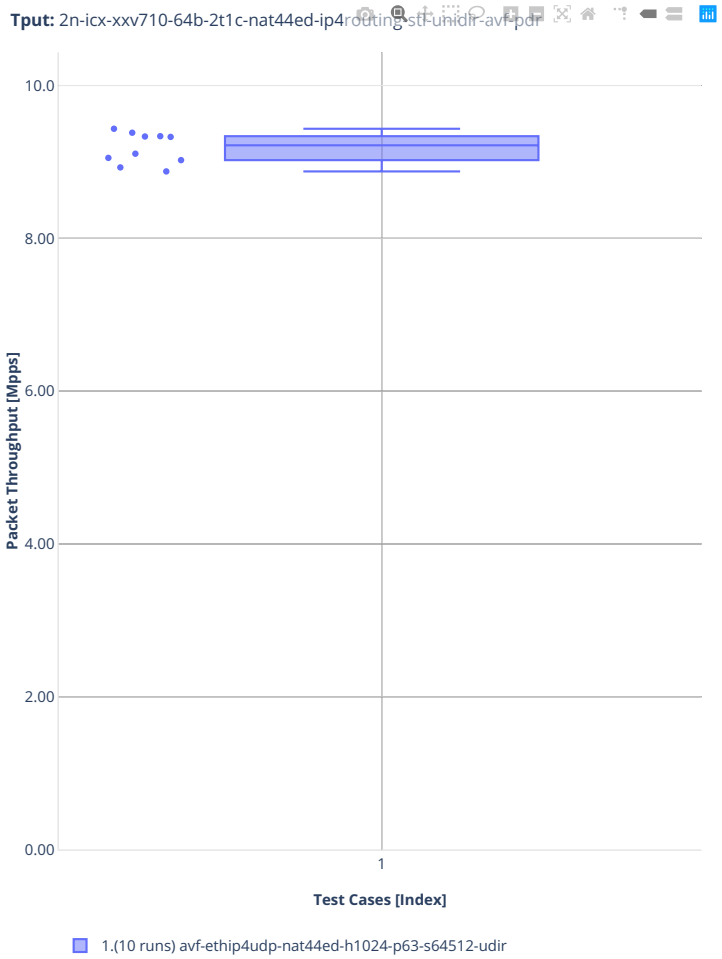

Tput: 2n-clx-xxv710-64b-2t1c-nat44det-ip4routing-stl-bidir-avf-pdr
Packet Throughput [Mpps]
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
Test Cases [Index]
1.(10 runs) avf-ethip4udp-nat44det-h1024-p63-s64512
2.(10 runs) avf-ethip4udp-nat44det-h16384-p63-s1032192
3.(10 runs) avf-ethip4udp-nat44det-h65536-p63-s4128758

ED UniDir

2n-icx-xxv710

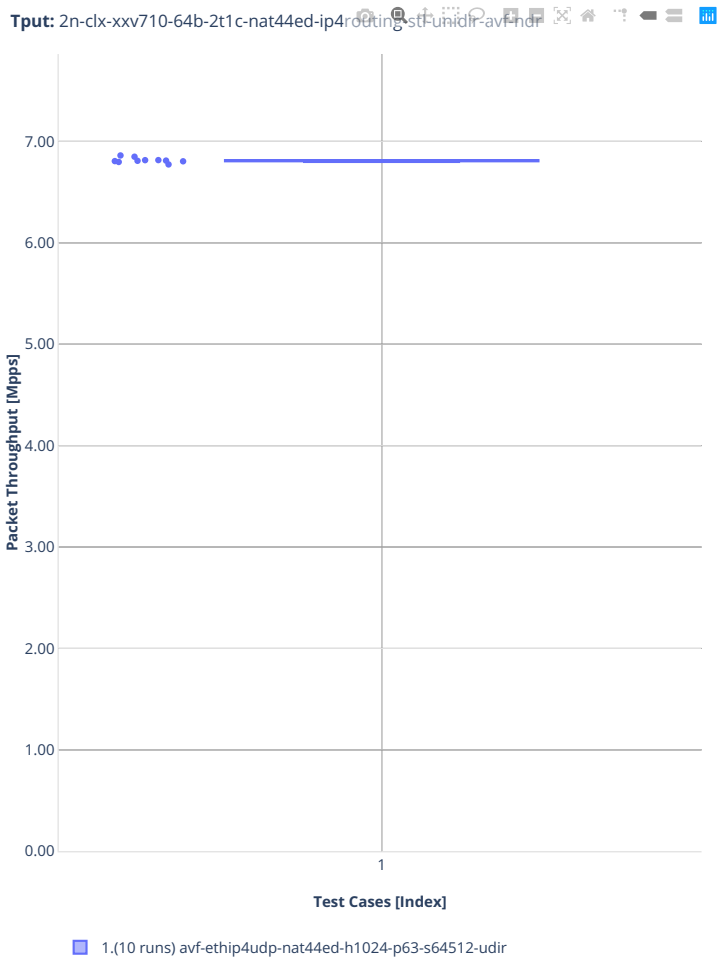
64b-nat44ed-ip4routing-stl-unidir-avf

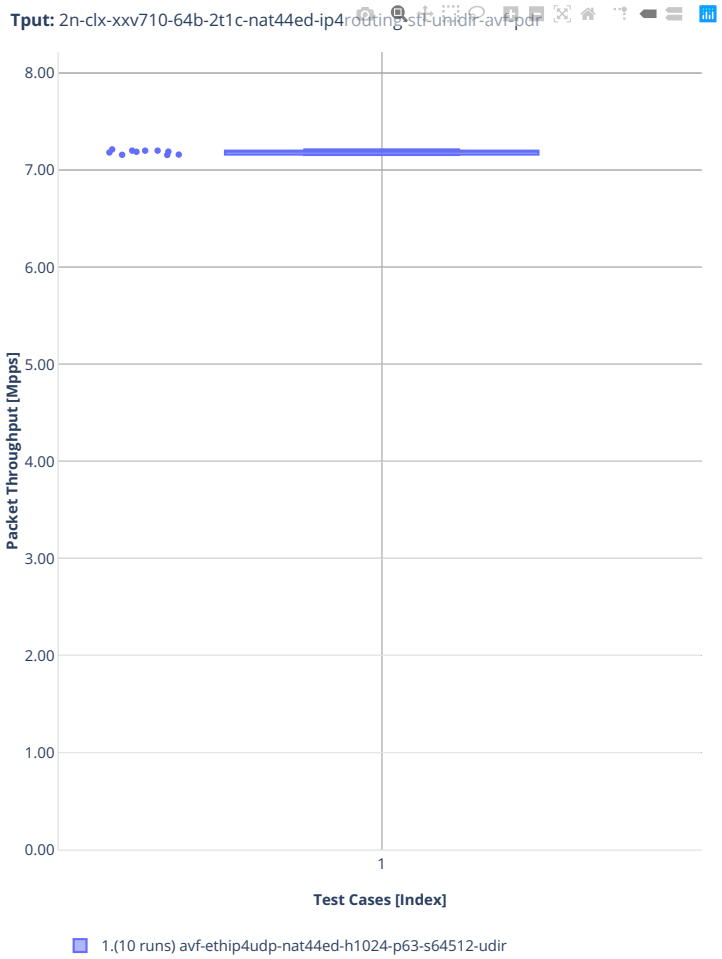



Tput: 2n-icx-xxv710-64b-2t1c-nat44ed-ip4routing-stl-unidir-avf-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
1.(10 runs) avf-ethip4udp-nat44ed-h1024-p63-s64512-udir

2n-clx-xxv710

64b-nat44ed-ip4routing-stl-unidir-avf

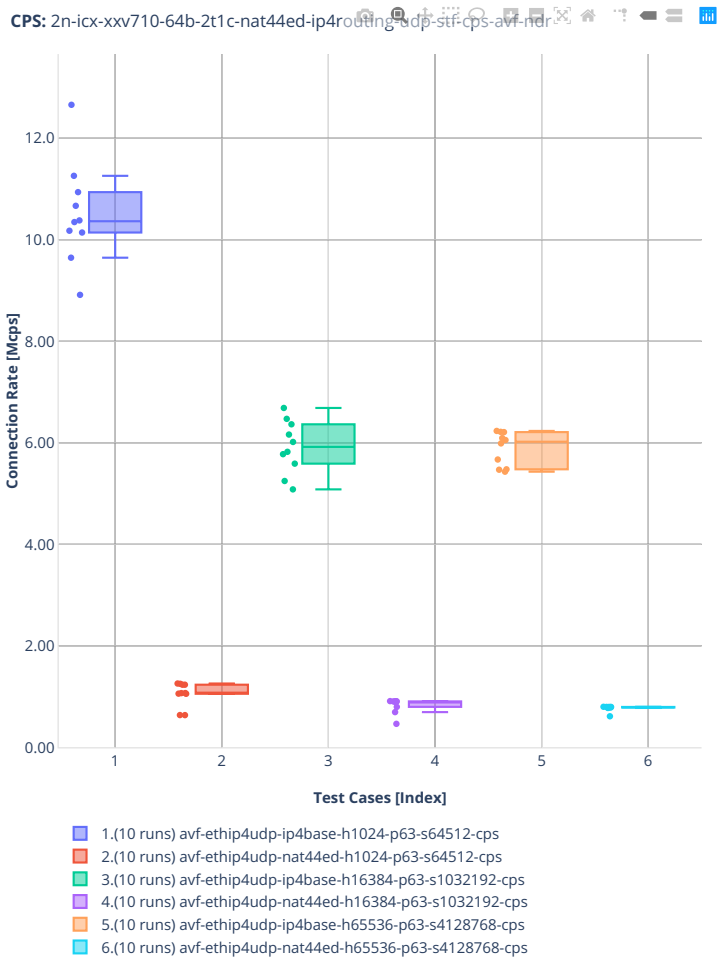


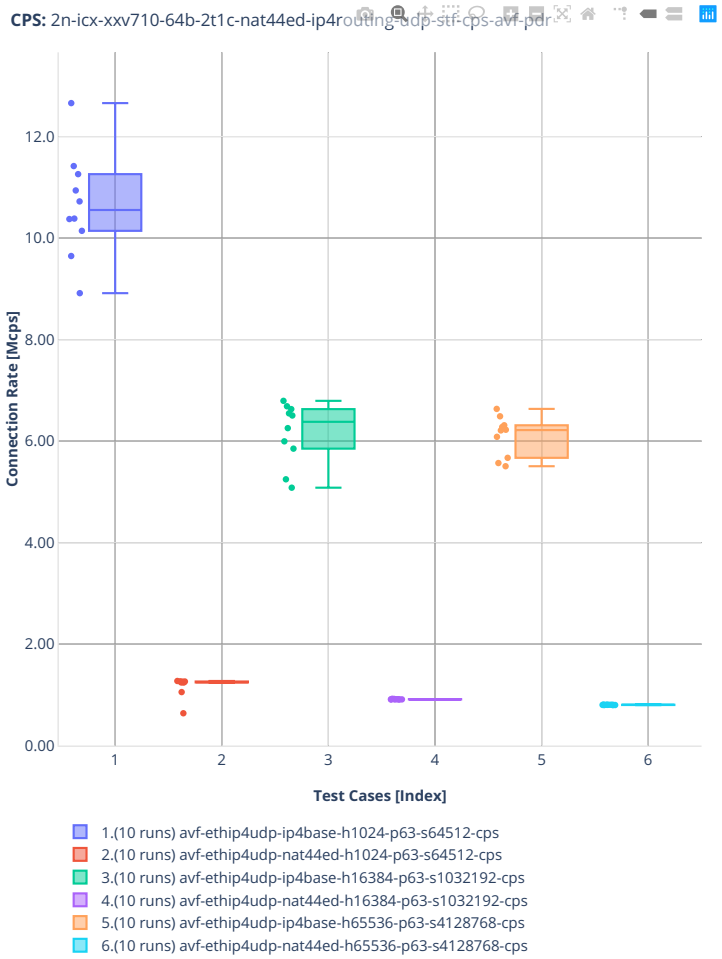

Tput: 2n-clx-xxv710-64b-2t1c-nat44ed-ip4routing-stl-unidir-avf-pdr
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
1.(10 runs) avf-ethip4udp-nat44ed-h1024-p63-s64512-udir

ED UDP CPS

2n-icx-xxv710

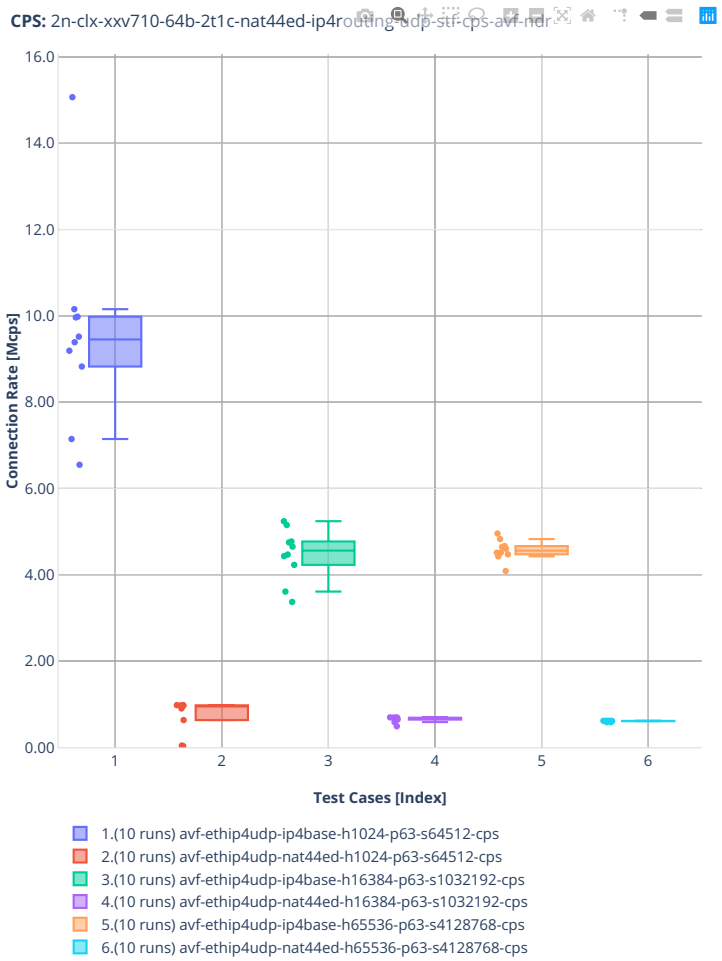
64b-nat44ed-ip4routing-udp-stf-cps-avf

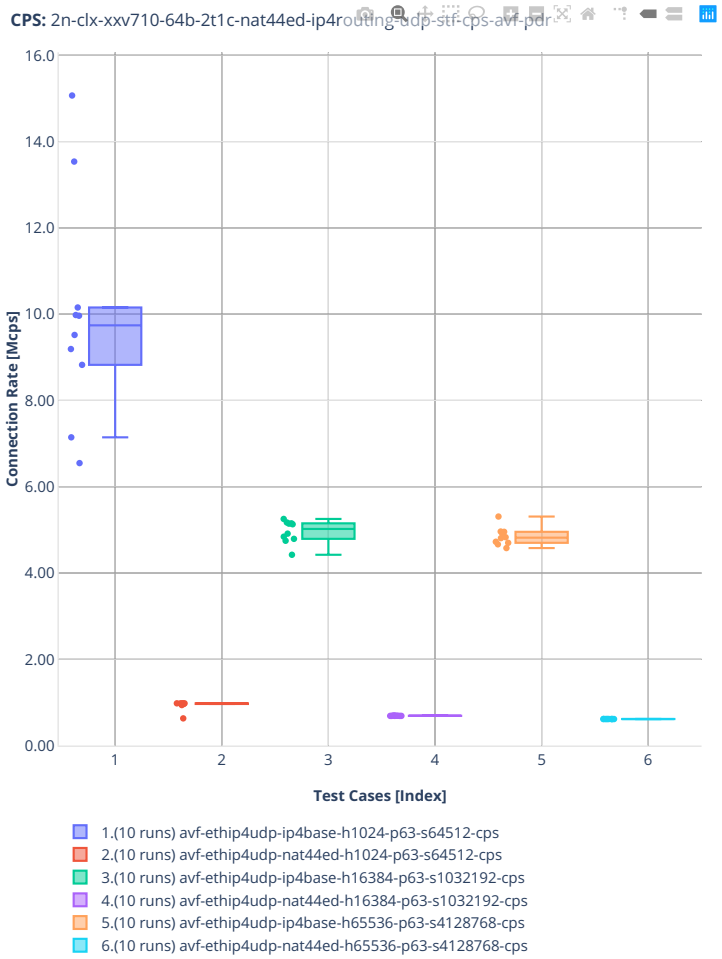



CPS: 2n-icx-xxv710-64b-2t1c-nat44ed-ip4routing-udp-stf-cps-avf-pdr
Connection Rate [Mcps]
12.0
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-h1024-p63-s64512-cps
2.(10 runs) avf-ethip4udp-nat44ed-h1024-p63-s64512-cps
3.(10 runs) avf-ethip4udp-ip4base-h16384-p63-s1032192-cps
4.(10 runs) avf-ethip4udp-nat44ed-h16384-p63-s1032192-cps
5.(10 runs) avf-ethip4udp-ip4base-h65536-p63-s4128768-cps
6.(10 runs) avf-ethip4udp-nat44ed-h65536-p63-s4128768-cps

2n-clx-xxv710

64b-nat44ed-ip4routing-udp-stf-cps-avf

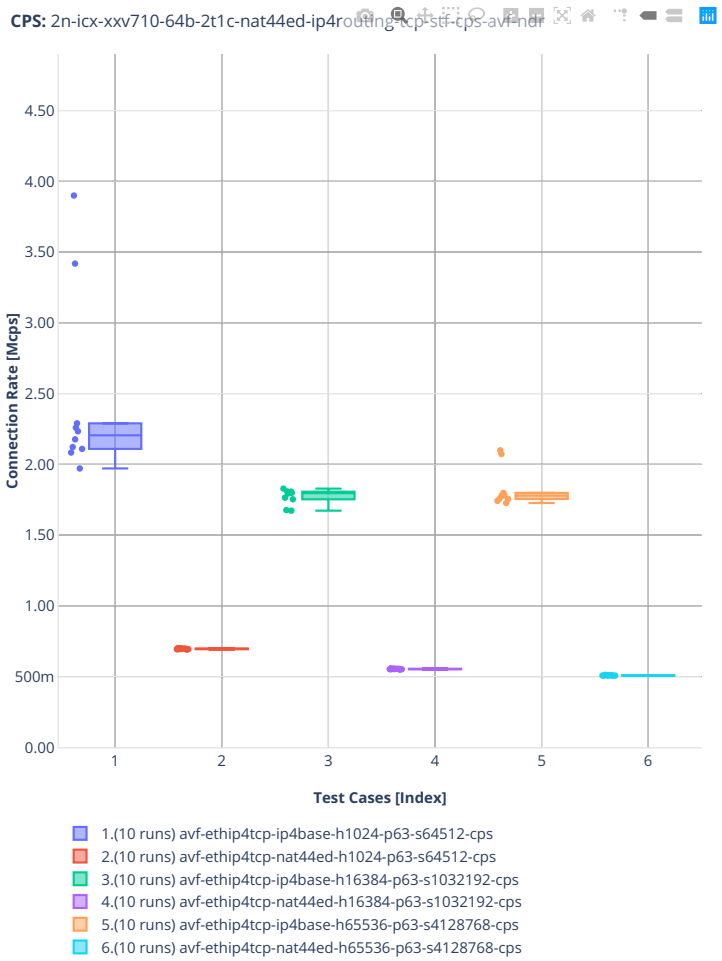


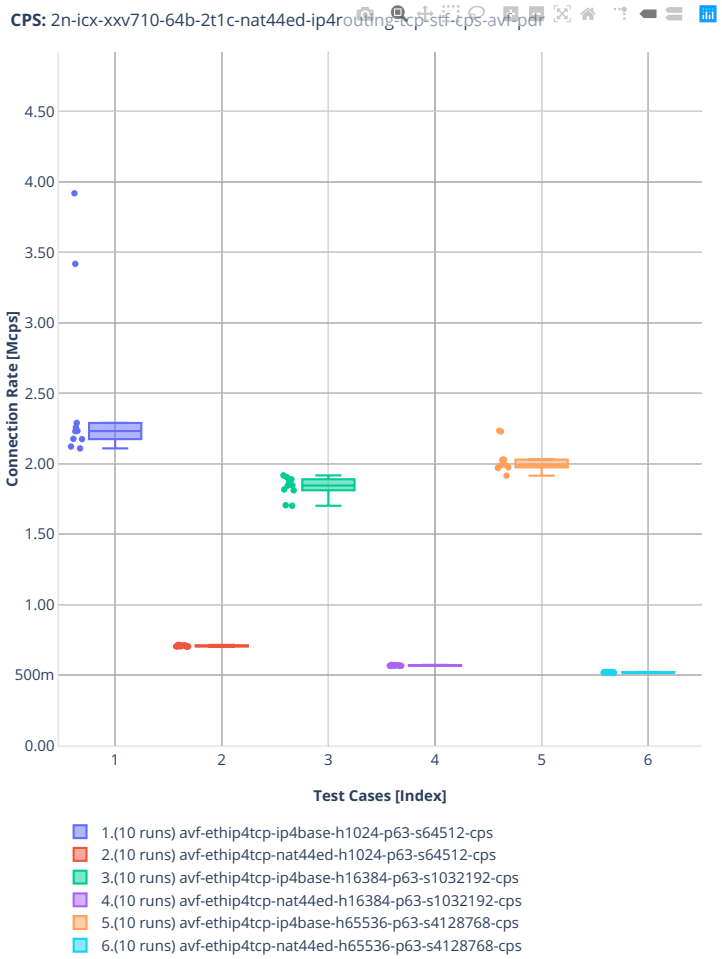

CPS: 2n-clx-xxv710-64b-2t1c-nat44ed-ip4routing-udp-stf-cps-avf-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Connection Rate [Mcps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-h1024-p63-s64512-cps
2.(10 runs) avf-ethip4udp-nat44ed-h1024-p63-s64512-cps
3.(10 runs) avf-ethip4udp-ip4base-h16384-p63-s1032192-cps
4.(10 runs) avf-ethip4udp-nat44ed-h16384-p63-s1032192-cps
5.(10 runs) avf-ethip4udp-ip4base-h65536-p63-s4128768-cps
6.(10 runs) avf-ethip4udp-nat44ed-h65536-p63-s4128768-cps

ED TCP CPS

2n-icx-xxv710

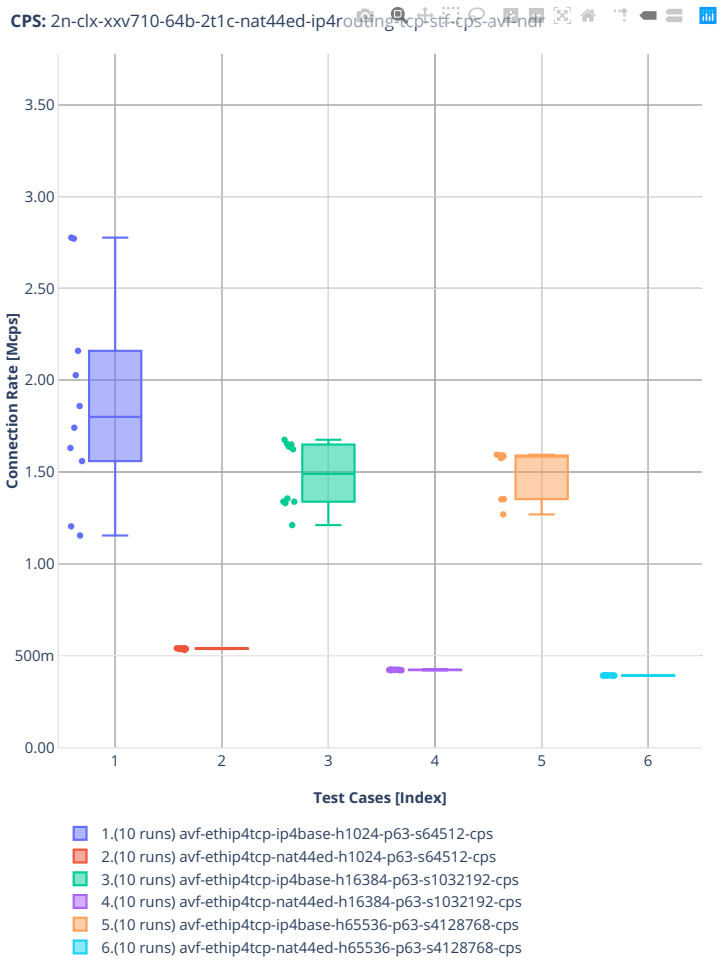
64b-nat44ed-ip4routing-tcp-stf-cps-avf

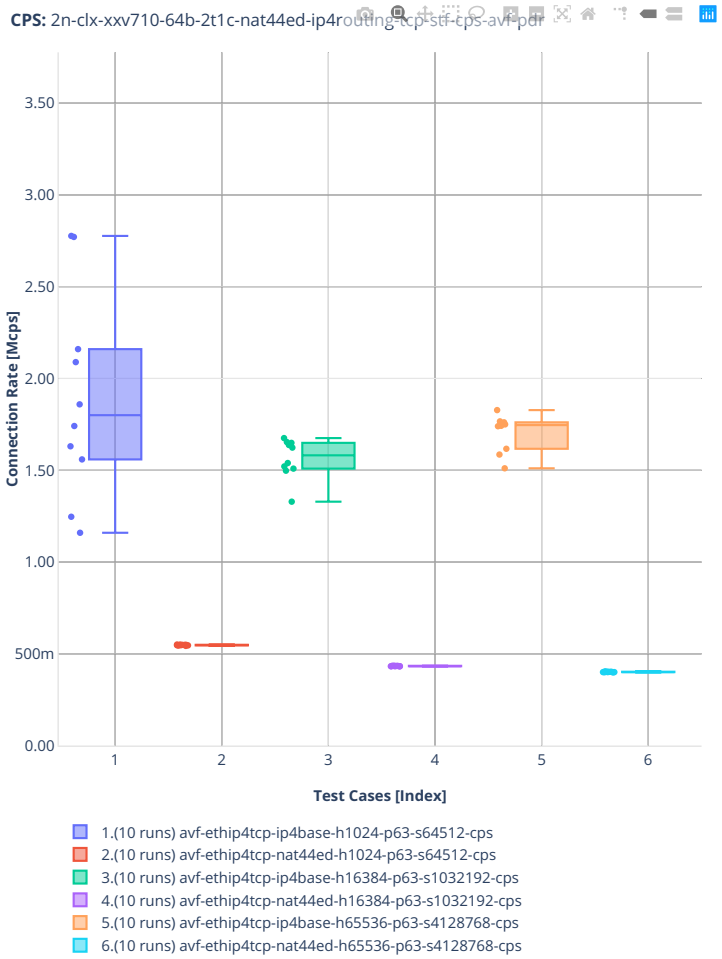



CPS: 2n-icx-xxv710-64b-2t1c-nat44ed-ip4routing-tcp-stf-cps-avf-pdr
Connection Rate [Mcps]
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
500m
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4tcp-ip4base-h1024-p63-s64512-cps
2.(10 runs) avf-ethip4tcp-nat44ed-h1024-p63-s64512-cps
3.(10 runs) avf-ethip4tcp-ip4base-h16384-p63-s1032192-cps
4.(10 runs) avf-ethip4tcp-nat44ed-h16384-p63-s1032192-cps
5.(10 runs) avf-ethip4tcp-ip4base-h65536-p63-s4128768-cps
6.(10 runs) avf-ethip4tcp-nat44ed-h65536-p63-s4128768-cps

2n-clx-xxv710

64b-nat44ed-ip4routing-tcp-stf-cps-avf

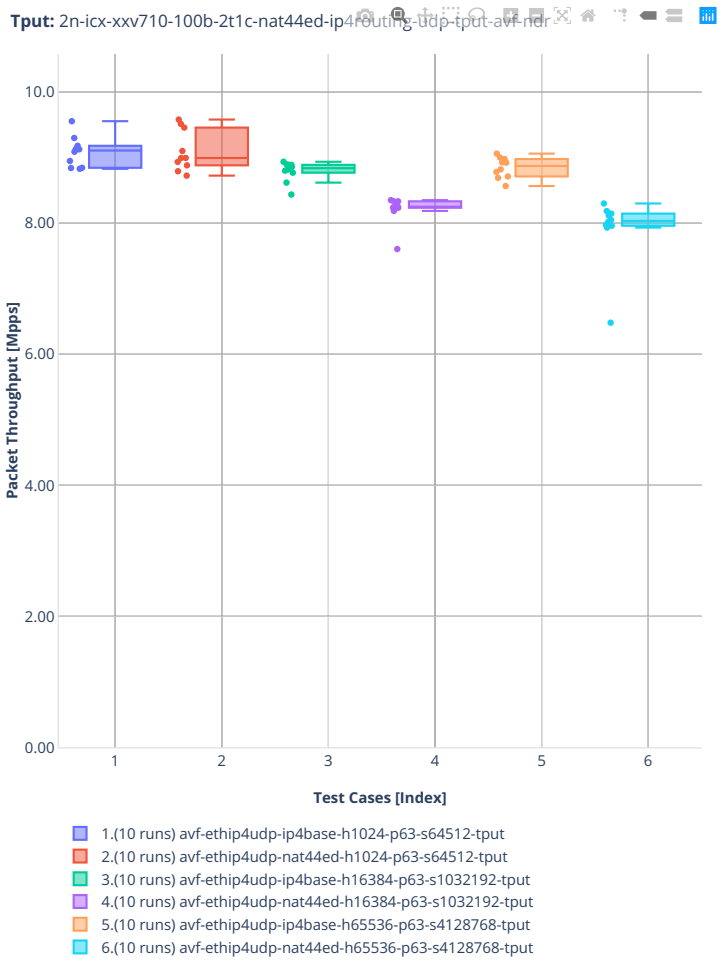


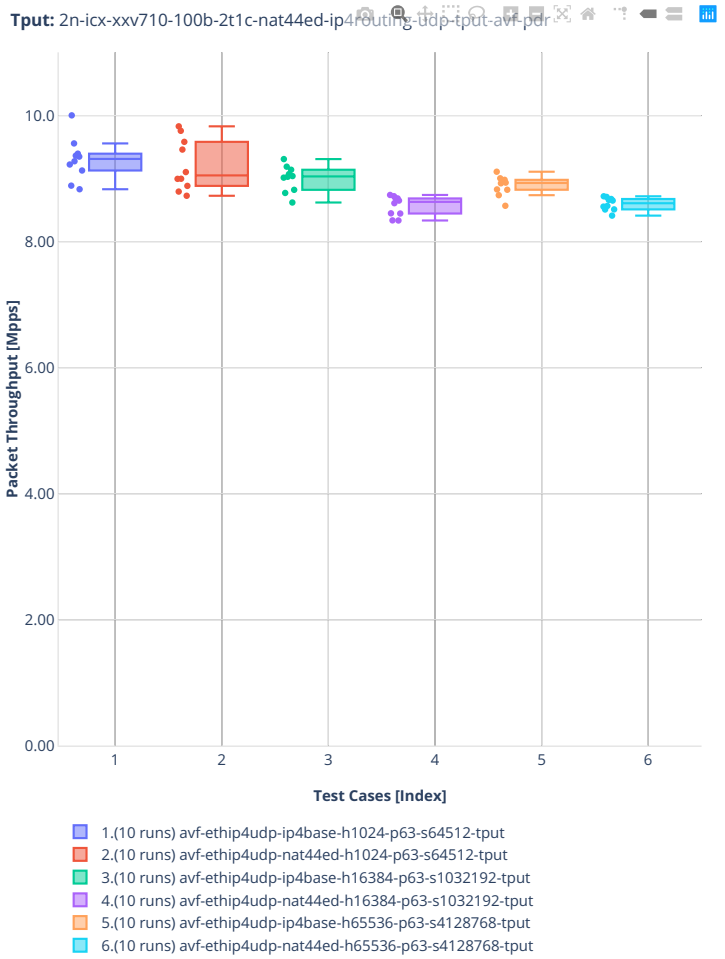

CPS: 2n-clx-xxv710-64b-2t1c-nat44ed-ip4routing-tcp-stf-cps-avf-pdr
Connection Rate [Mcps]
3.50
3.00
2.50
2.00
1.50
1.00
500m
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4tcp-ip4base-h1024-p63-s64512-cps
2.(10 runs) avf-ethip4tcp-nat44ed-h1024-p63-s64512-cps
3.(10 runs) avf-ethip4tcp-ip4base-h16384-p63-s1032192-cps
4.(10 runs) avf-ethip4tcp-nat44ed-h16384-p63-s1032192-cps
5.(10 runs) avf-ethip4tcp-ip4base-h65536-p63-s4128768-cps
6.(10 runs) avf-ethip4tcp-nat44ed-h65536-p63-s4128768-cps

ED UDP TPUT

2n-icx-xxv710

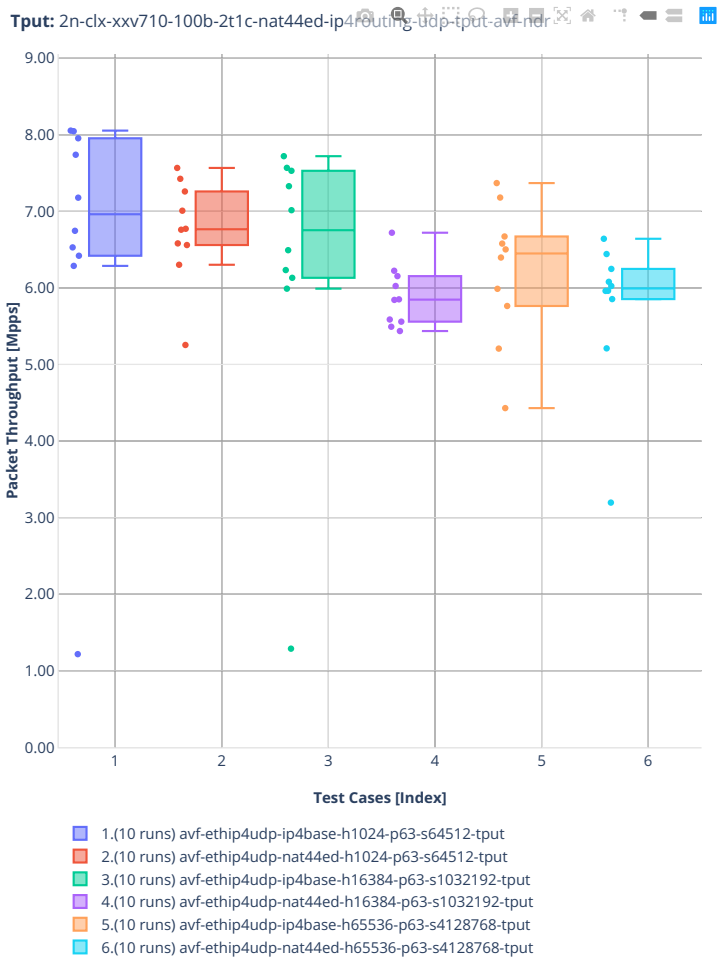
100b-nat44ed-ip4routing-udp-tput-avf

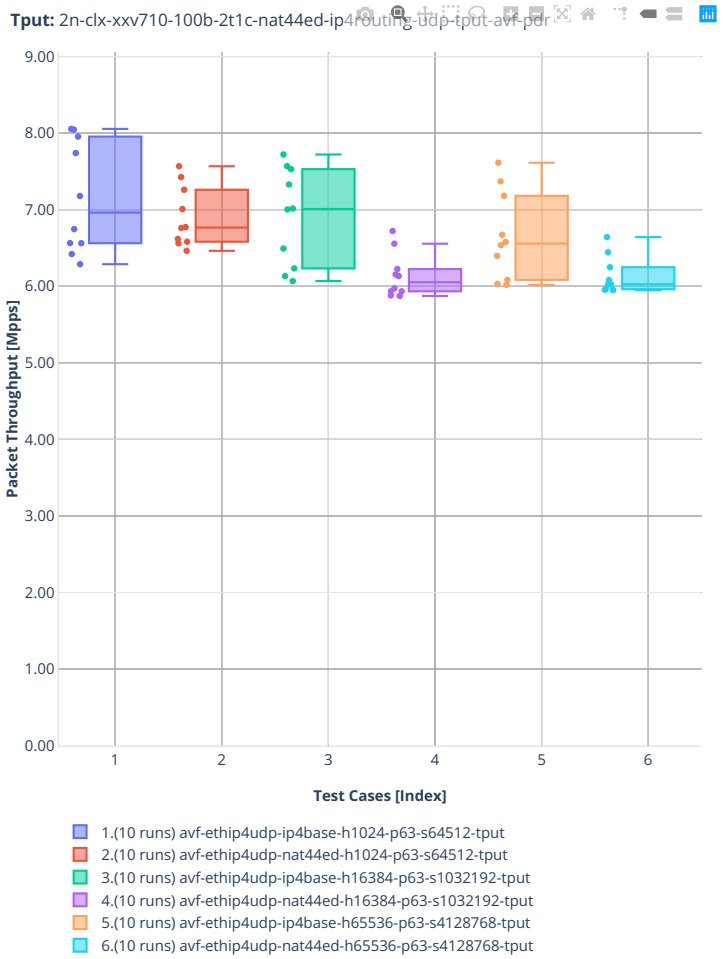



Tput: 2n-icx-xxv710-100b-2t1c-nat44ed-ip4routing-udp-tput-avf-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-h1024-p63-s64512-tput
2.(10 runs) avf-ethip4udp-nat44ed-h1024-p63-s64512-tput
3.(10 runs) avf-ethip4udp-ip4base-h16384-p63-s1032192-tput
4.(10 runs) avf-ethip4udp-nat44ed-h16384-p63-s1032192-tput
5.(10 runs) avf-ethip4udp-ip4base-h65536-p63-s4128768-tput
6.(10 runs) avf-ethip4udp-nat44ed-h65536-p63-s4128768-tput

2n-clx-xxv710

100b-nat44ed-ip4routing-udp-tput-avf

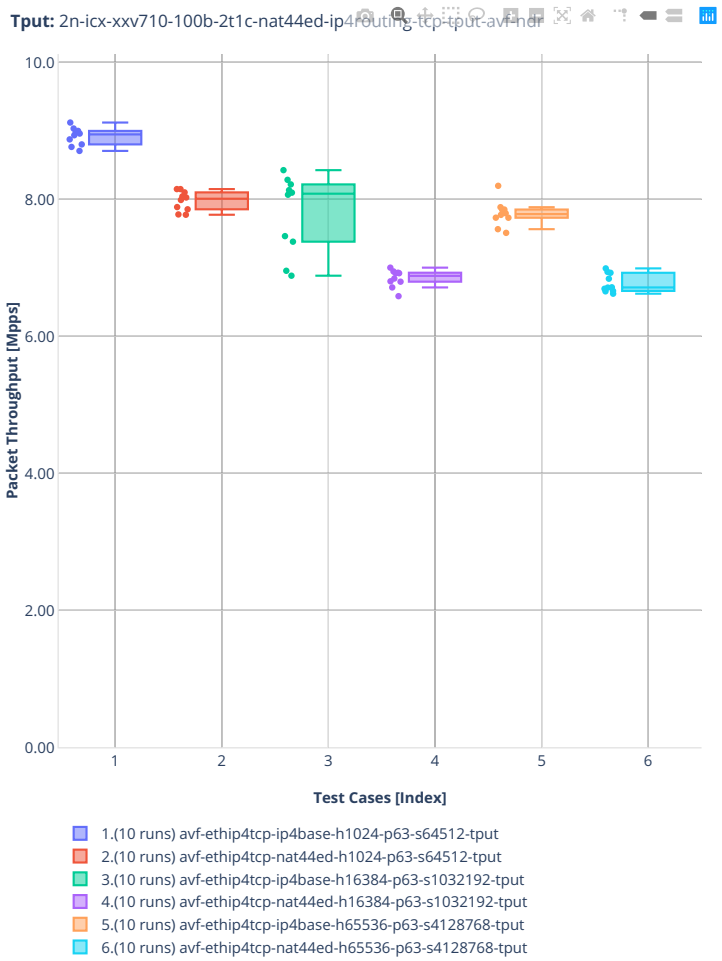


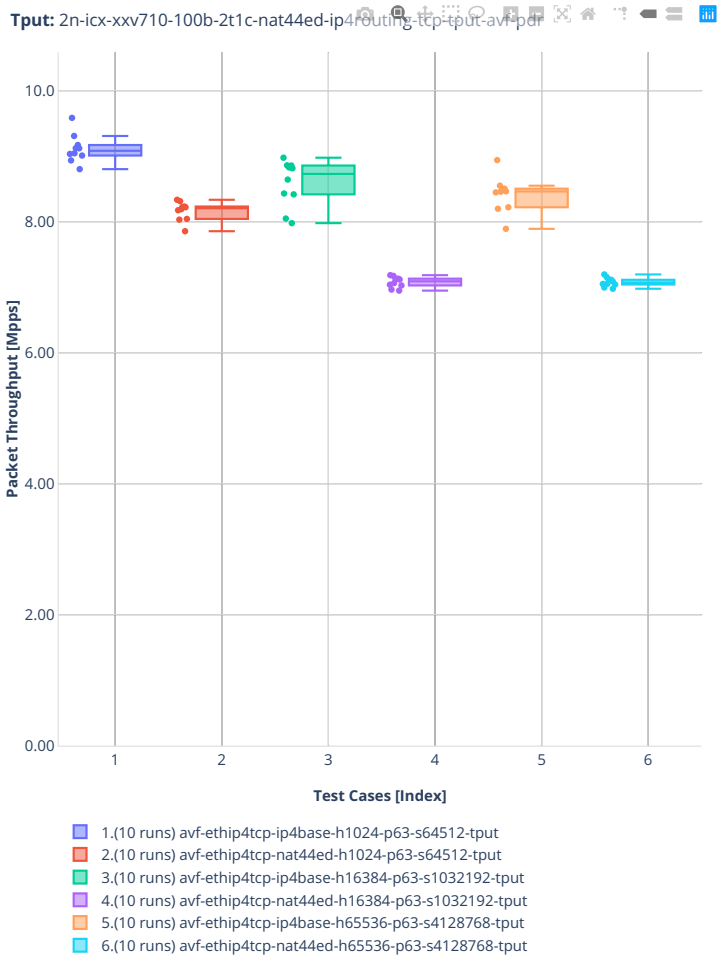

Tput: 2n-clx-xxv710-100b-2t1c-nat44ed-ip4routing-udp-tput-avf-pdr
Packet Throughput [Mpps]
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4udp-ip4base-h1024-p63-s64512-tput
2.(10 runs) avf-ethip4udp-nat44ed-h1024-p63-s64512-tput
3.(10 runs) avf-ethip4udp-ip4base-h16384-p63-s1032192-tput
4.(10 runs) avf-ethip4udp-nat44ed-h16384-p63-s1032192-tput
5.(10 runs) avf-ethip4udp-ip4base-h65536-p63-s4128768-tput
6.(10 runs) avf-ethip4udp-nat44ed-h65536-p63-s4128768-tput

ED TCP TPUT

2n-icx-xxv710

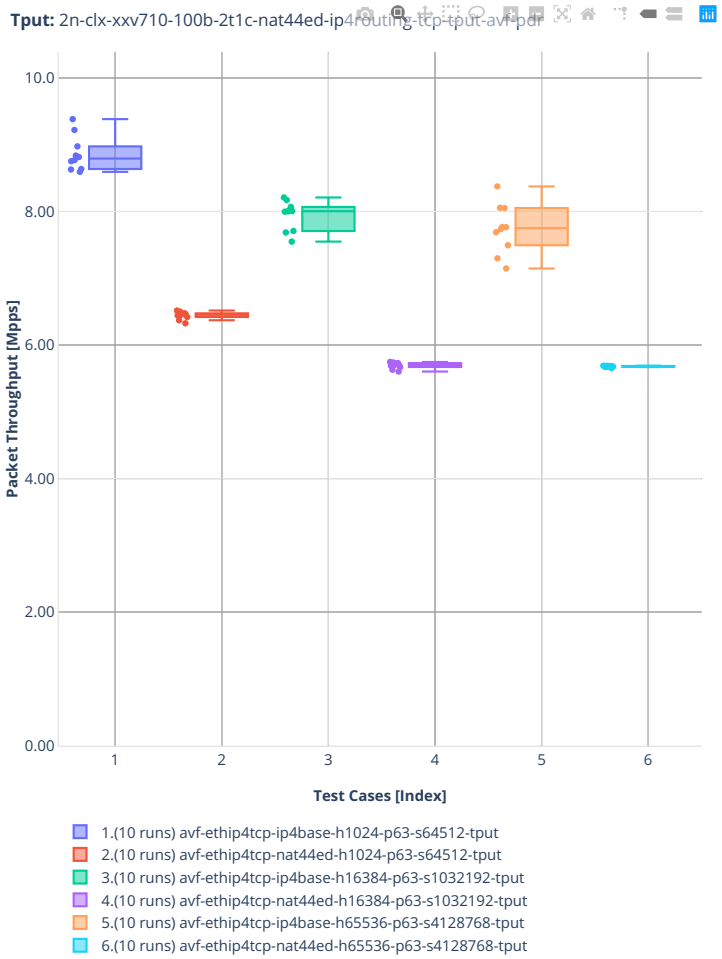
100b-nat44ed-ip4routing-tcp-tput-avf




Tput: 2n-icx-xxv710-100b-2t1c-nat44ed-ip4routing-tcp-tput-avf-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4tcp-ip4base-h1024-p63-s64512-tput
2.(10 runs) avf-ethip4tcp-nat44ed-h1024-p63-s64512-tput
3.(10 runs) avf-ethip4tcp-ip4base-h16384-p63-s1032192-tput
4.(10 runs) avf-ethip4tcp-nat44ed-h16384-p63-s1032192-tput
5.(10 runs) avf-ethip4tcp-ip4base-h65536-p63-s4128768-tput
6.(10 runs) avf-ethip4tcp-nat44ed-h65536-p63-s4128768-tput

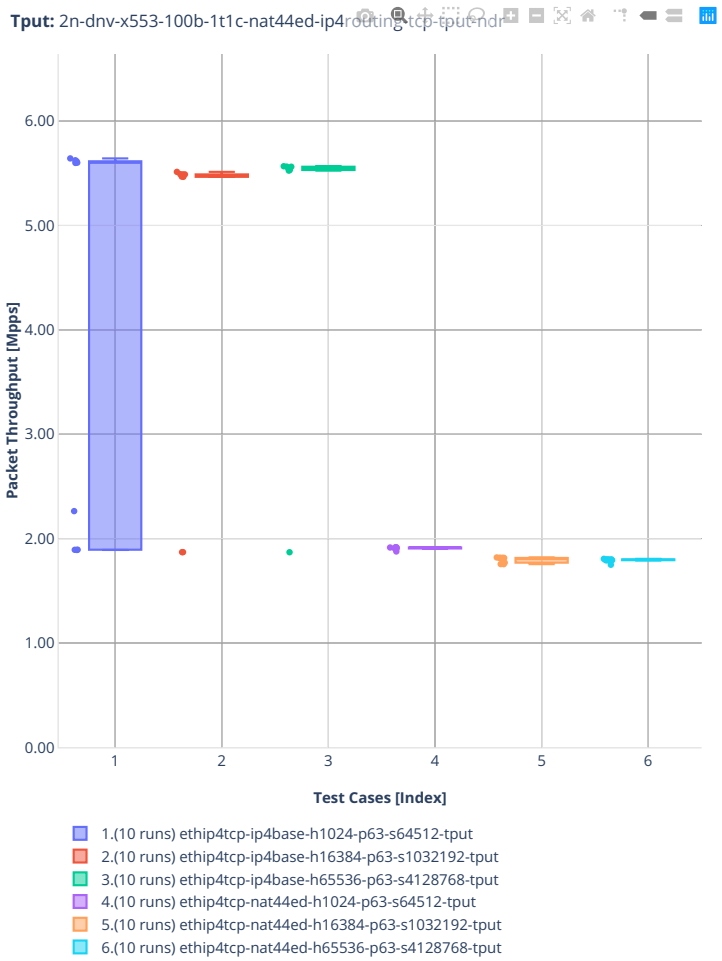
2n-clx-xxv710

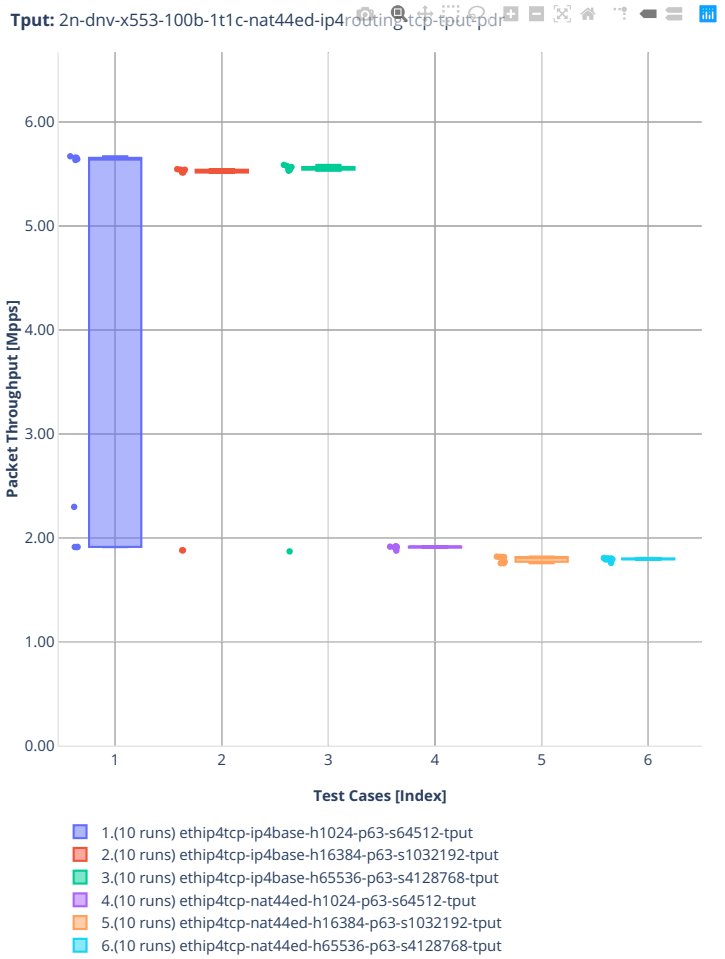
100b-nat44ed-ip4routing-tcp-tput-avf


Tput: 2n-clx-xxv710-100b-2t1c-nat44ed-ip4routing-tcp-tput-avf-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-ethip4tcp-ip4base-h1024-p63-s64512-tput
2.(10 runs) avf-ethip4tcp-nat44ed-h1024-p63-s64512-tput
3.(10 runs) avf-ethip4tcp-ip4base-h16384-p63-s1032192-tput
4.(10 runs) avf-ethip4tcp-nat44ed-h16384-p63-s1032192-tput
5.(10 runs) avf-ethip4tcp-ip4base-h65536-p63-s4128768-tput
6.(10 runs) avf-ethip4tcp-nat44ed-h65536-p63-s4128768-tput

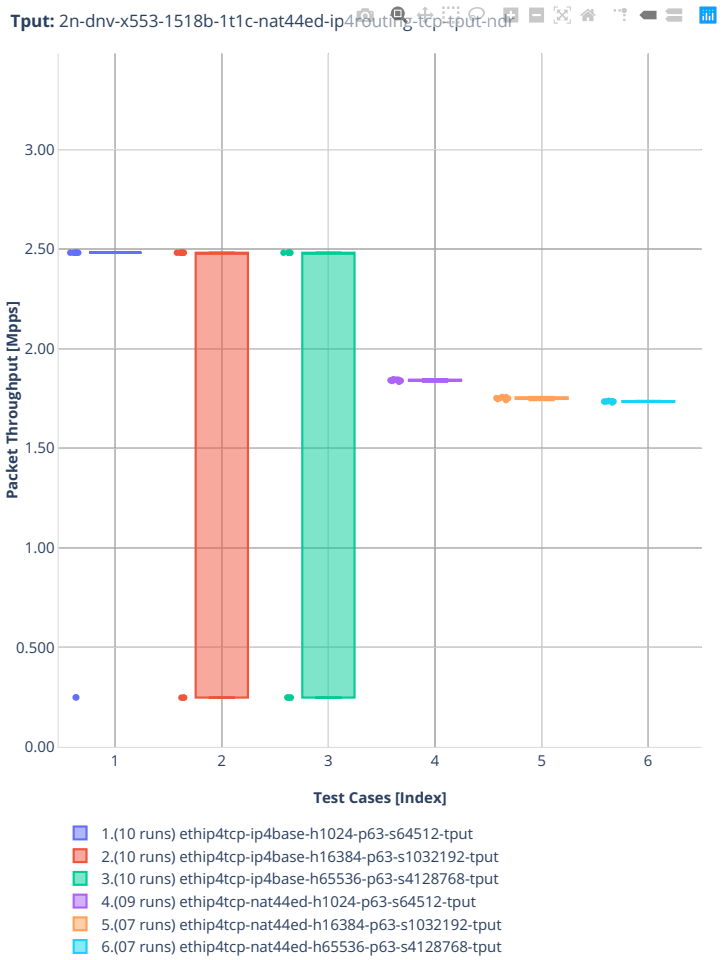
2n-dnv-x553

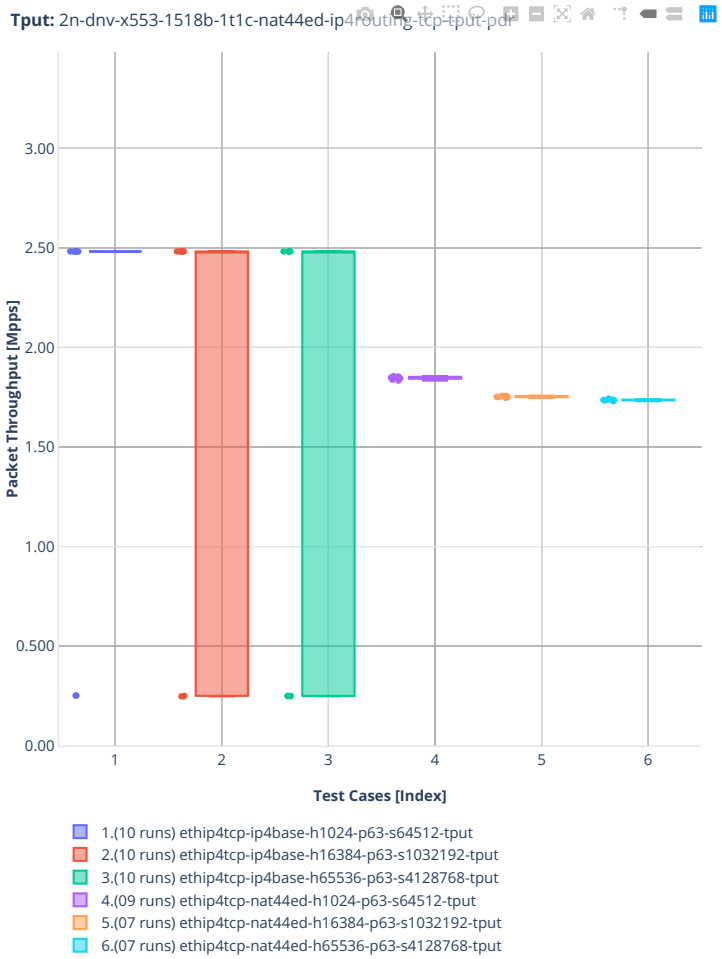
100b-nat44ed-ip4routing-tcp-tput




Tput: 2n-dnv-x553-100b-1t1c-nat44ed-ip4routing-tcp-tput-pdr
Packet Throughput [Mpps]
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4tcp-ip4base-h1024-p63-s64512-tput
2.(10 runs) ethip4tcp-ip4base-h16384-p63-s1032192-tput
3.(10 runs) ethip4tcp-ip4base-h65536-p63-s4128768-tput
4.(10 runs) ethip4tcp-nat44ed-h1024-p63-s64512-tput
5.(10 runs) ethip4tcp-nat44ed-h16384-p63-s1032192-tput
6.(10 runs) ethip4tcp-nat44ed-h65536-p63-s4128768-tput

1518b-nat44ed-ip4routing-tcp-tput




Tput: 2n-dnv-x553-1518b-1t1c-nat44ed-ip4routing-tcp-tput-pdr
Packet Throughput [Mpps]
3.00
2.50
2.00
1.50
1.00
0.500
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4tcp-ip4base-h1024-p63-s64512-tput
2.(10 runs) ethip4tcp-ip4base-h16384-p63-s1032192-tput
3.(10 runs) ethip4tcp-ip4base-h65536-p63-s4128768-tput
4.(09 runs) ethip4tcp-nat44ed-h1024-p63-s64512-tput
5.(07 runs) ethip4tcp-nat44ed-h16384-p63-s1032192-tput
6.(07 runs) ethip4tcp-nat44ed-h65536-p63-s4128768-tput

2.3.7 KVM VMs vhost-user

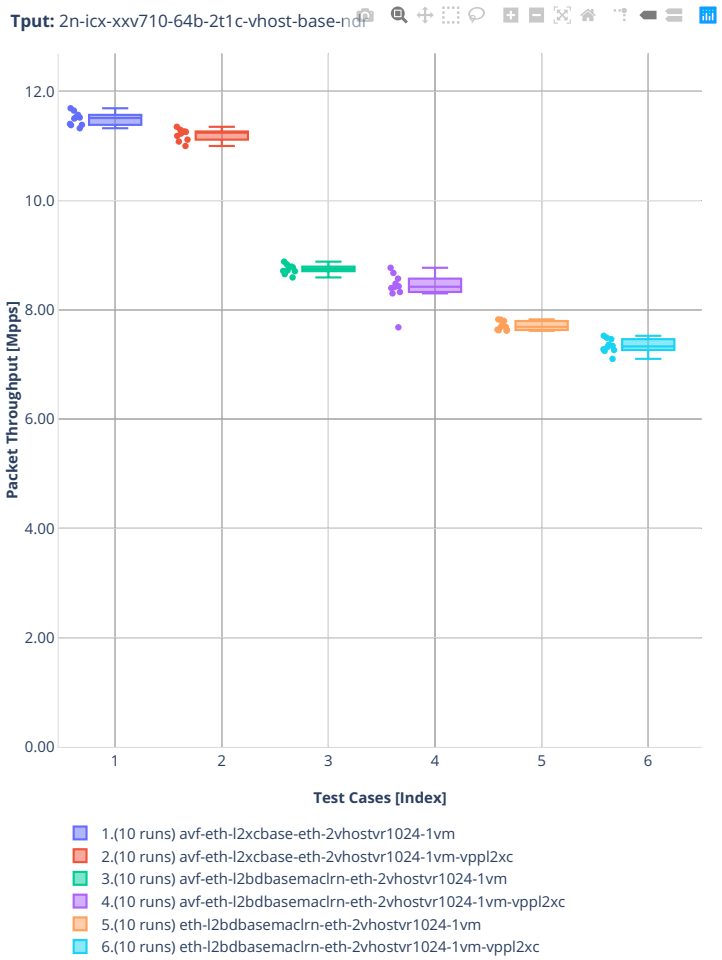
Following sections include summary graphs of VPP Phy-to-VM(s)-to-Phy performance with VM virtio and VPP vhost-user virtual interfaces, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

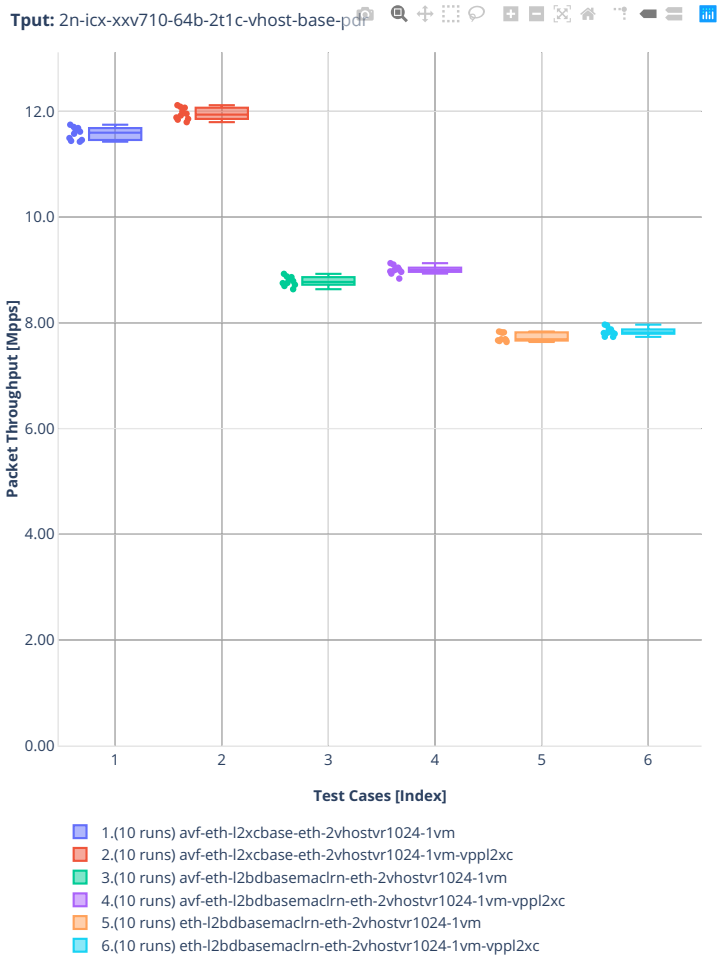
CSIT source code for the test cases used for plots can be found in CSIT git repository[120].

[120] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls2210

2n-icx-xxv710

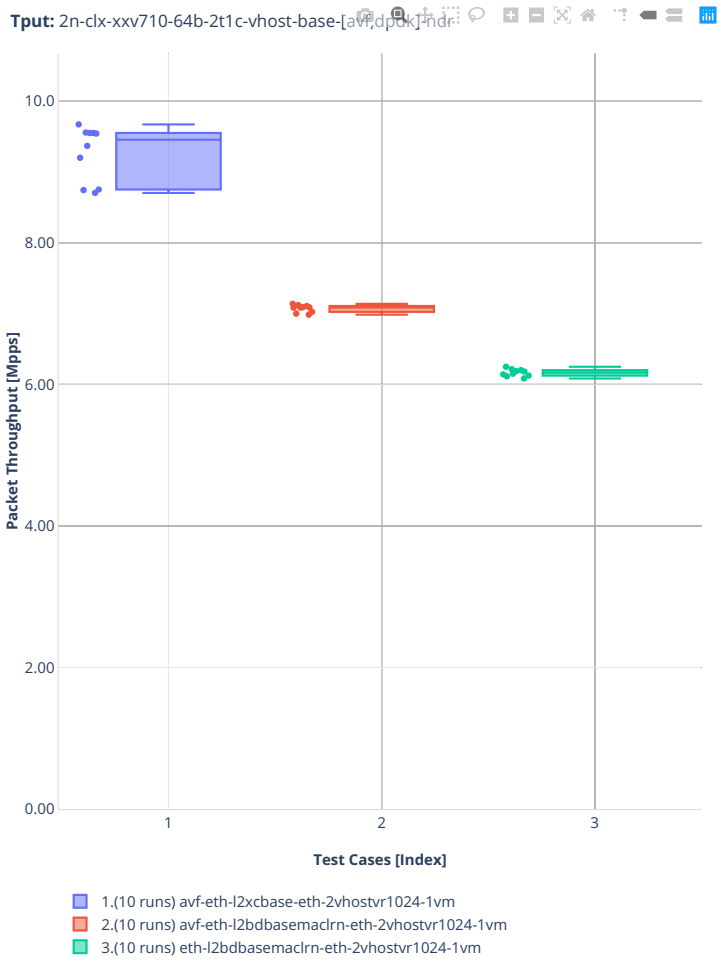
64b-2t1c-vhost-base

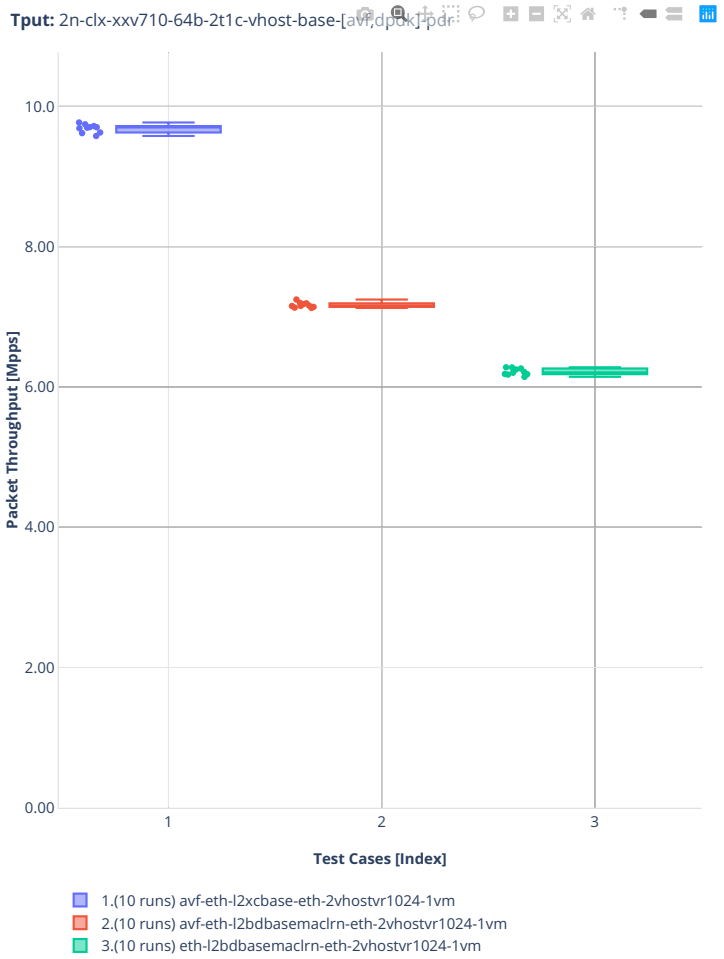



Tput: 2n-icx-xxv710-64b-2t1c-vhost-base-pdr
Packet Throughput [Mpps]
12.0
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2vhostvr1024-1vm
2.(10 runs) avf-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
3.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
4.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
5.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
6.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

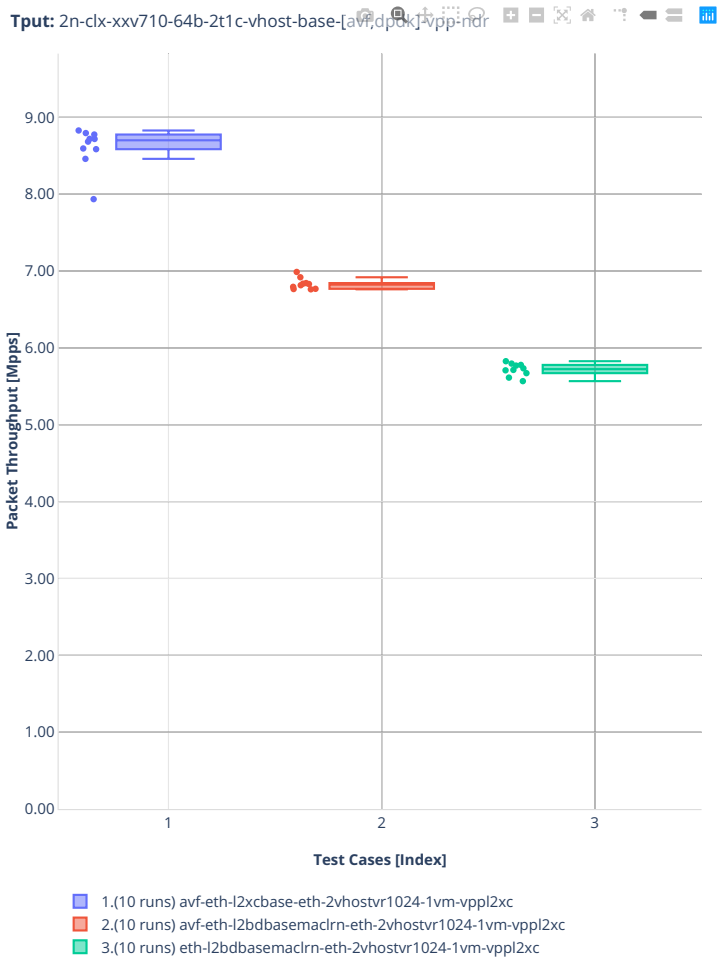
2n-clx-xxv710

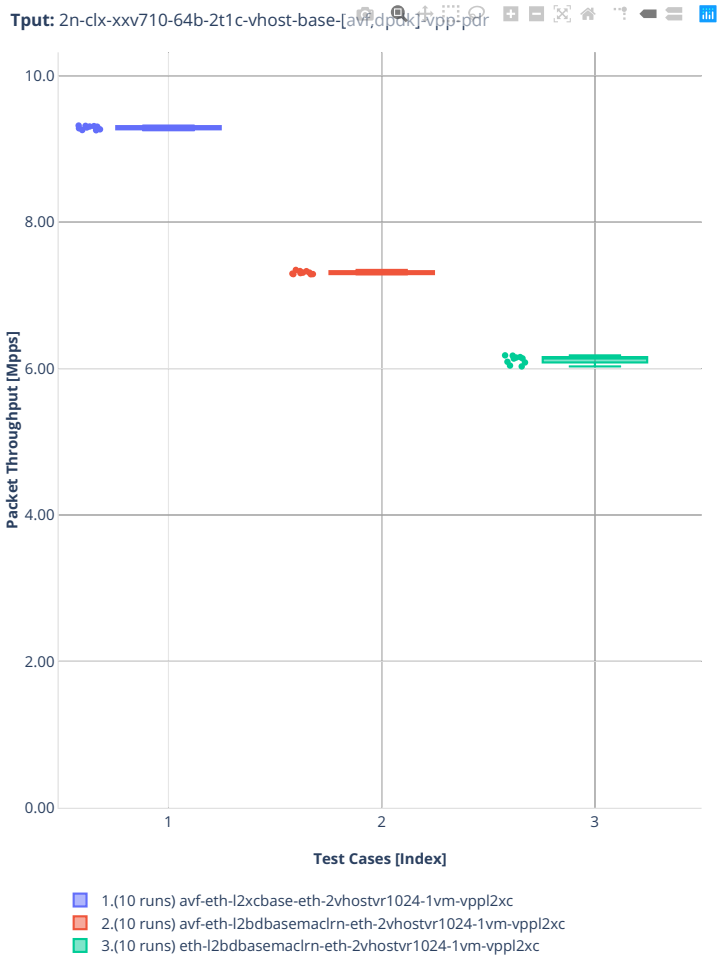
64b-2t1c-vhost-base-testpmd




Tput: 2n-clx-xxv710-64b-2t1c-vhost-base-[avf,dpdk]-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2vhostvr1024-1vm
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
3.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm

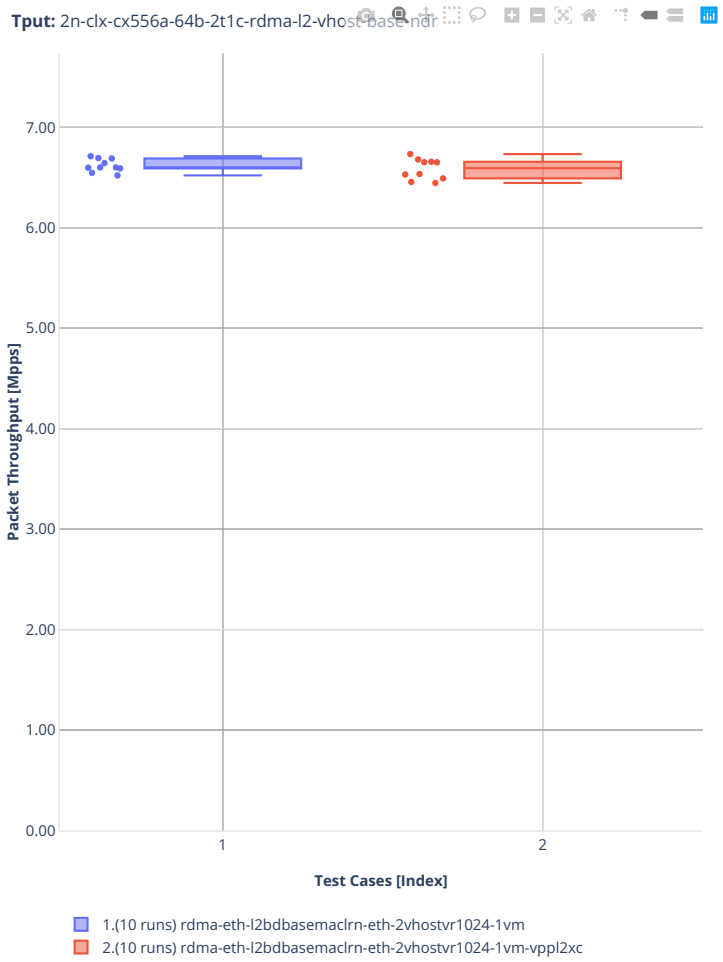
64b-2t1c-vhost-base-vpp

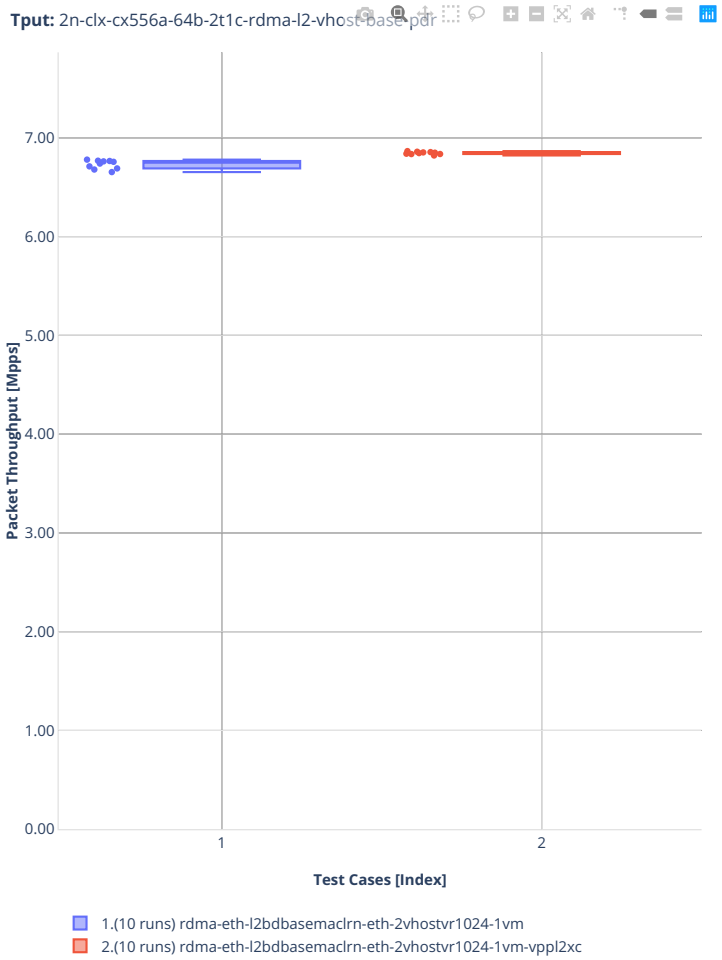



Tput: 2n-clx-xxv710-64b-2t1c-vhost-base-[avf,dpdk]-vpp-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
3.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2n-clx-cx556a

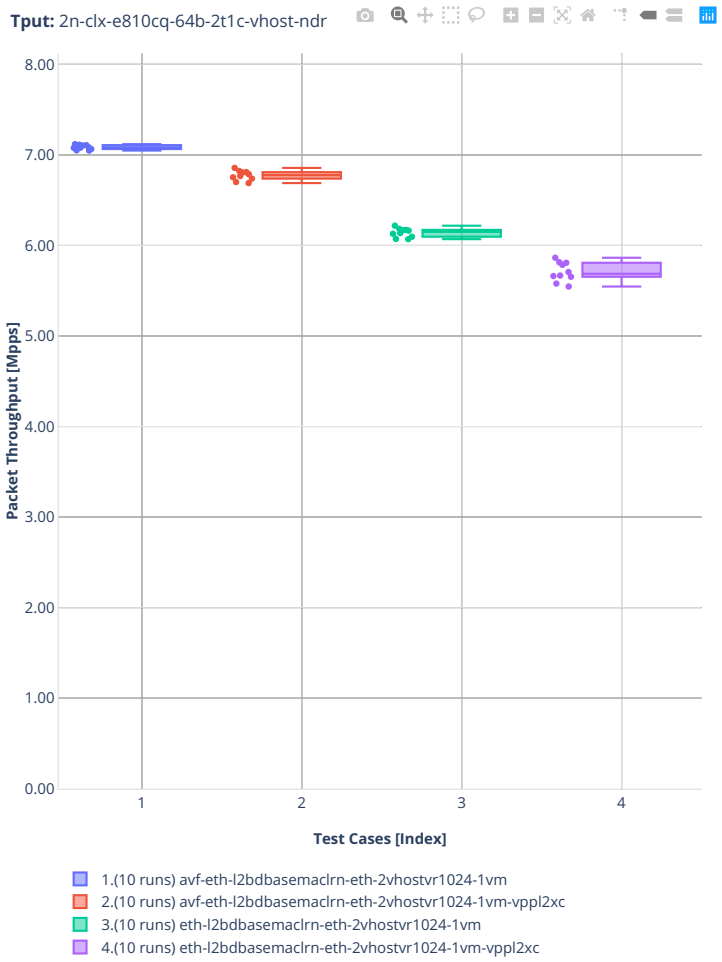
64b-2t1c-vhost-base-rdma-core

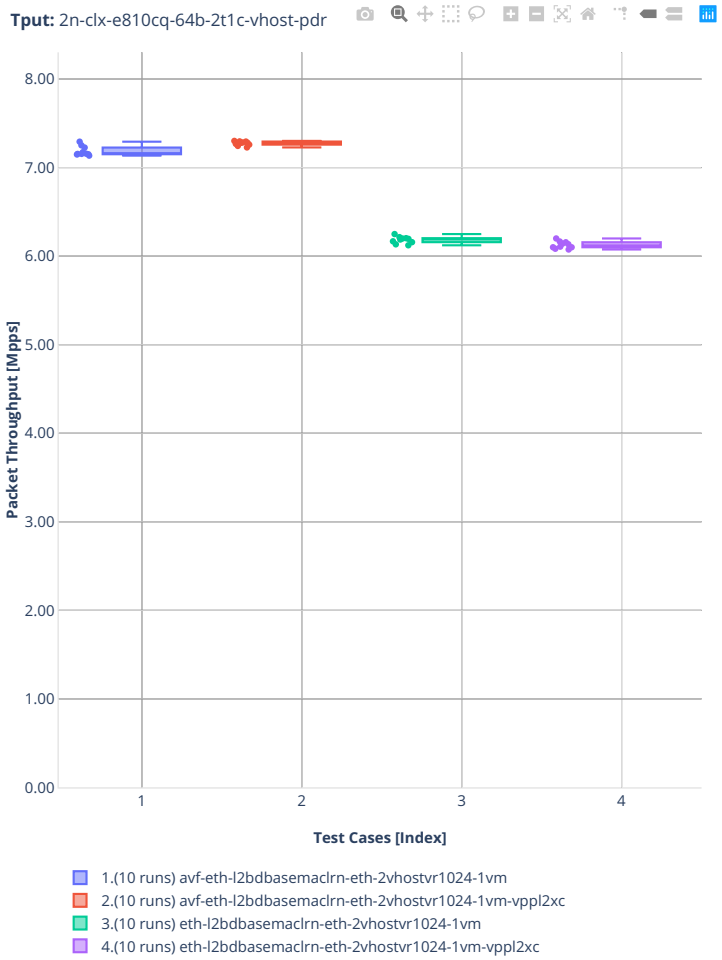



Tput: 2n-clx-cx556a-64b-2t1c-rdma-l2-vhost-base-pdr
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
2.(10 runs) rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2n-clx-e810cq

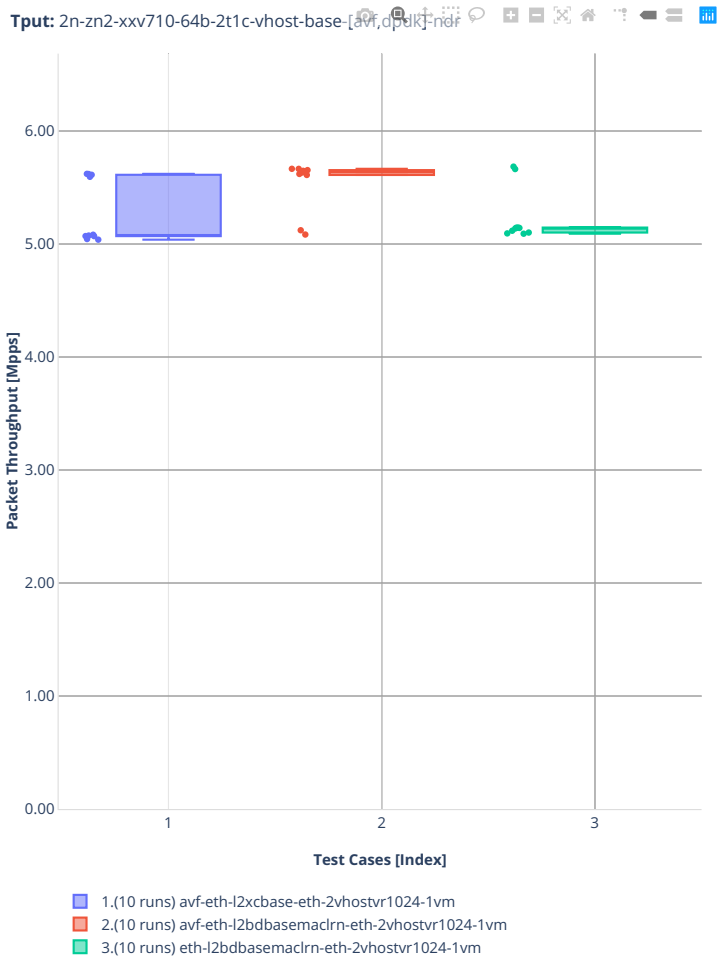
64b-2t1c-vhost

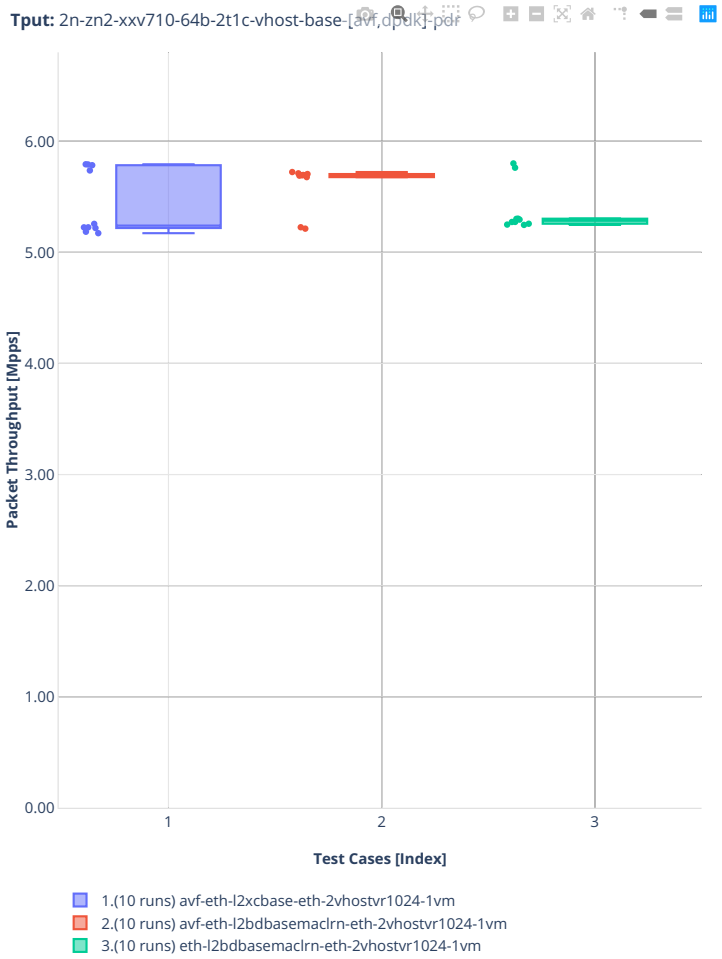



Tput: 2n-clx-e810cq-64b-2t1c-vhost-pdr
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
3.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
4.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

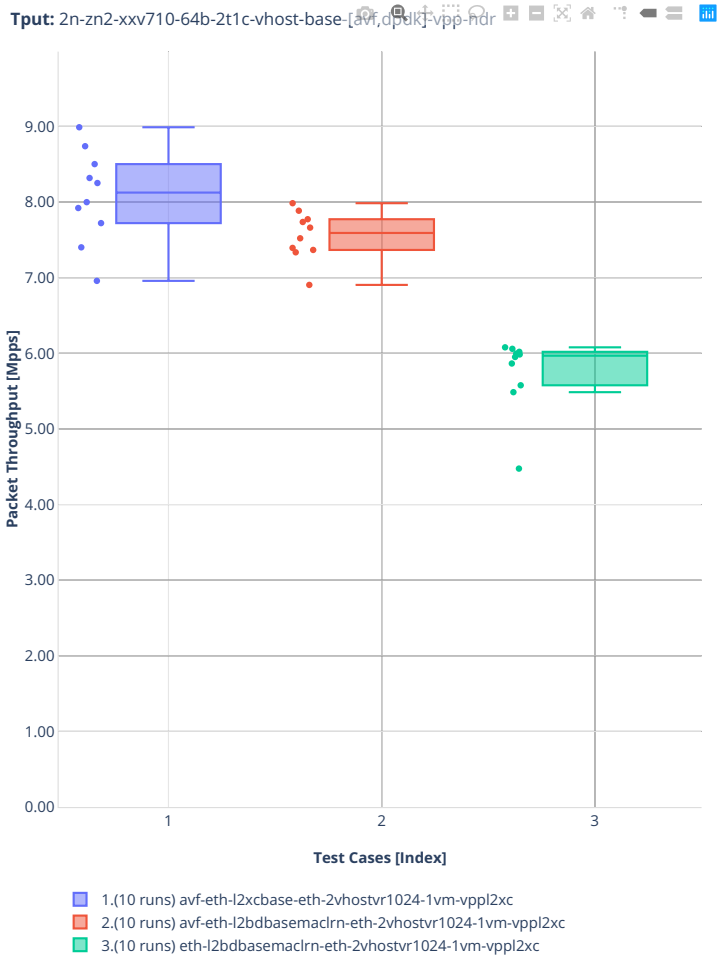
2n-zn2-xxv710

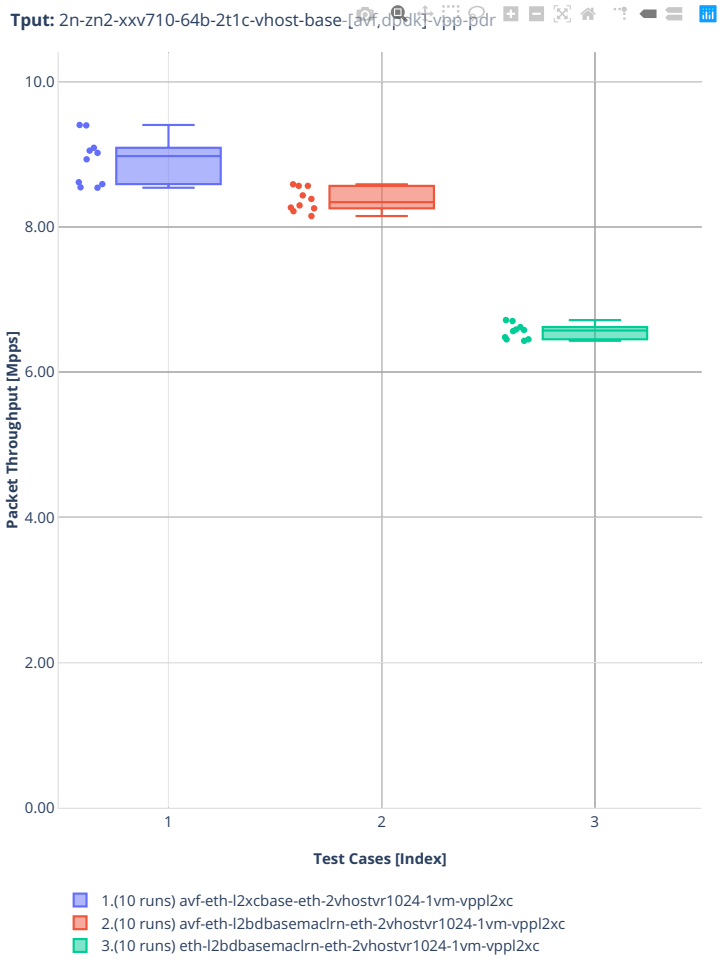
64b-2t1c-vhost-base-testpmd




Tput: 2n-zn2-xxv710-64b-2t1c-vhost-base-[avf,dpdk]-pdr
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2vhostvr1024-1vm
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
3.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm

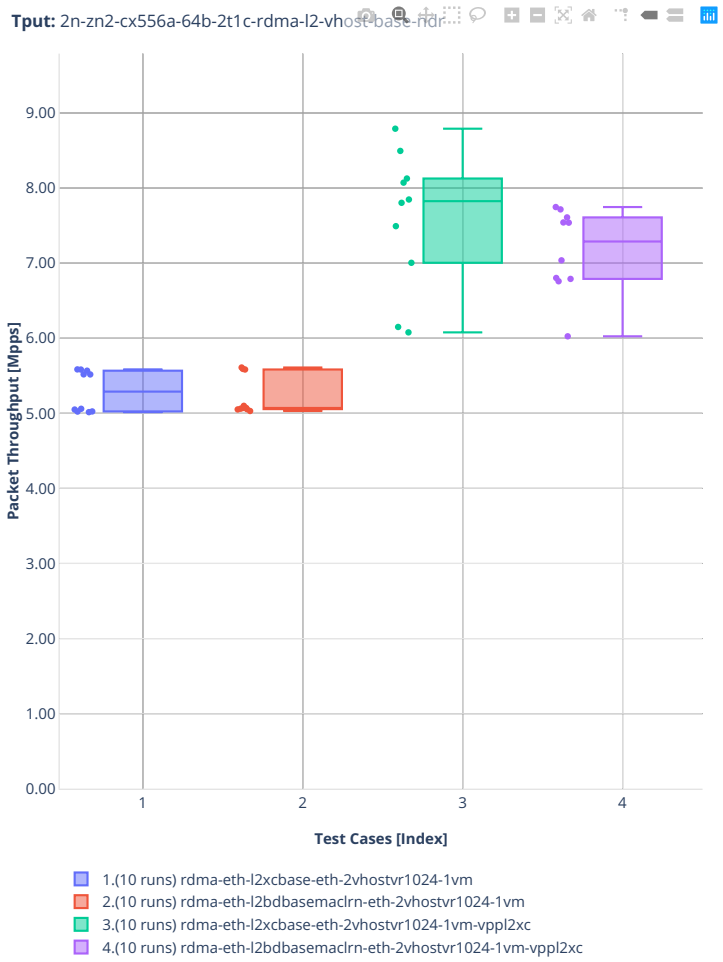
64b-2t1c-vhost-base-vpp

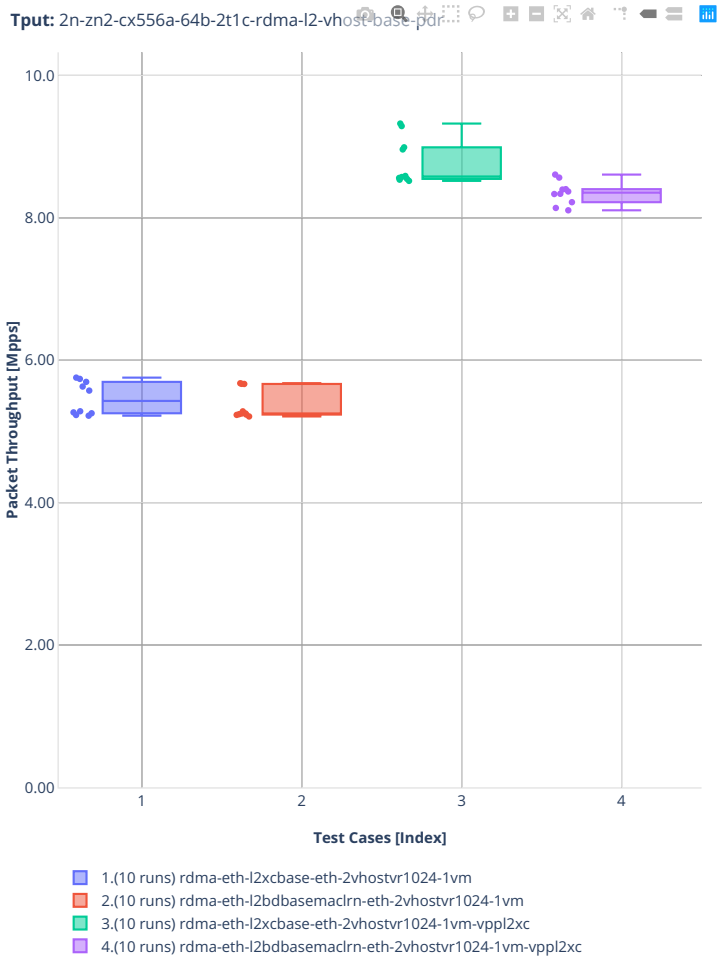



Tput: 2n-zn2-xxv710-64b-2t1c-vhost-base-[avf,dpdk]-vpp-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
3.(10 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2n-zn2-cx556a

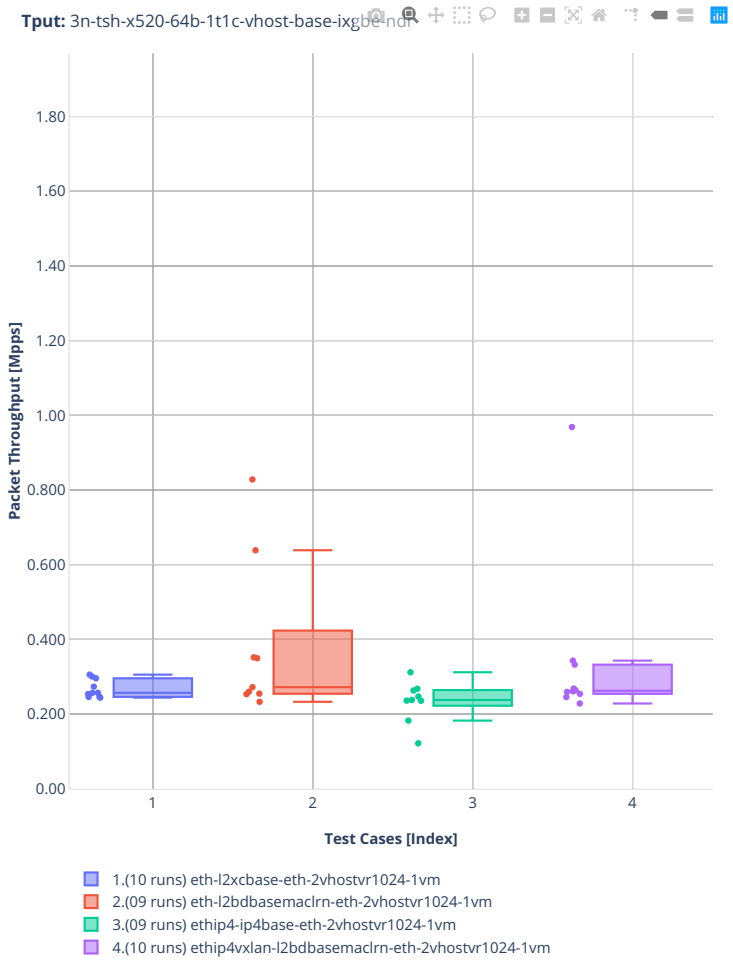
64b-2t1c-vhost-base-rdma-core

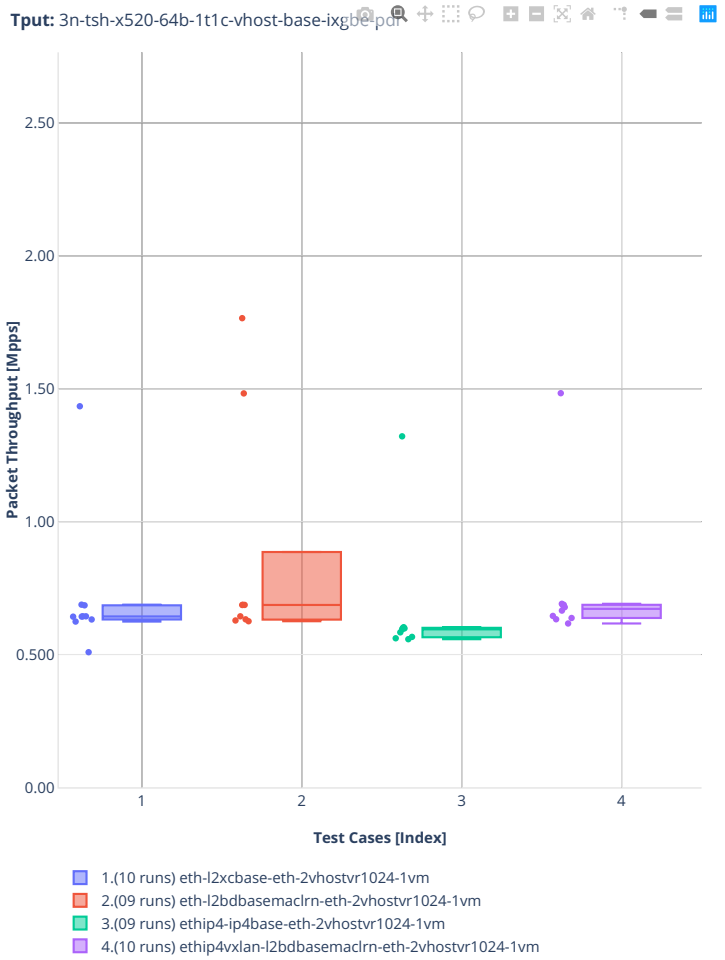



Tput: 2n-zn2-cx556a-64b-2t1c-rdma-l2-vhost-base-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) rdma-eth-l2xcbase-eth-2vhostvr1024-1vm
2.(10 runs) rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
3.(10 runs) rdma-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
4.(10 runs) rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

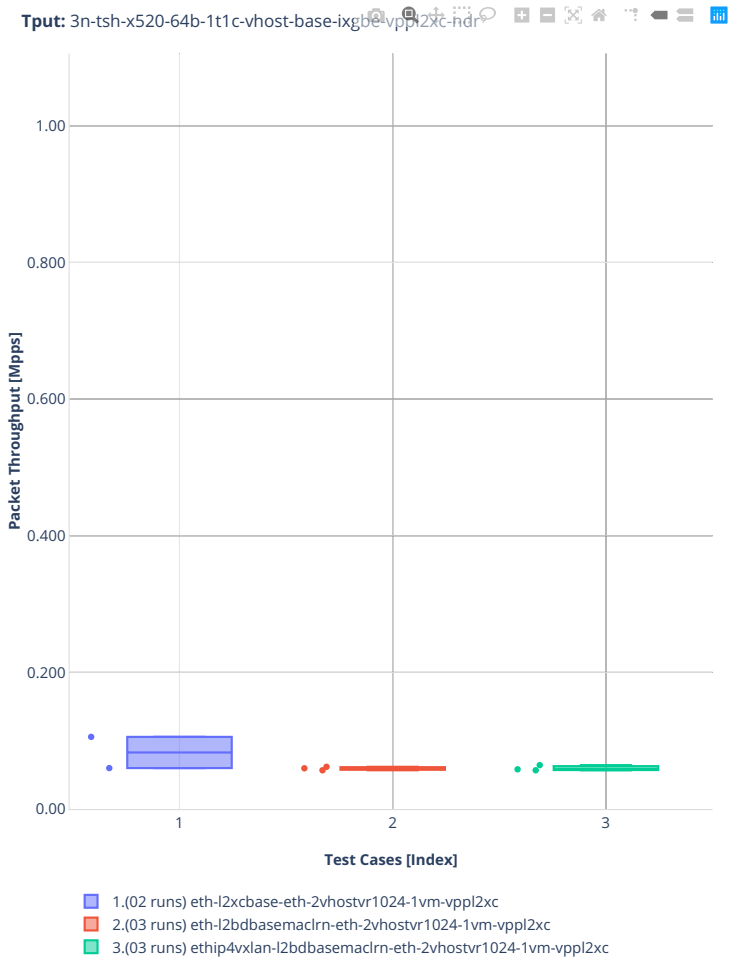
3n-tsh-x520

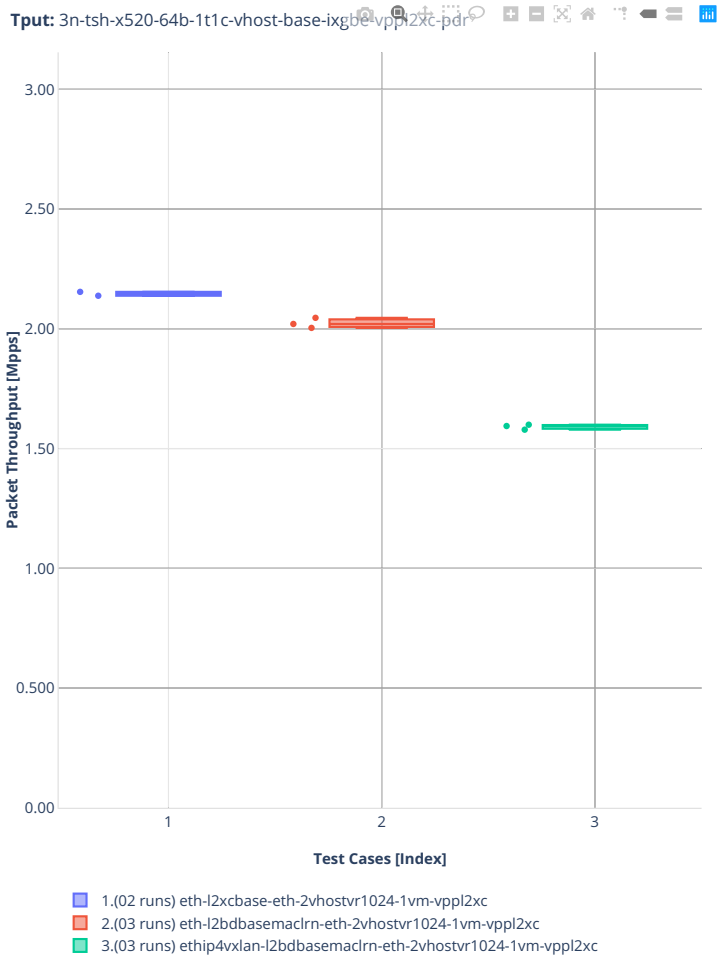
64b-1t1c-vhost-base-ixgbe




Tput: 3n-tsh-x520-64b-1t1c-vhost-base-ixgbe-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(10 runs) eth-l2xcbase-eth-2vhostvr1024-1vm
2.(09 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
3.(09 runs) ethip4-ip4base-eth-2vhostvr1024-1vm
4.(10 runs) ethip4vxlan-l2bdbasemaclrn-eth-2vhostvr1024-1vm

64b-1t1c-vhost-base-ixgbe-vppl2xc




Tput: 3n-tsh-x520-64b-1t1c-vhost-base-ixgbe-vppl2xc-pdr
3.00
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(02 runs) eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
2.(03 runs) eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
3.(03 runs) ethip4vxlan-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2.3.8 LXC/DRC Container Memif

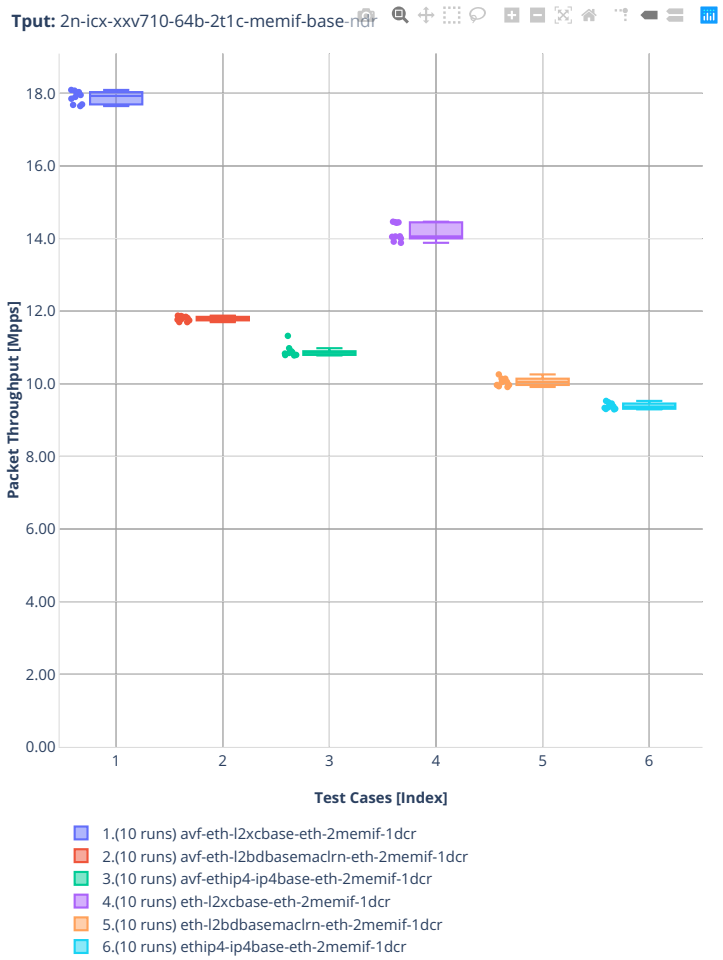
Following sections include summary graphs of VPP Phy-to-Phy performance with Container memif Connections, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

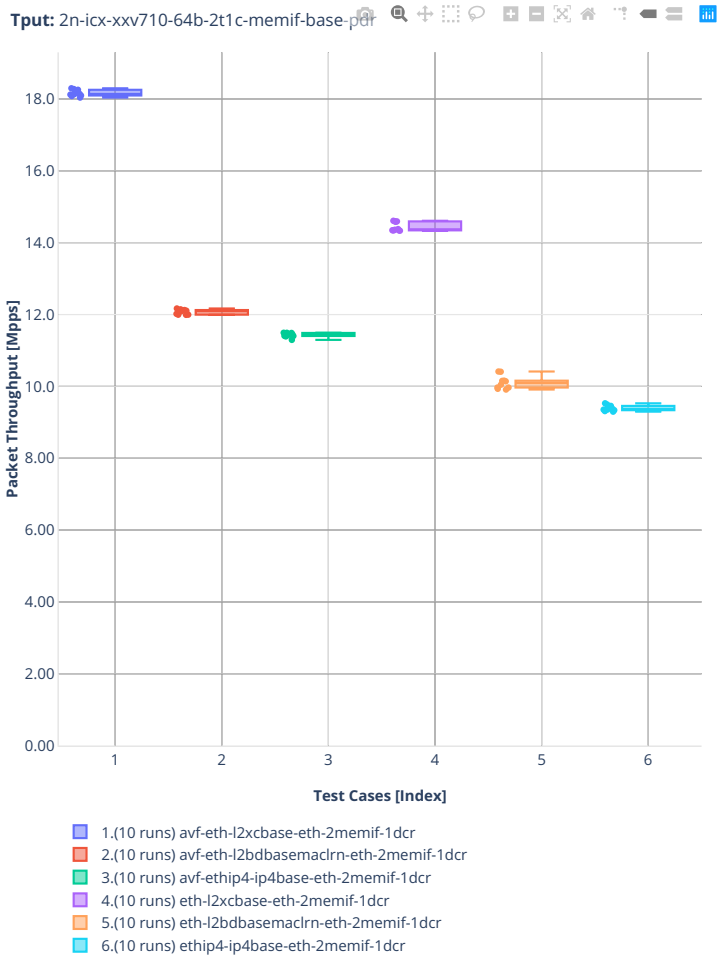
CSIT source code for the test cases used for plots can be found in CSIT git repository[121].

[121] https://git.fd.io/csit/tree/tests/vpp/perf/container_memif?h=rls2210

2n-icx-xxv710

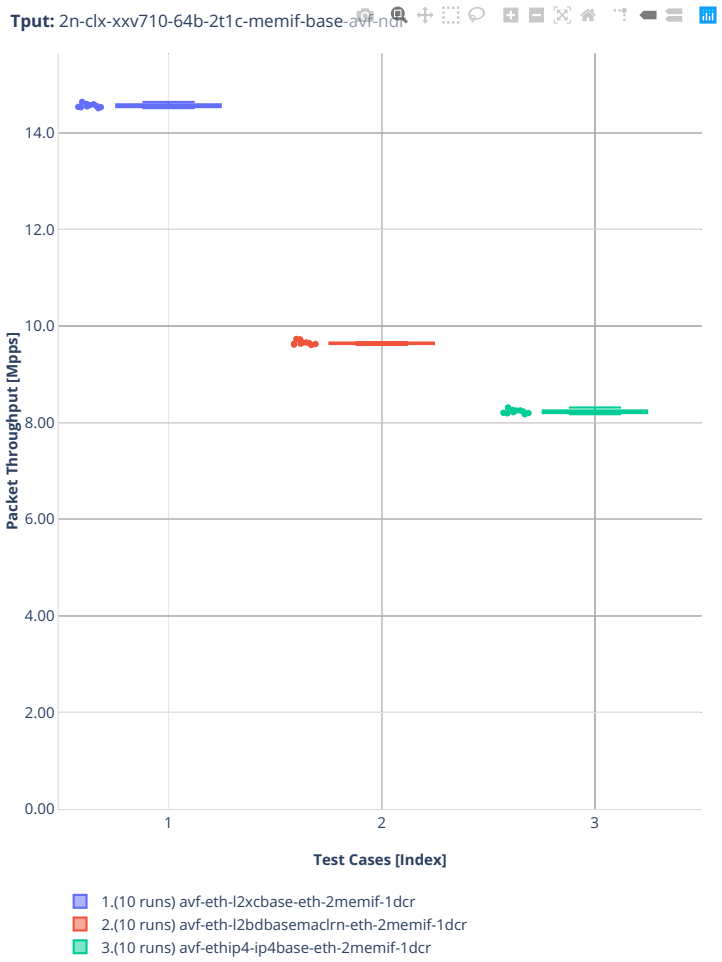
64b-2t1c-memif-base

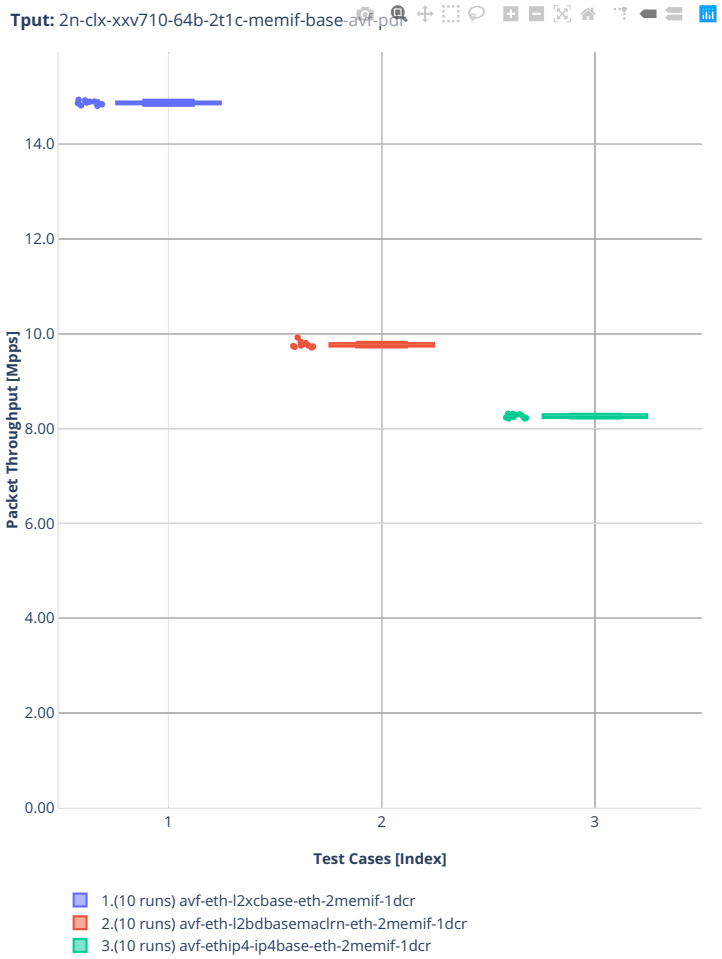



Tput: 2n-icx-xxv710-64b-2t1c-memif-base-pdr
Packet Throughput [Mpps]
18.0
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2memif-1dcr
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
3.(10 runs) avf-ethip4-ip4base-eth-2memif-1dcr
4.(10 runs) eth-l2xcbase-eth-2memif-1dcr
5.(10 runs) eth-l2bdbasemaclrn-eth-2memif-1dcr
6.(10 runs) ethip4-ip4base-eth-2memif-1dcr

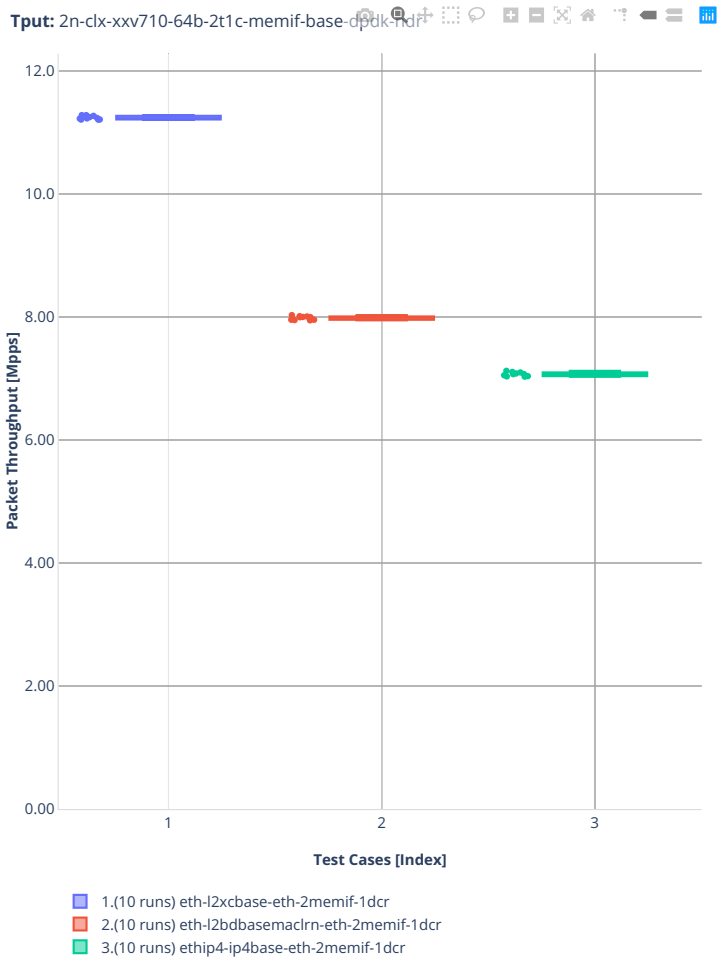
2n-clx-xxv710

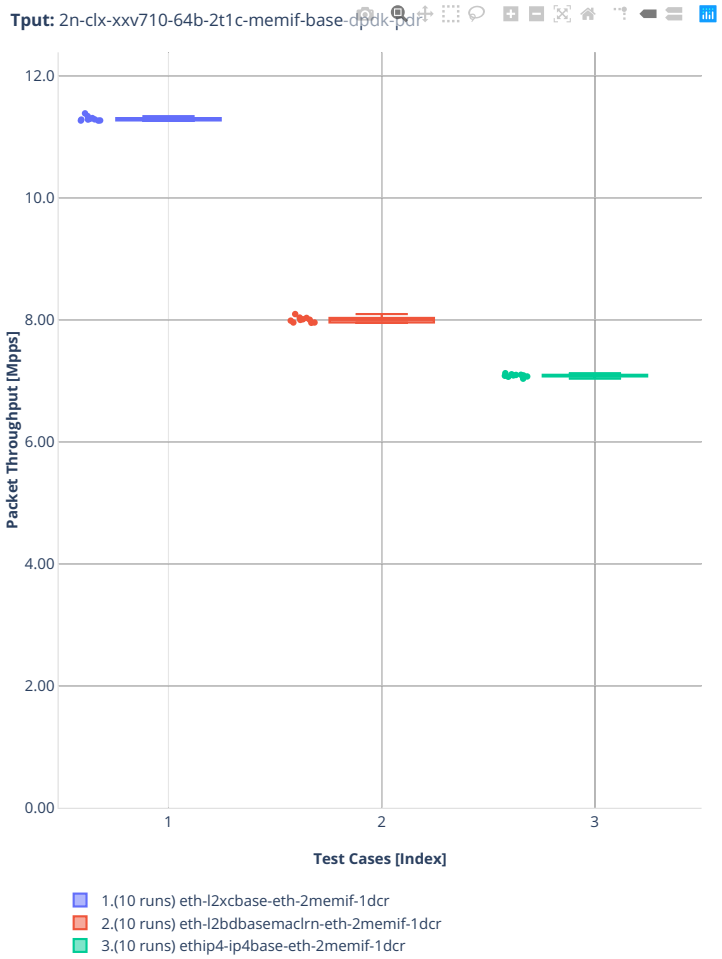
64b-2t1c-memif-base-avf




Tput: 2n-clx-xxv710-64b-2t1c-memif-base-avf-pdr
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2memif-1dcr
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
3.(10 runs) avf-ethip4-ip4base-eth-2memif-1dcr

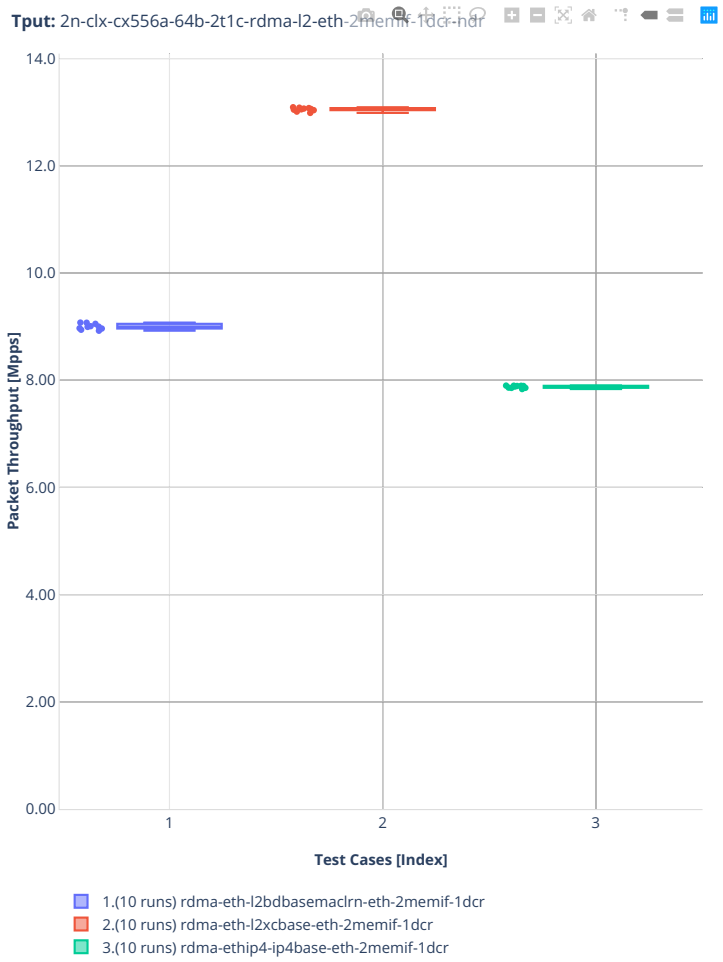
64b-2t1c-memif-base-dpdk

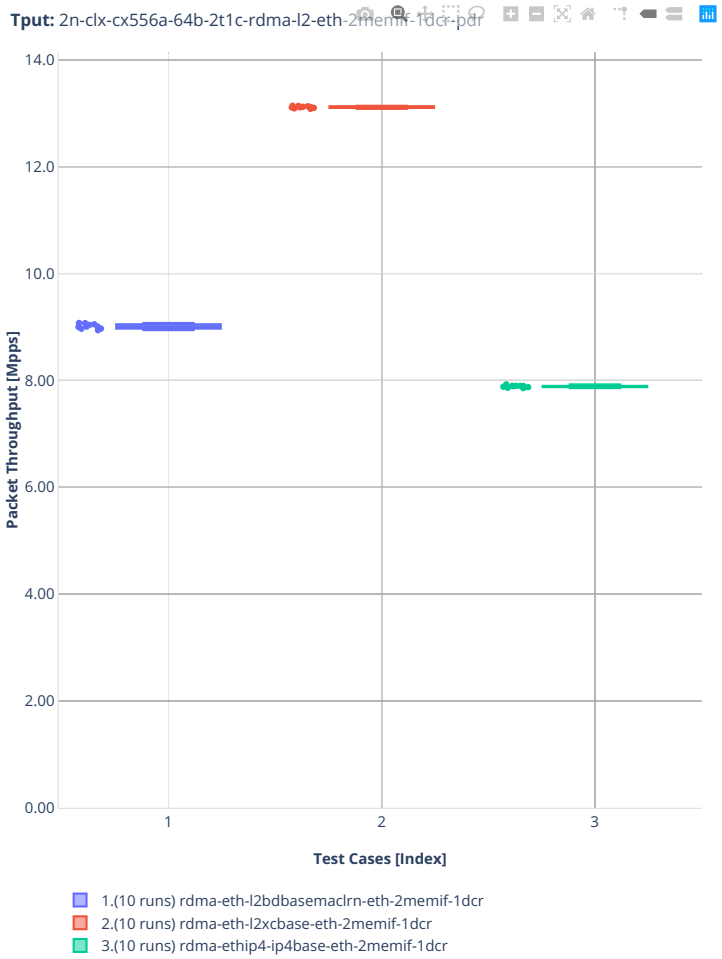



Tput: 2n-clx-xxv710-64b-2t1c-memif-base-dpdk-pdr
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) eth-l2xcbase-eth-2memif-1dcr
2.(10 runs) eth-l2bdbasemaclrn-eth-2memif-1dcr
3.(10 runs) ethip4-ip4base-eth-2memif-1dcr

2n-clx-cx556a

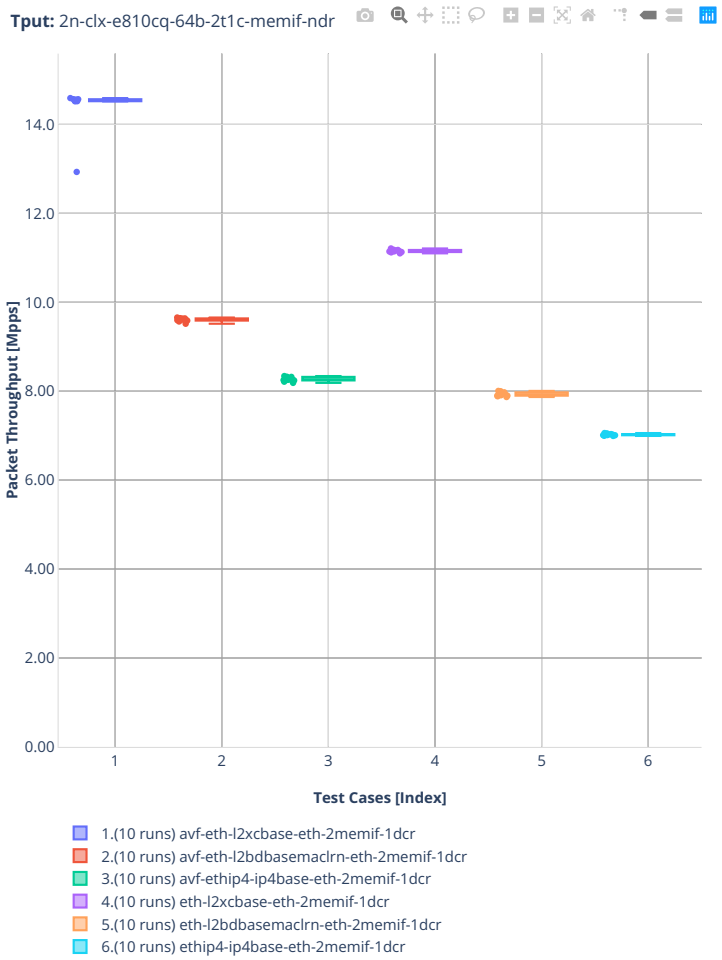
64b-2t1c-memif-base-rdma-core

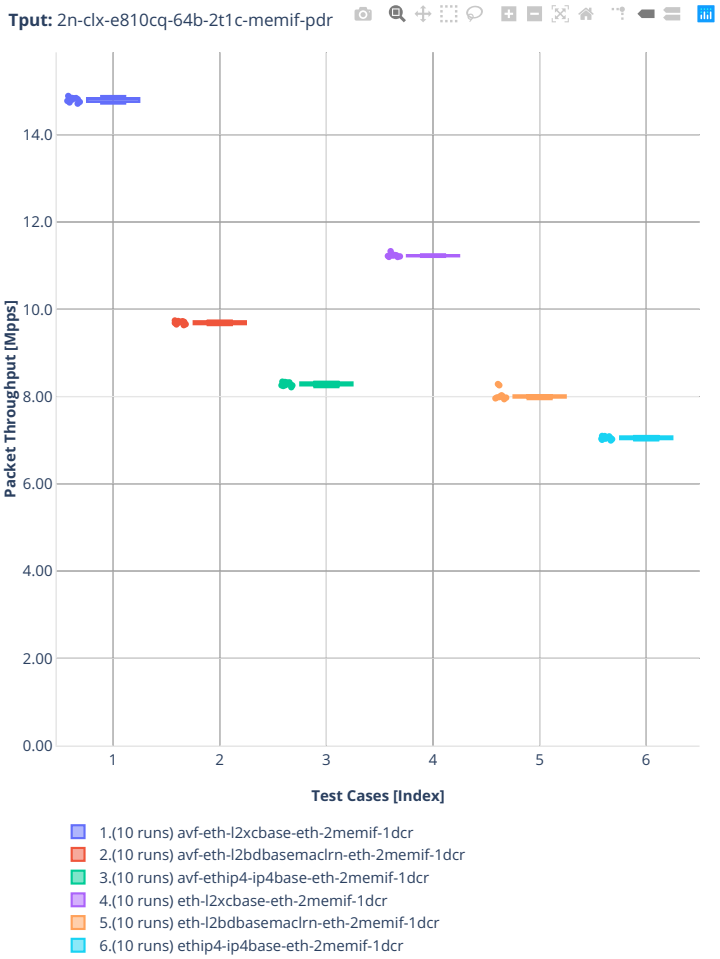



Tput: 2n-clx-cx556a-64b-2t1c-rdma-l2-eth-2memif-1dcr-pdr
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) rdma-eth-l2bdbasemaclrn-eth-2memif-1dcr
2.(10 runs) rdma-eth-l2xcbase-eth-2memif-1dcr
3.(10 runs) rdma-ethip4-ip4base-eth-2memif-1dcr

2n-clx-e810cq

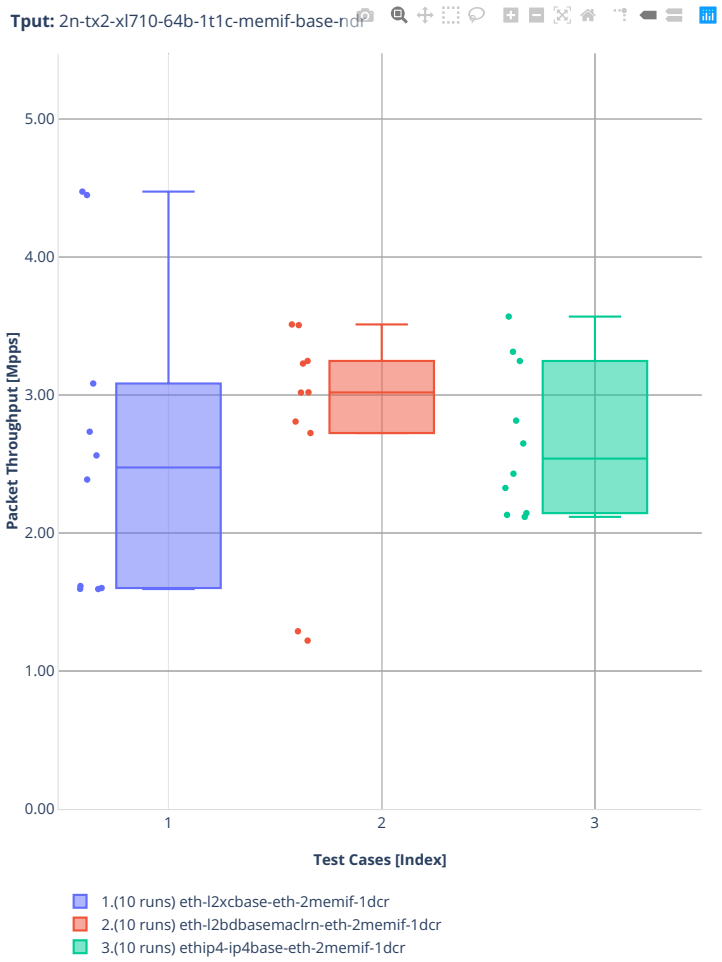
64b-2t1c-memif-base

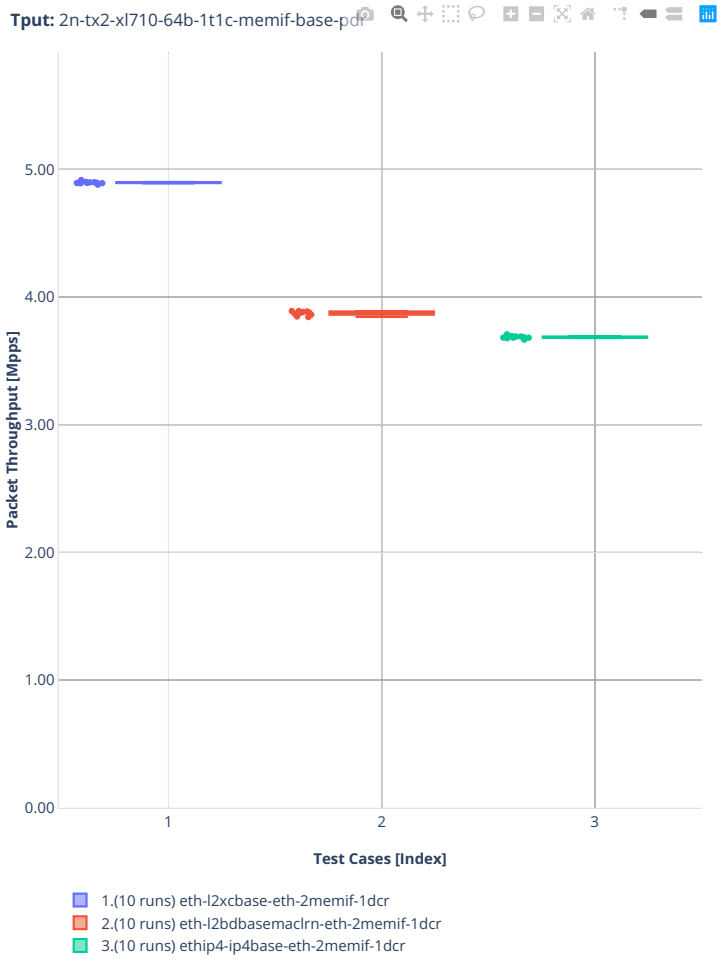



Tput: 2n-clx-e810cq-64b-2t1c-memif-pdr
Packet Throughput [Mpps]
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2memif-1dcr
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
3.(10 runs) avf-ethip4-ip4base-eth-2memif-1dcr
4.(10 runs) eth-l2xcbase-eth-2memif-1dcr
5.(10 runs) eth-l2bdbasemaclrn-eth-2memif-1dcr
6.(10 runs) ethip4-ip4base-eth-2memif-1dcr

2n-tx2-xl710

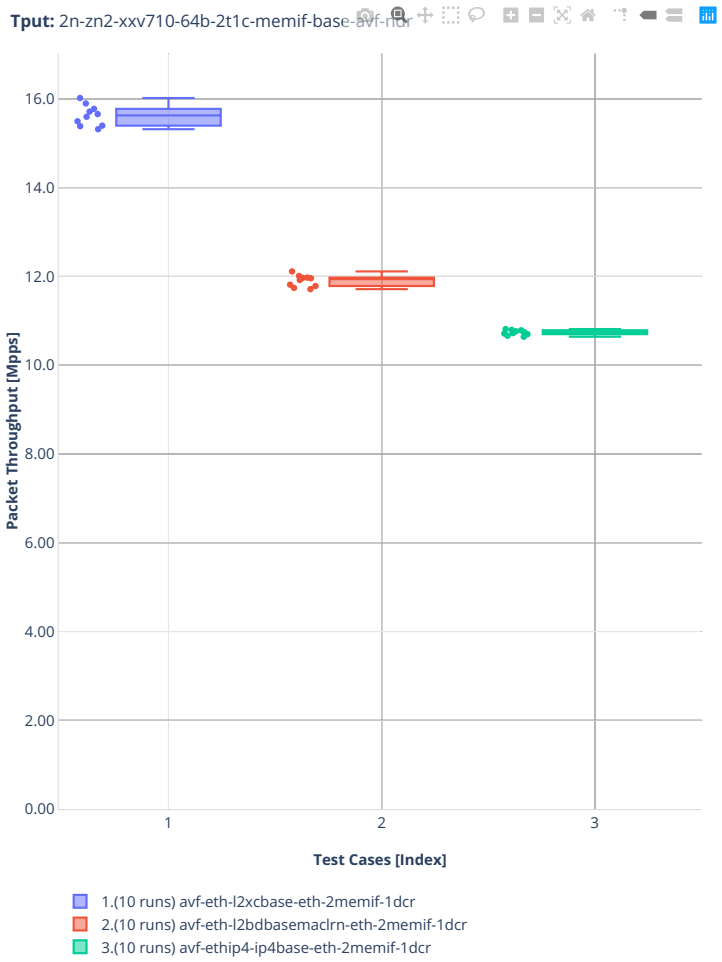
64b-1t1c-memif-base-dpdk

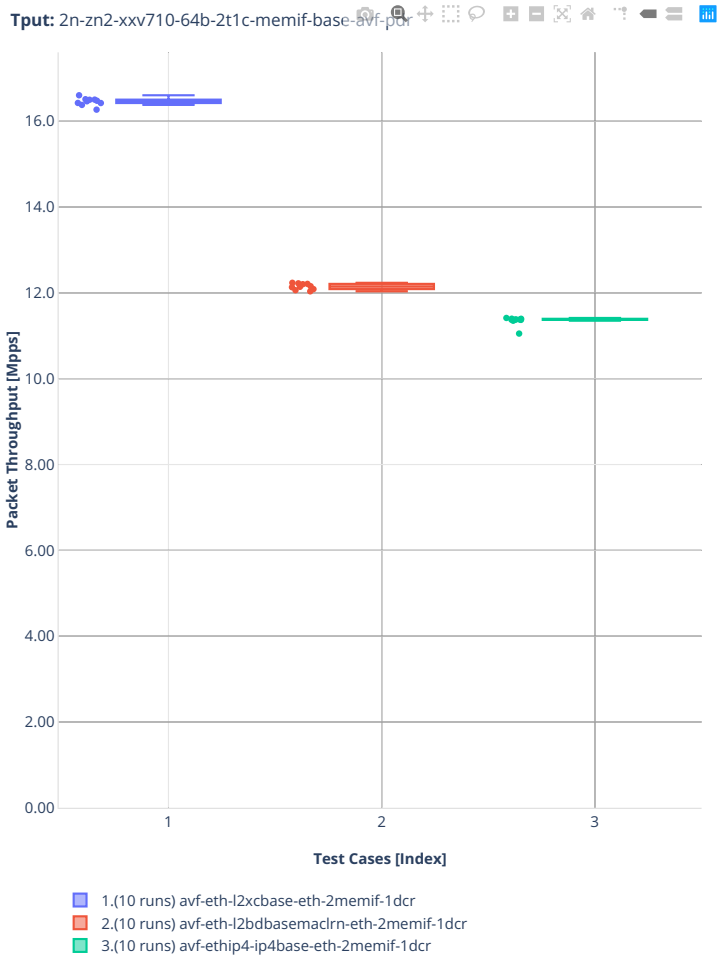



Tput: 2n-tx2-xl710-64b-1t1c-memif-base-pdr
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) eth-l2xcbase-eth-2memif-1dcr
2.(10 runs) eth-l2bdbasemaclrn-eth-2memif-1dcr
3.(10 runs) ethip4-ip4base-eth-2memif-1dcr

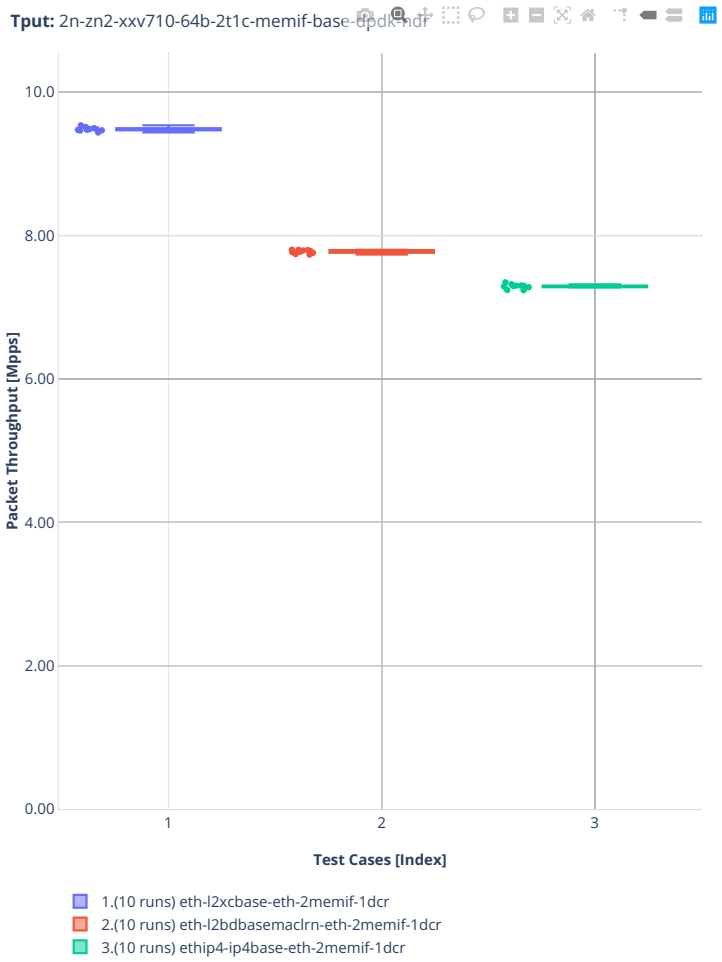
2n-zn2-xxv710

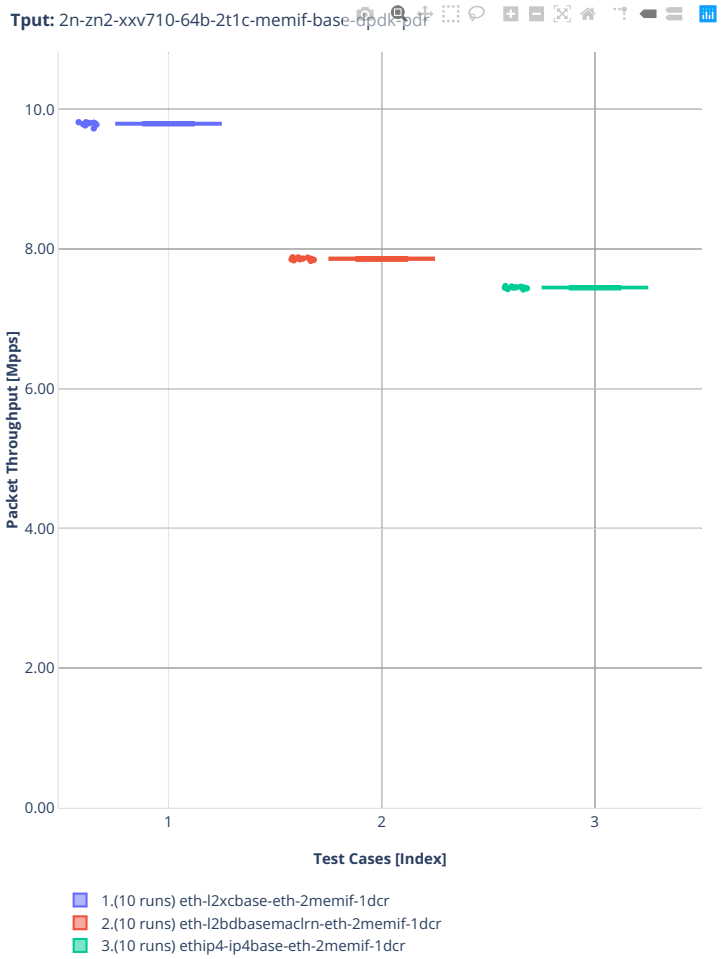
64b-2t1c-memif-base-avf




Tput: 2n-zn2-xxv710-64b-2t1c-memif-base-avf-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) avf-eth-l2xcbase-eth-2memif-1dcr
2.(10 runs) avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
3.(10 runs) avf-ethip4-ip4base-eth-2memif-1dcr

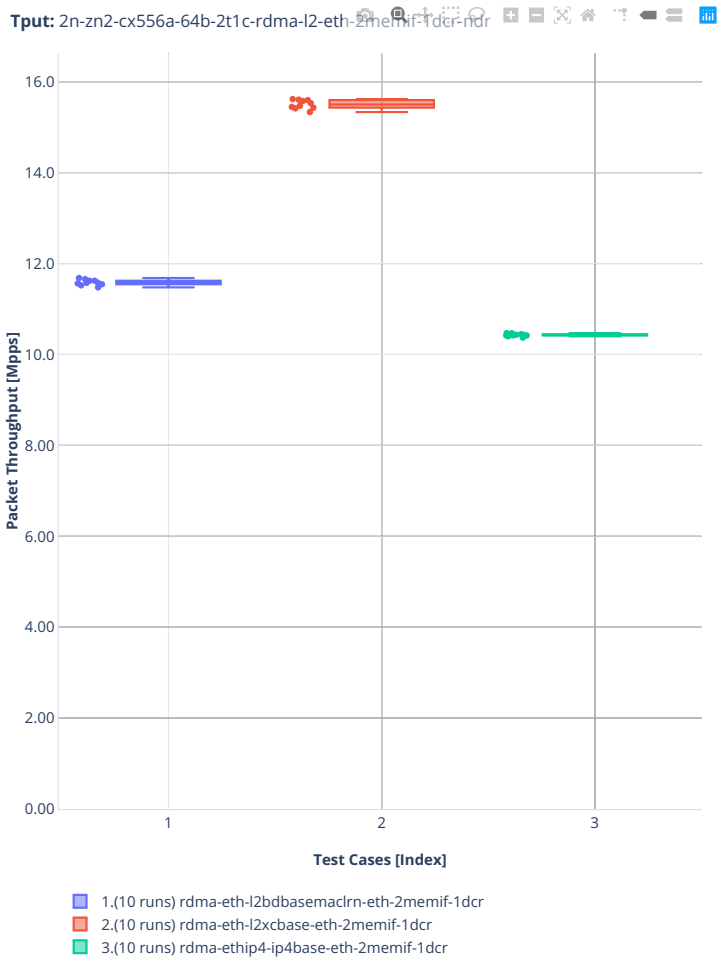
64b-2t1c-memif-base-dpdk

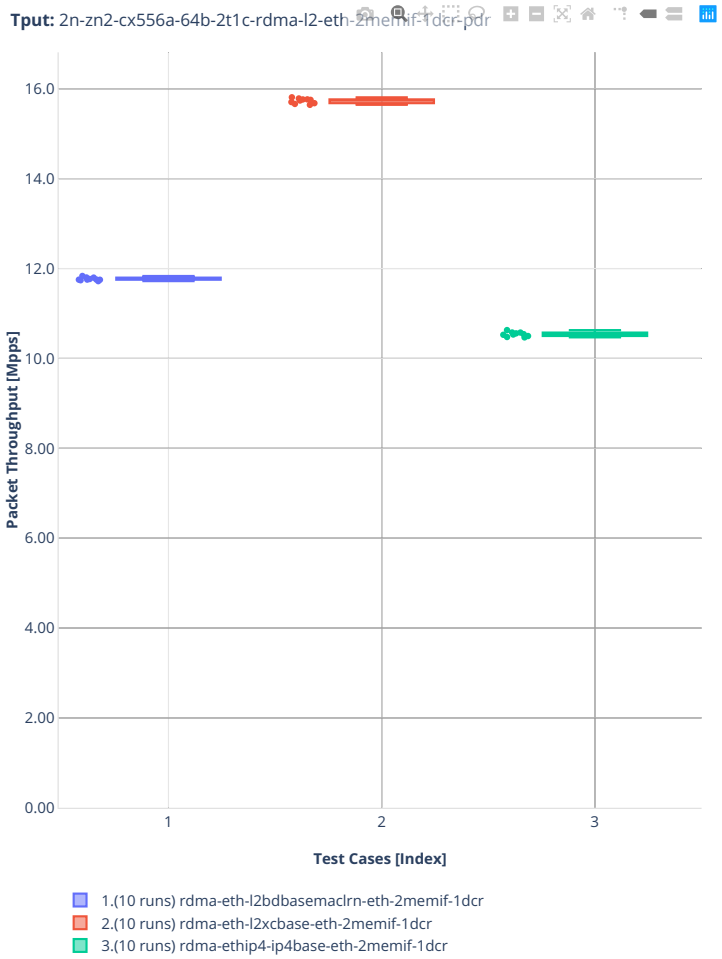



Tput: 2n-zn2-xxv710-64b-2t1c-memif-base-dpdk-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) eth-l2xcbase-eth-2memif-1dcr
2.(10 runs) eth-l2bdbasemaclrn-eth-2memif-1dcr
3.(10 runs) ethip4-ip4base-eth-2memif-1dcr

2n-zn2-cx556a

64b-2t1c-memif-base-rdma-core




Tput: 2n-zn2-cx556a-64b-2t1c-rdma-l2-eth-2memif-1dcr-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) rdma-eth-l2bdbasemaclrn-eth-2memif-1dcr
2.(10 runs) rdma-eth-l2xcbase-eth-2memif-1dcr
3.(10 runs) rdma-ethip4-ip4base-eth-2memif-1dcr

2.3.9 IPSec IPv4 Routing

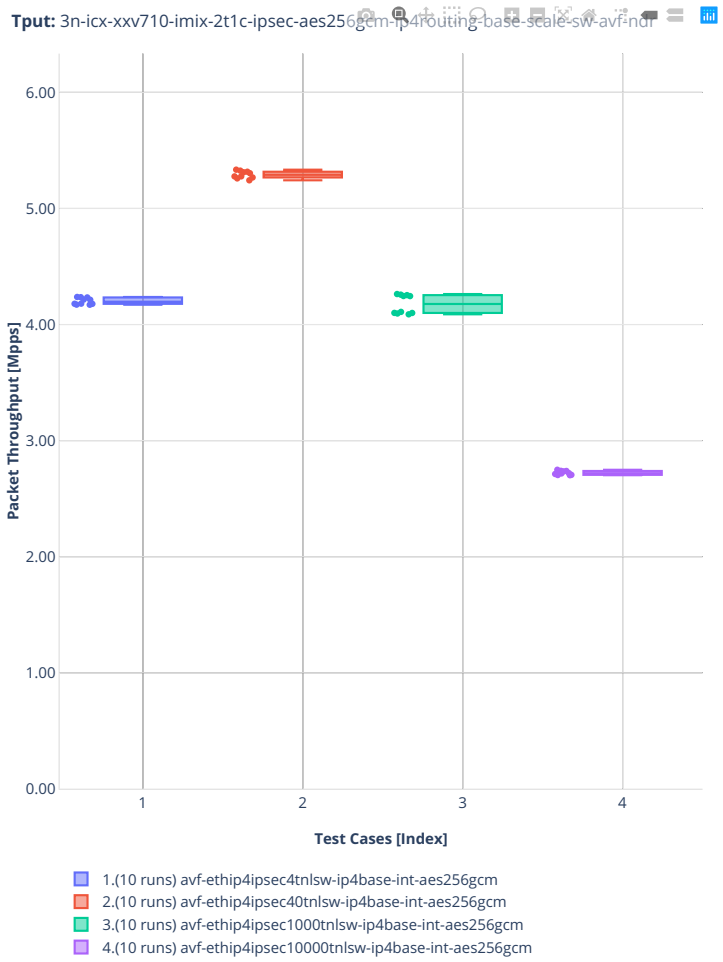
Following sections include summary graphs of VPP Phy-to-Phy performance with IPSec encryption used in combination with IPv4 routed-forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). VPP IPSec encryption is accelerated using DPDK cryptodev library driving Intel Quick Assist (QAT) crypto PCIe hardware cards. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

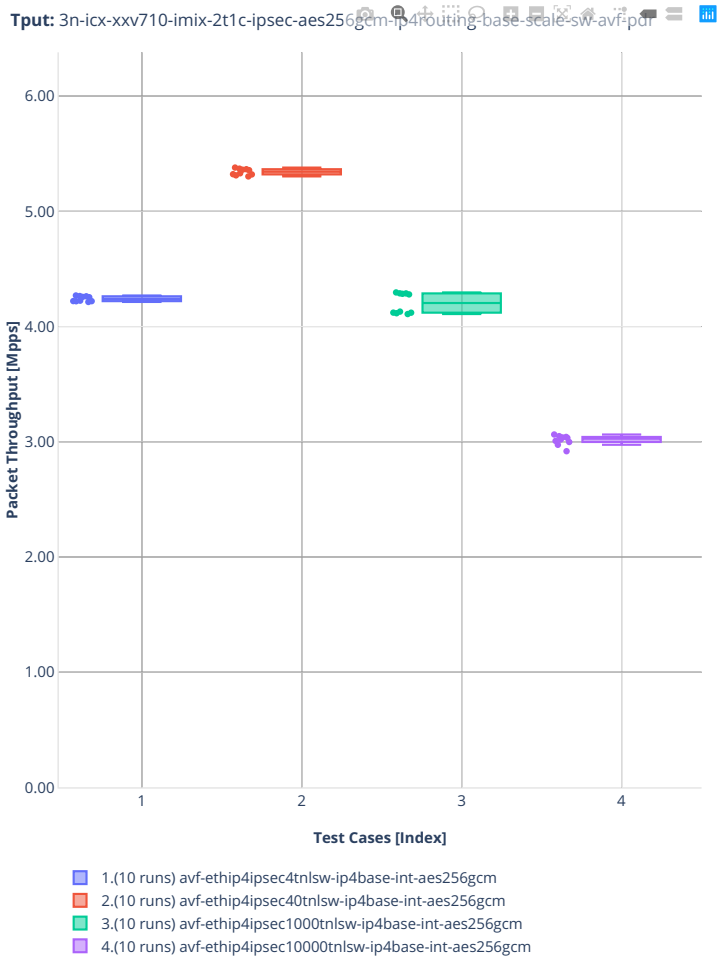
CSIT source code for the test cases used for plots can be found in CSIT git repository[122].

[122] https://git.fd.io/csit/tree/tests/vpp/perf/crypto?h=rls2210

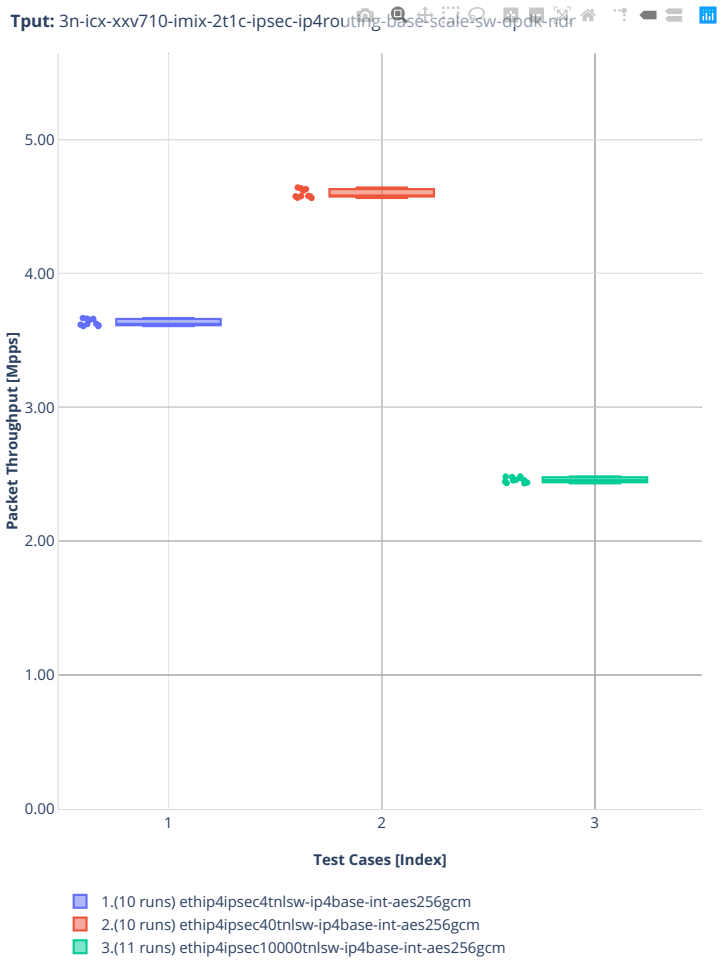
3n-icx-xxv710

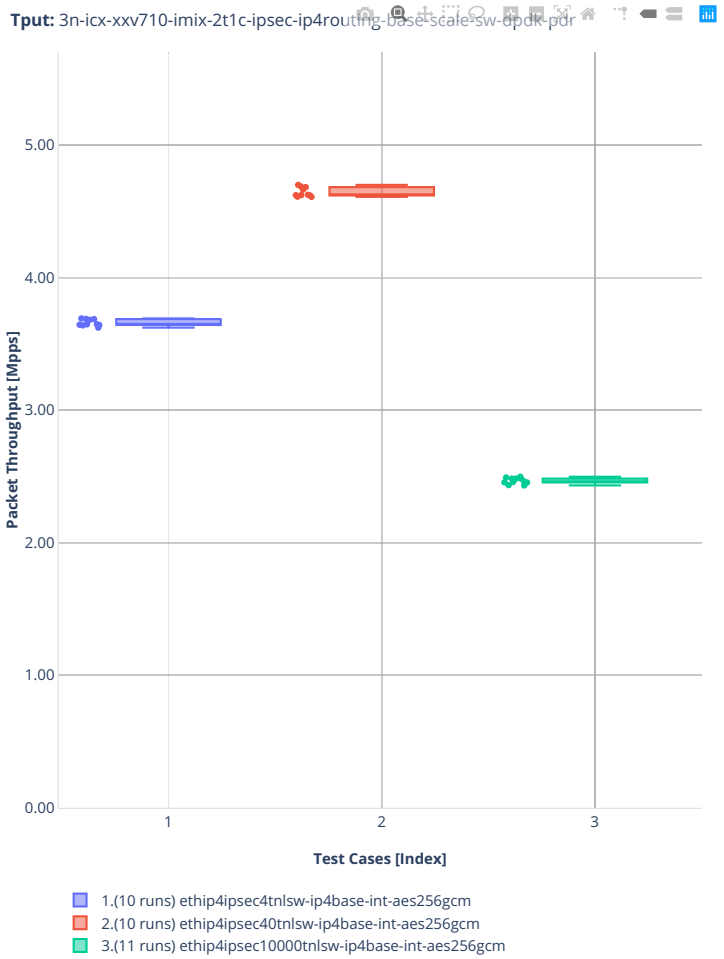
imix-2t1c-ipsec-aes256gcm-ip4routing-base-scale-sw-avf



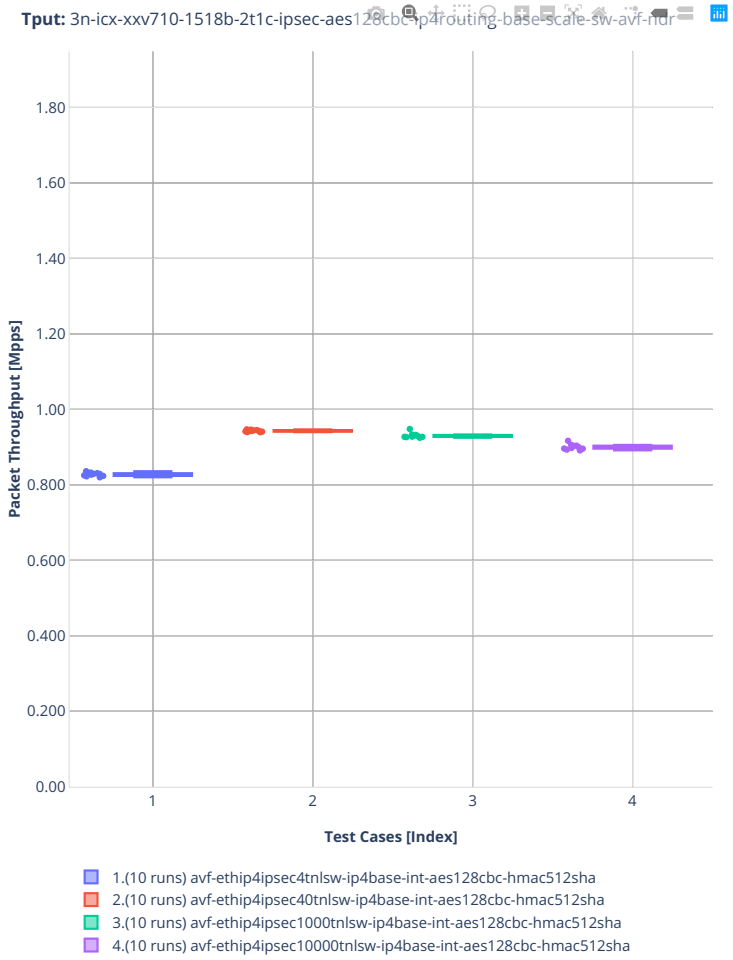

Tput: 3n-icx-xxv710-imix-2t1c-ipsec-aes256gcm-ip4routing-base-scale-sw-avf-pdr
Packet Throughput [Mpps]
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(10 runs) avf-ethip4ipsec40tnlsw-ip4base-int-aes256gcm
3.(10 runs) avf-ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
4.(10 runs) avf-ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

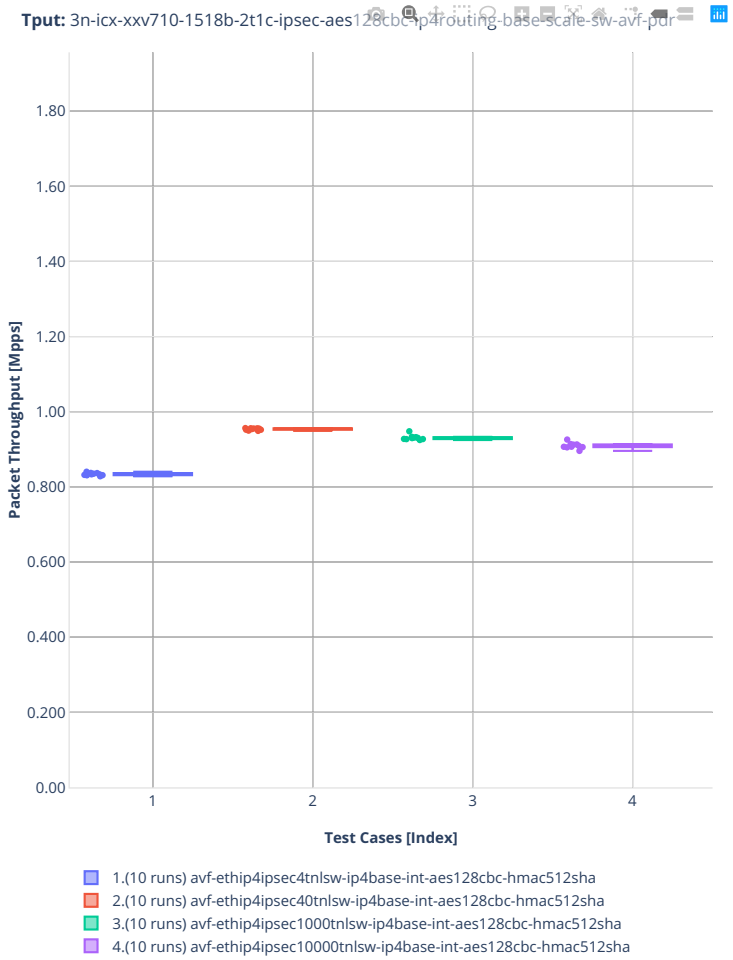
imix-2t1c-ipsec-ip4routing-base-scale-sw-dpdk



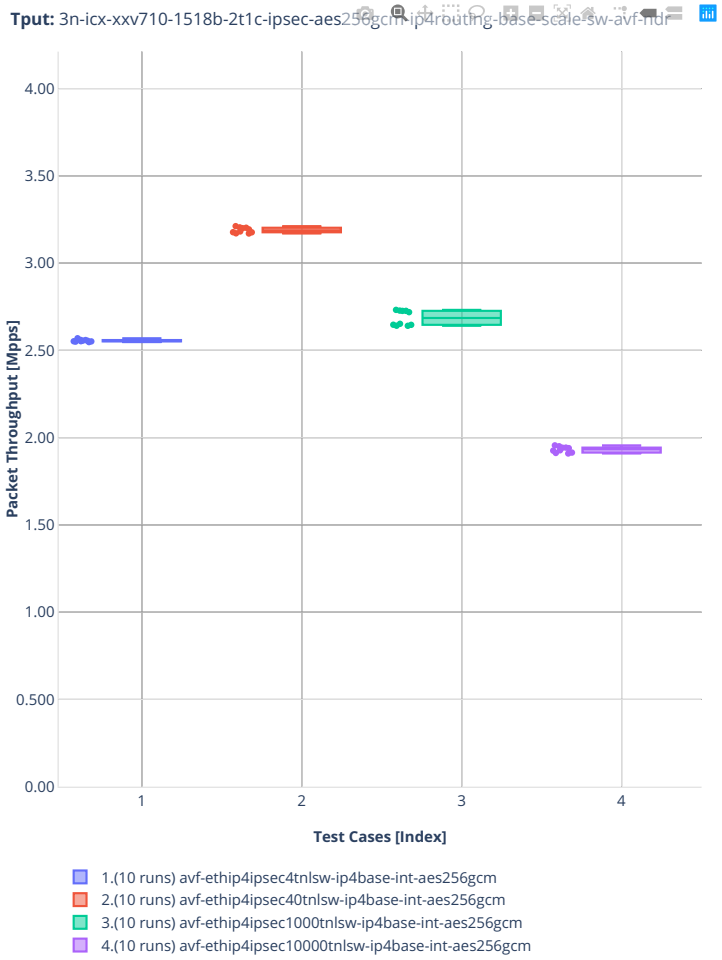

Tput: 3n-icx-xxv710-imix-2t1c-ipsec-ip4routing-base-scale-sw-dpdk-pdr
5.00
4.00
3.00
2.00
1.00
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(10 runs) ethip4ipsec40tnlsw-ip4base-int-aes256gcm
3.(11 runs) ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

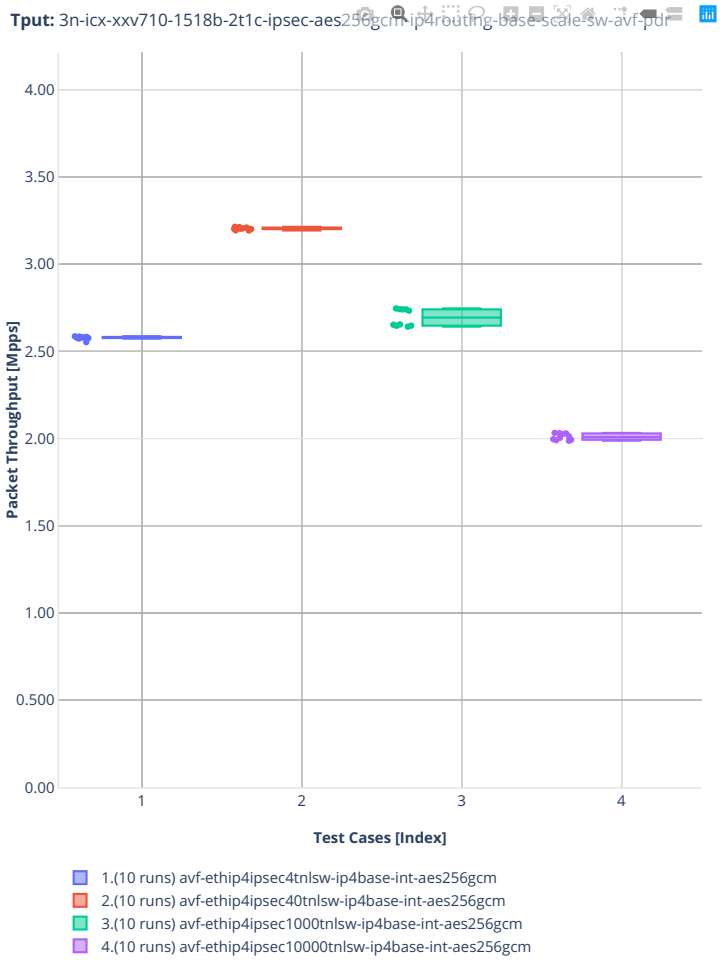
1518b-2t1c-ipsec-aes128cbc-ip4routing-base-scale-sw-avf



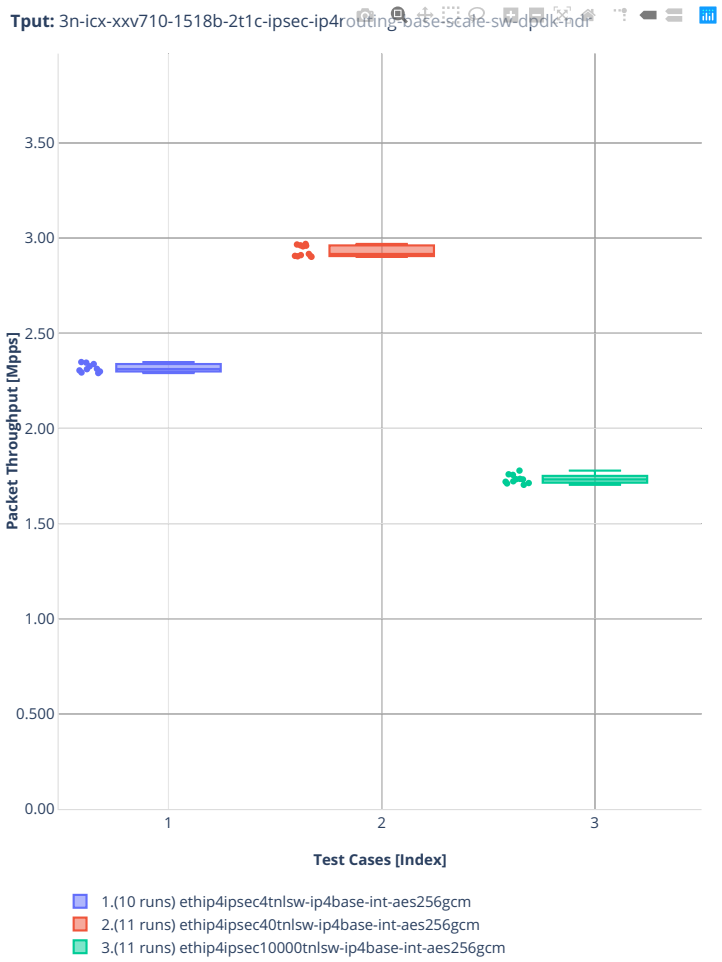

Tput: 3n-icx-xxv710-1518b-2t1c-ipsec-aes128cbc-ip4routing-base-scale-sw-avf-pdr
Packet Throughput [Mpps]
1.80
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
2.(10 runs) avf-ethip4ipsec40tnlsw-ip4base-int-aes128cbc-hmac512sha
3.(10 runs) avf-ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
4.(10 runs) avf-ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha

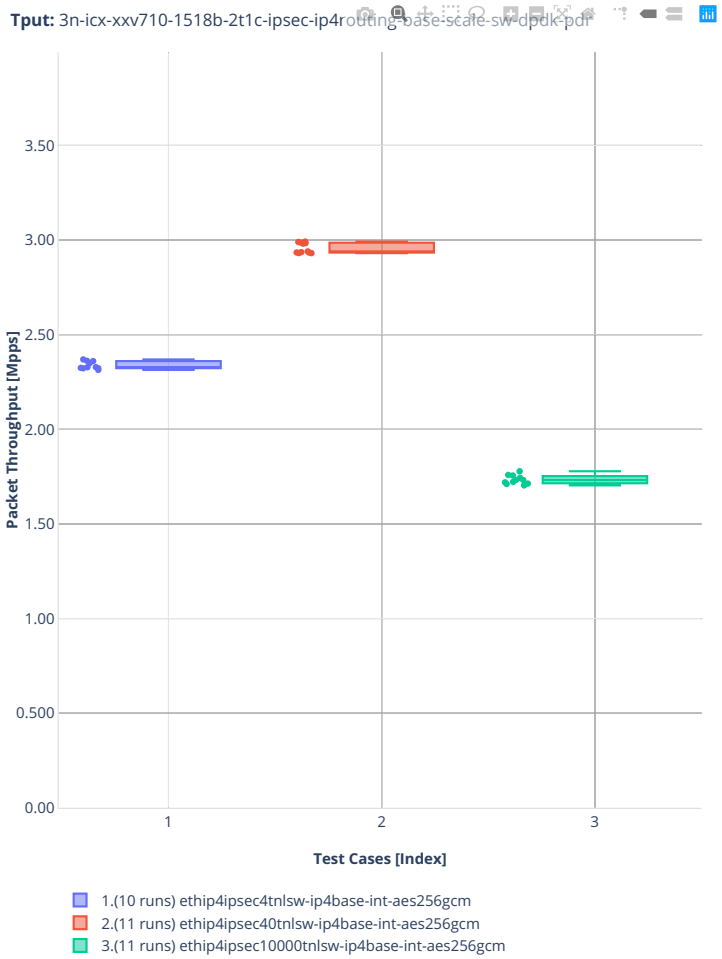
1518b-2t1c-ipsec-aes256gcm-ip4routing-base-scale-sw-avf




Tput: 3n-icx-xxv710-1518b-2t1c-ipsec-aes256gcm-ip4routing-base-scale-sw-avf-pdr
Packet Throughput [Mpps]
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.500
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) avf-ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(10 runs) avf-ethip4ipsec40tnlsw-ip4base-int-aes256gcm
3.(10 runs) avf-ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
4.(10 runs) avf-ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

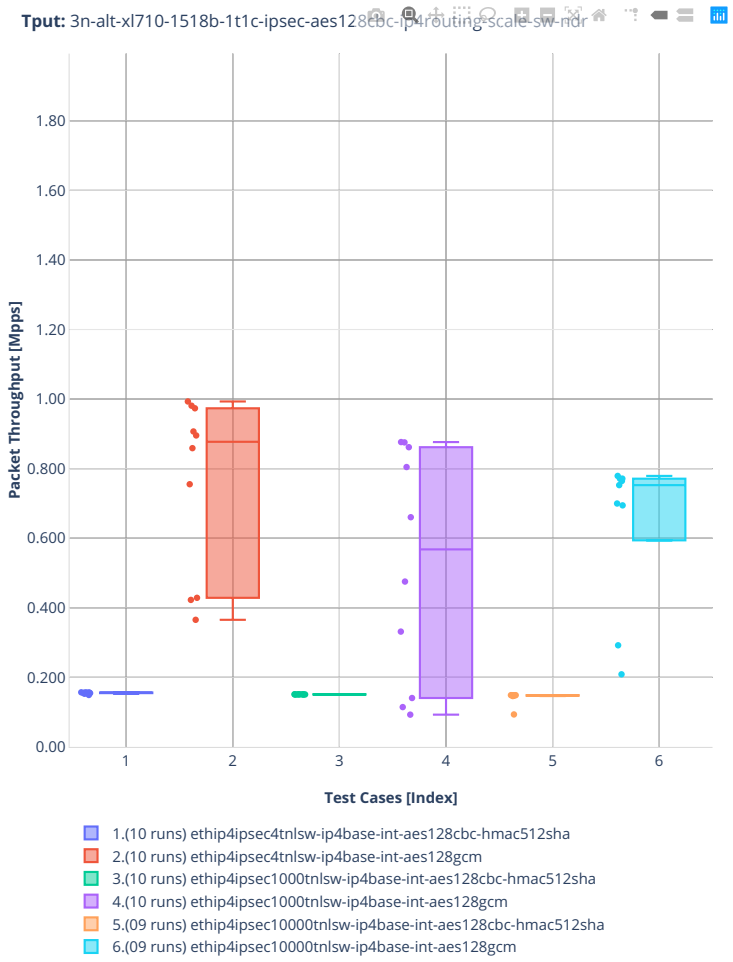
1518b-2t1c-ipsec-ip4routing-base-scale-sw-dpdk

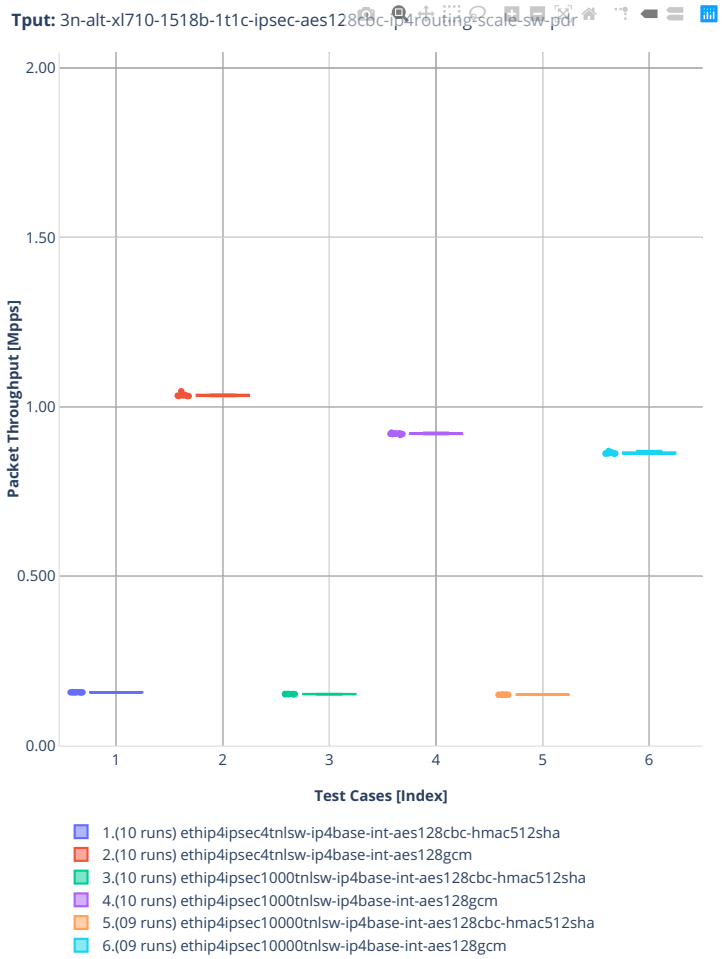



Tput: 3n-icx-xxv710-1518b-2t1c-ipsec-ip4routing-base-scale-sw-dpdk-pdr
Packet Throughput [Mpps]
3.50
3.00
2.50
2.00
1.50
1.00
0.500
0.00
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(11 runs) ethip4ipsec40tnlsw-ip4base-int-aes256gcm
3.(11 runs) ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

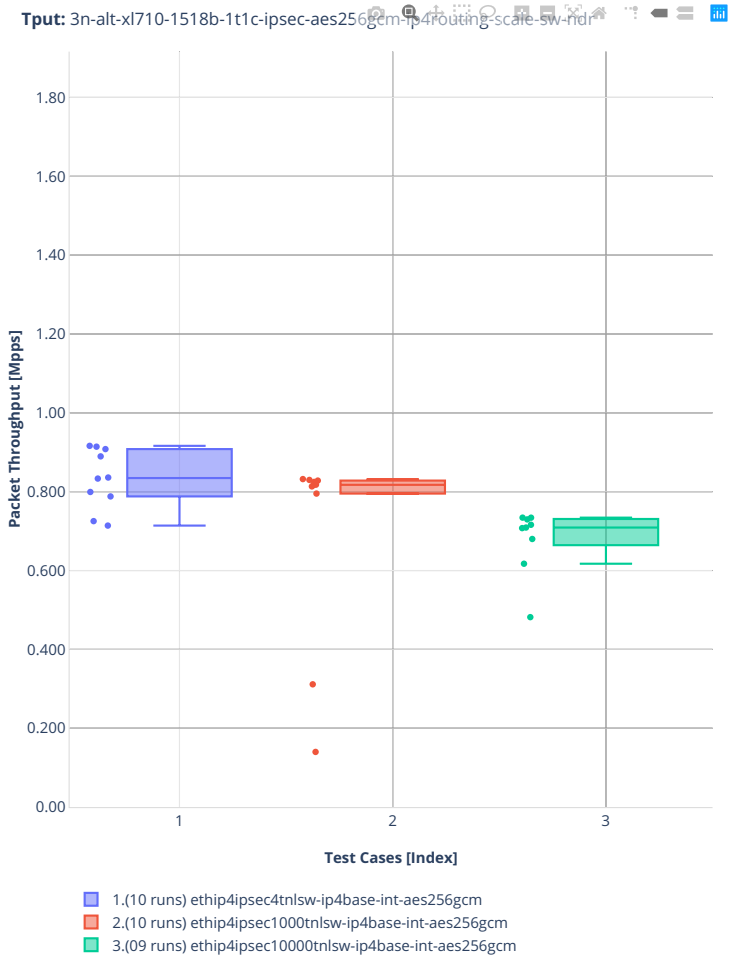
3n-alt-xl710

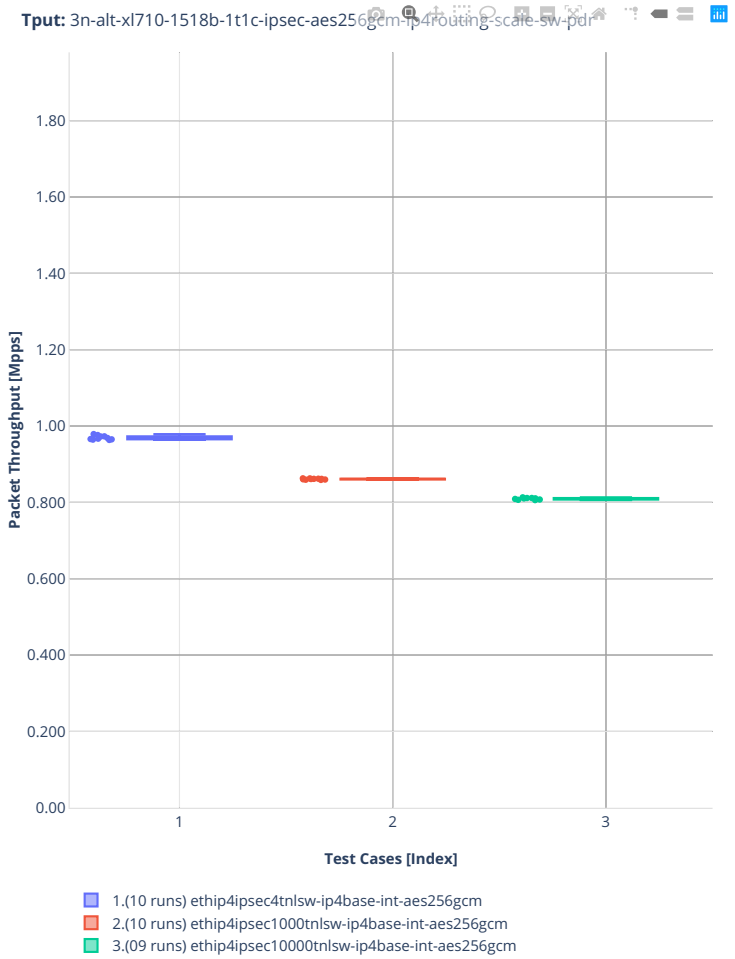
1518b-1t1c-ipsec-aes128cbc-ip4routing-scale-sw



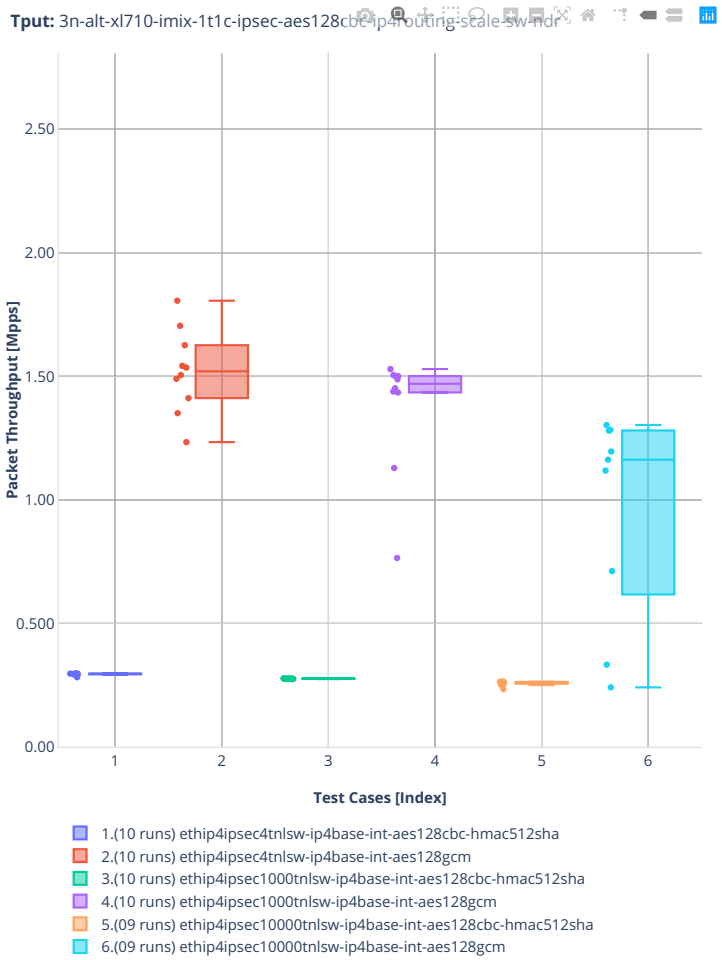

Tput: 3n-alt-xl710-1518b-1t1c-ipsec-aes128cbc-ip4routing-scale-sw-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
2.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128gcm
3.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
4.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128gcm
5.(09 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha
6.(09 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128gcm

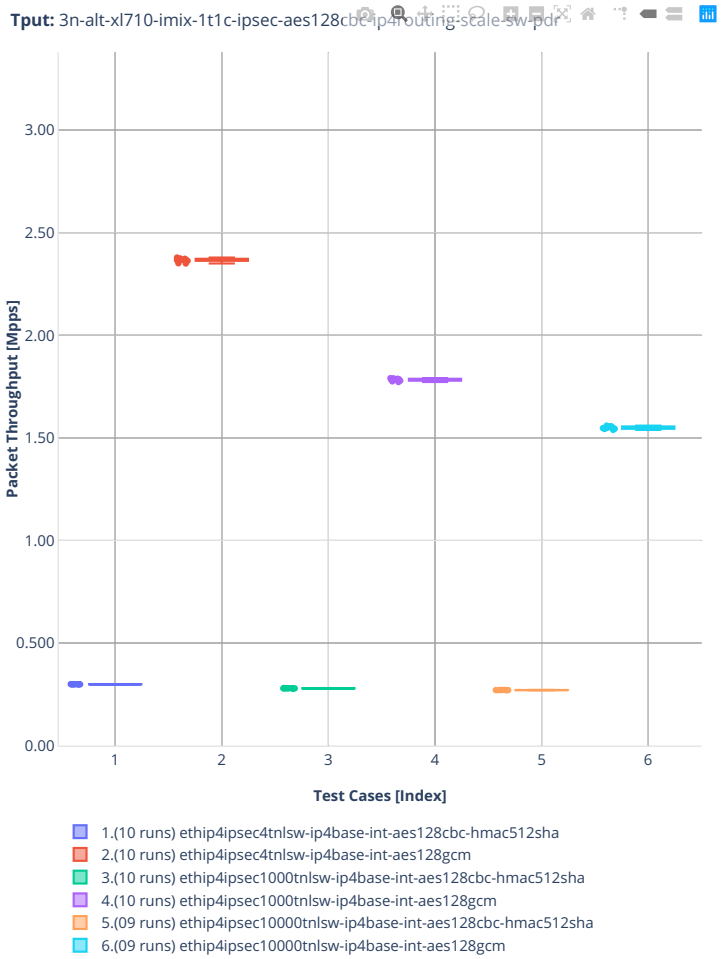
1518b-1t1c-ipsec-aes256gcm-ip4routing-scale-sw



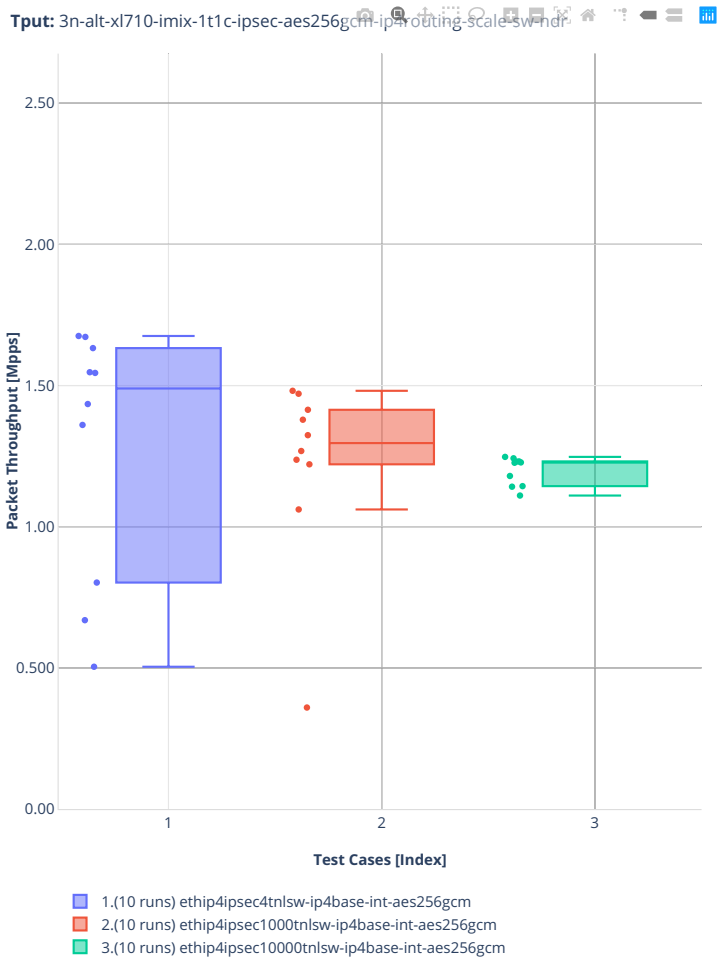

Tput: 3n-alt-xl710-1518b-1t1c-ipsec-aes256gcm-ip4routing-scale-sw-pdr
Packet Throughput [Mpps]
1.80
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
3.(09 runs) ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

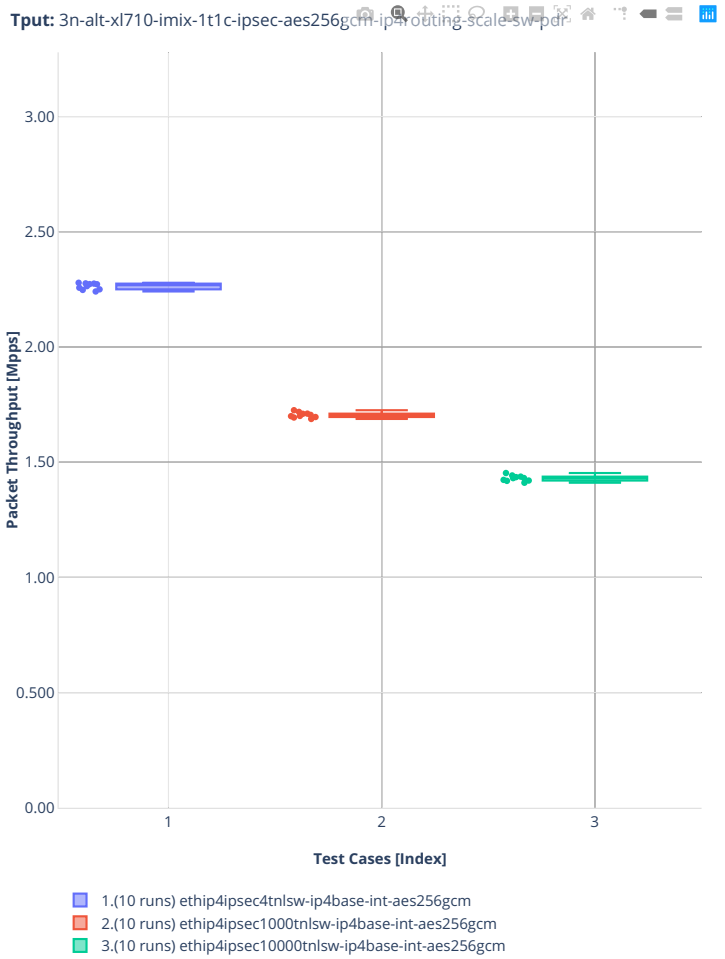
imix-1t1c-ipsec-aes128cbc-ip4routing-scale-sw



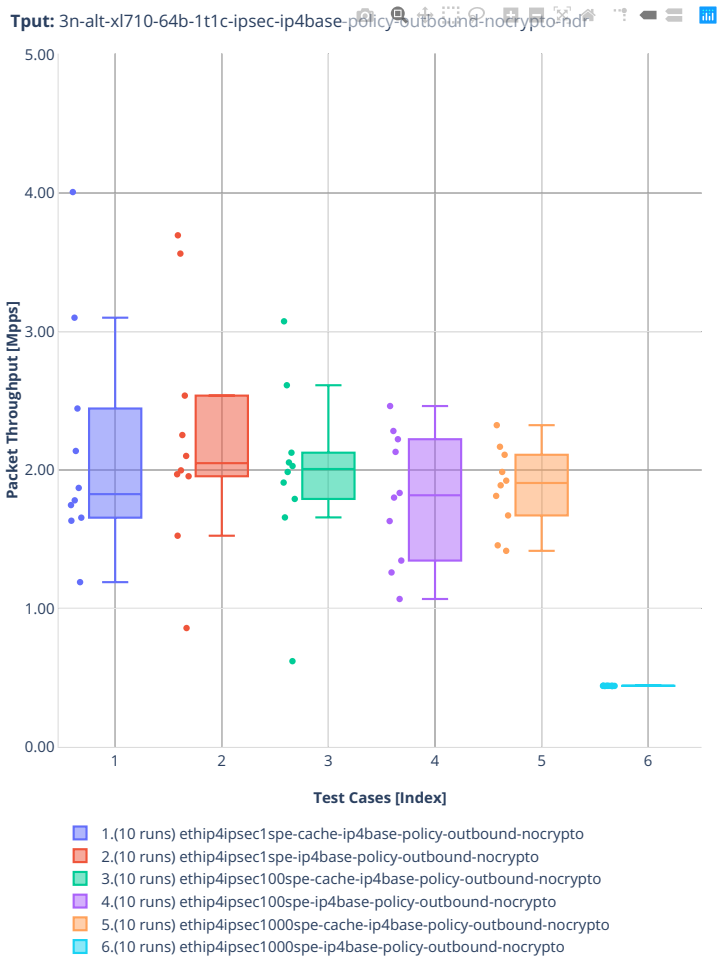

Tput: 3n-alt-xl710-imix-1t1c-ipsec-aes128cbc-ip4routing-scale-sw-pdr
3.00
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
2.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128gcm
3.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
4.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128gcm
5.(09 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha
6.(09 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128gcm

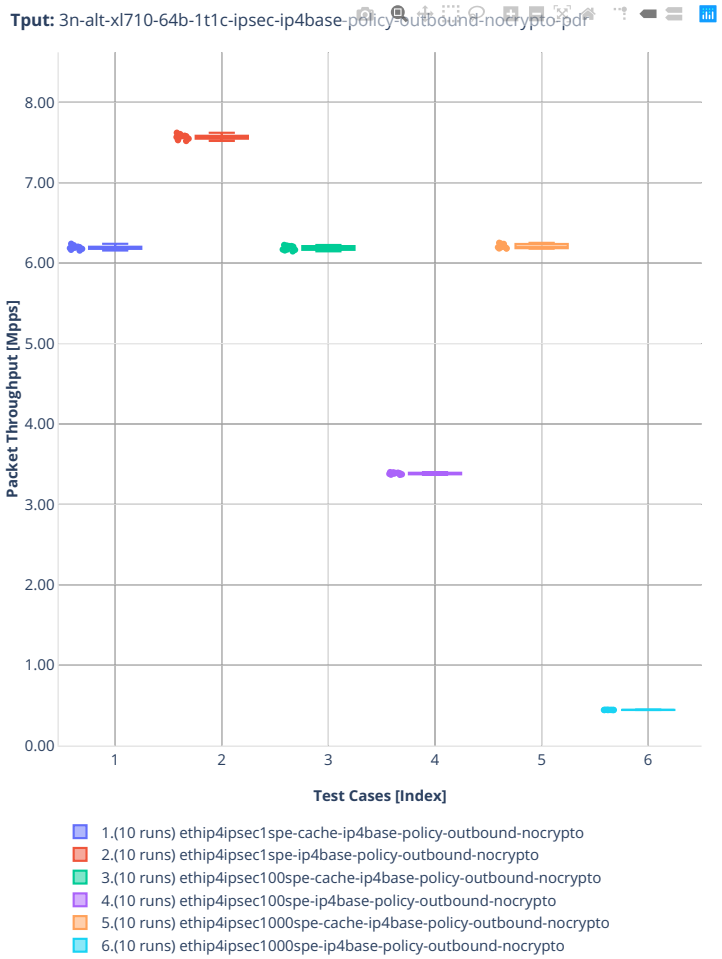
imix-1t1c-ipsec-aes256gcm-ip4routing-scale-sw



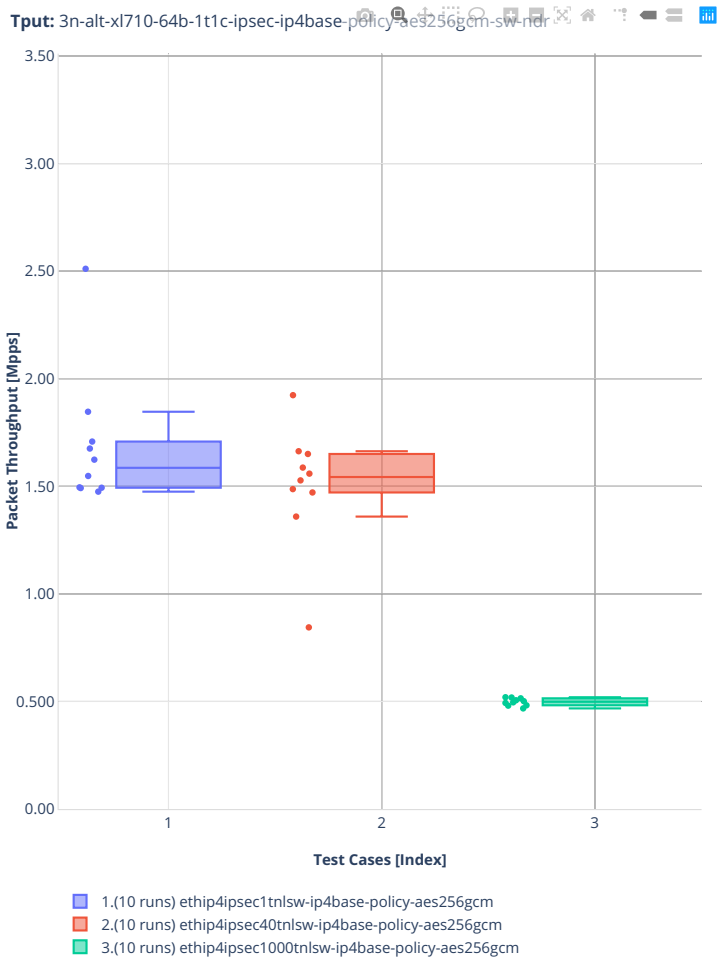

Tput: 3n-alt-xl710-imix-1t1c-ipsec-aes256gcm-ip4routing-scale-sw-pdr
3.00
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
3.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

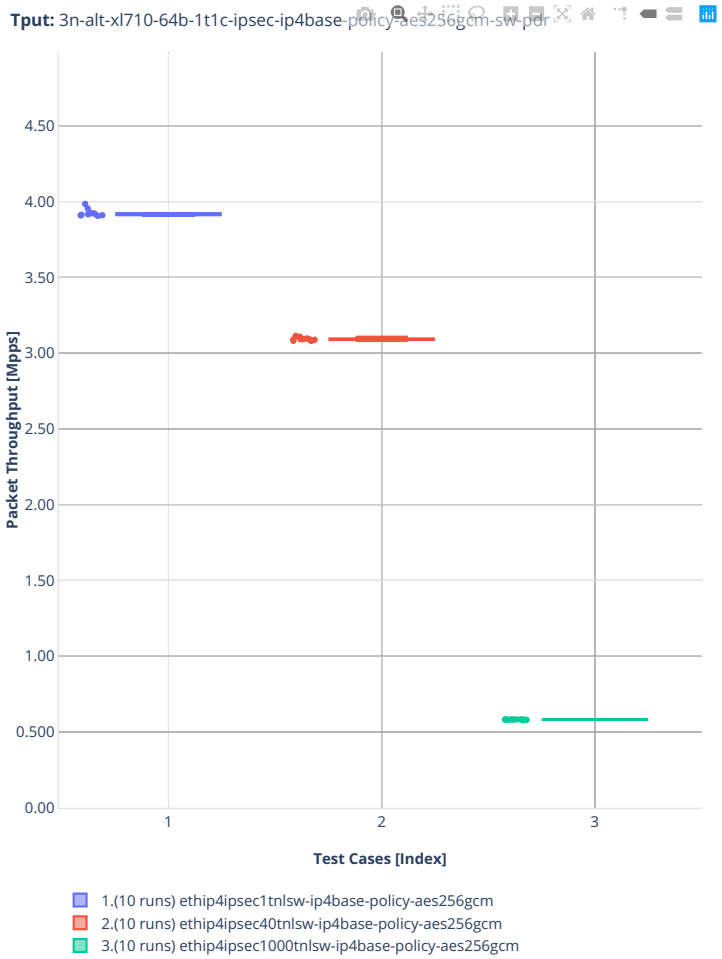
64b-1t1c-ipsec-ip4base-policy-outbound-nocrypto




Tput: 3n-alt-xl710-64b-1t1c-ipsec-ip4base-policy-outbound-nocrypto-pdr
Packet Throughput [Mpps]
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4ipsec1spe-cache-ip4base-policy-outbound-nocrypto
2.(10 runs) ethip4ipsec1spe-ip4base-policy-outbound-nocrypto
3.(10 runs) ethip4ipsec100spe-cache-ip4base-policy-outbound-nocrypto
4.(10 runs) ethip4ipsec100spe-ip4base-policy-outbound-nocrypto
5.(10 runs) ethip4ipsec1000spe-cache-ip4base-policy-outbound-nocrypto
6.(10 runs) ethip4ipsec1000spe-ip4base-policy-outbound-nocrypto

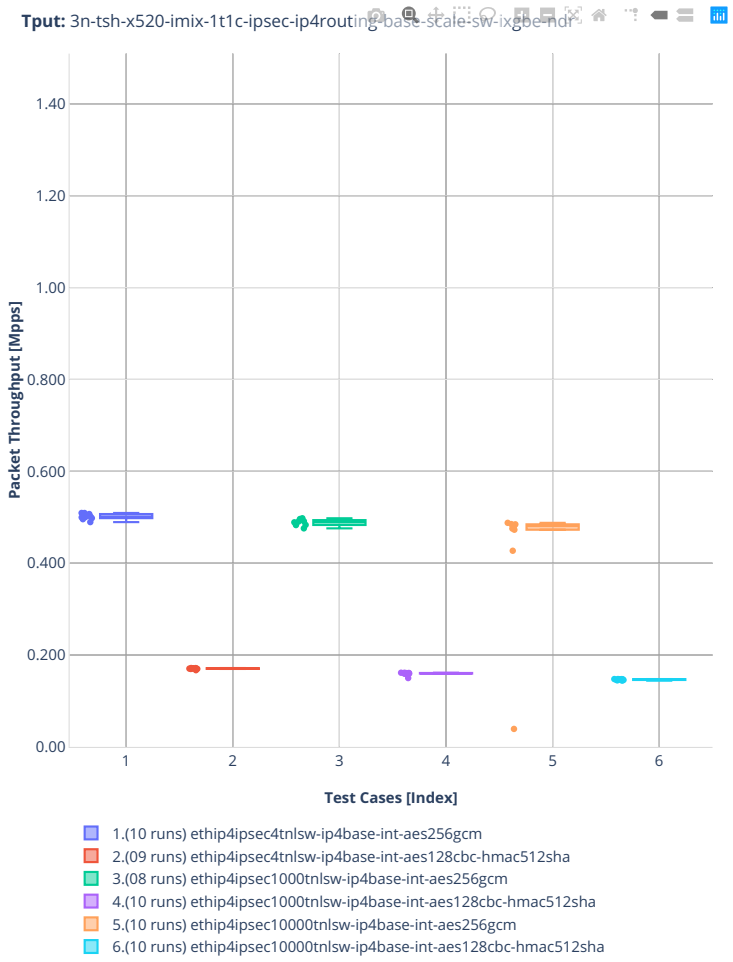
64b-1t1c-ipsec-ip4base-policy-aes256gcm-sw

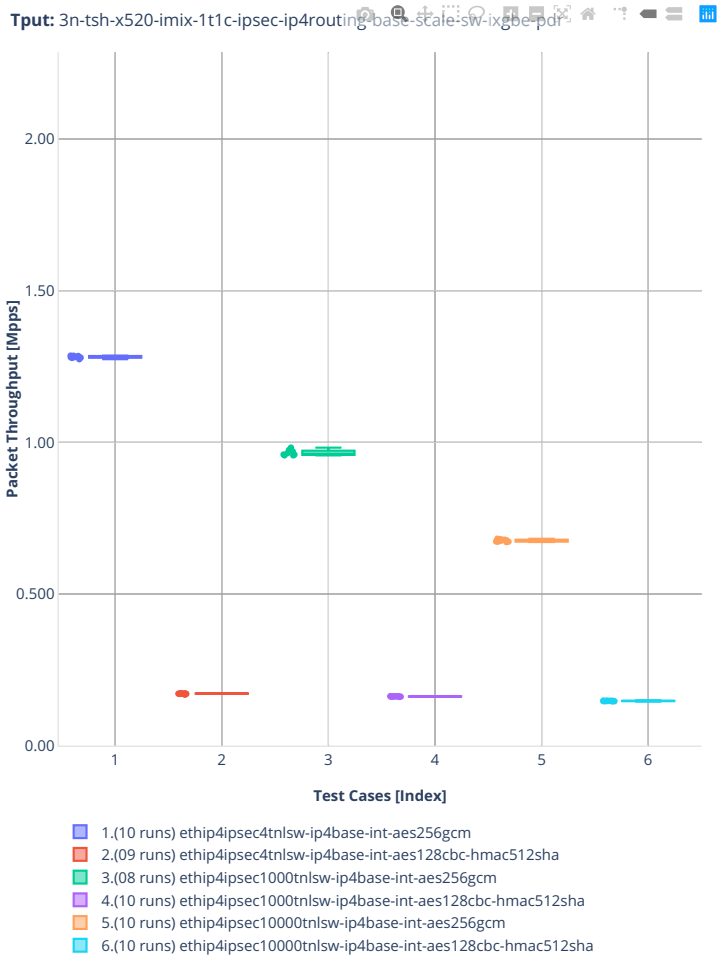



Tput: 3n-alt-xl710-64b-1t1c-ipsec-ip4base-policy-aes256gcm-sw-pdr
Packet Throughput [Mpps]
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.500
0.00
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec1tnlsw-ip4base-policy-aes256gcm
2.(10 runs) ethip4ipsec40tnlsw-ip4base-policy-aes256gcm
3.(10 runs) ethip4ipsec1000tnlsw-ip4base-policy-aes256gcm

3n-tsh-x520

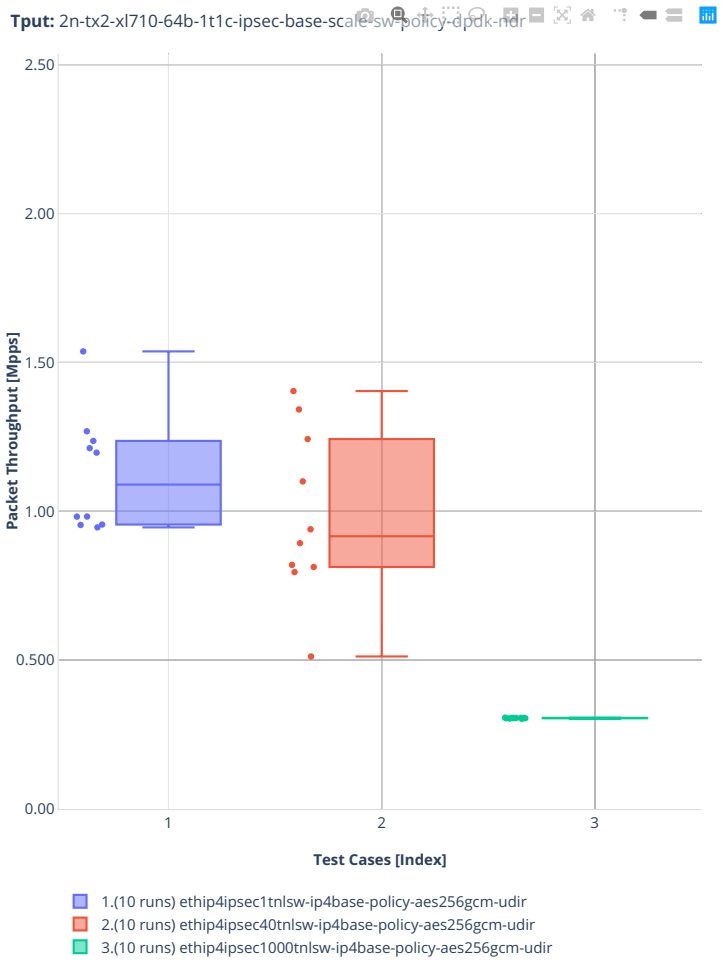
imix-1t1c-ipsec-ip4routing-base-scale-sw-ixgbe

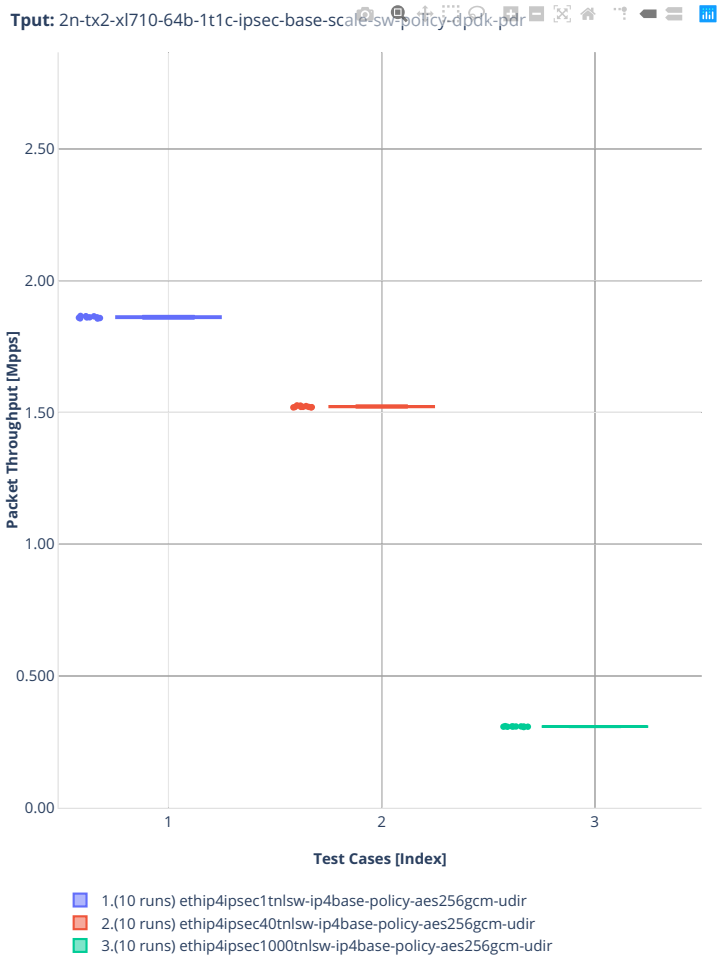



Tput: 3n-tsh-x520-imix-1t1c-ipsec-ip4routing-base-scale-sw-ixgbe-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(09 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
3.(08 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
4.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
5.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes256gcm
6.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha

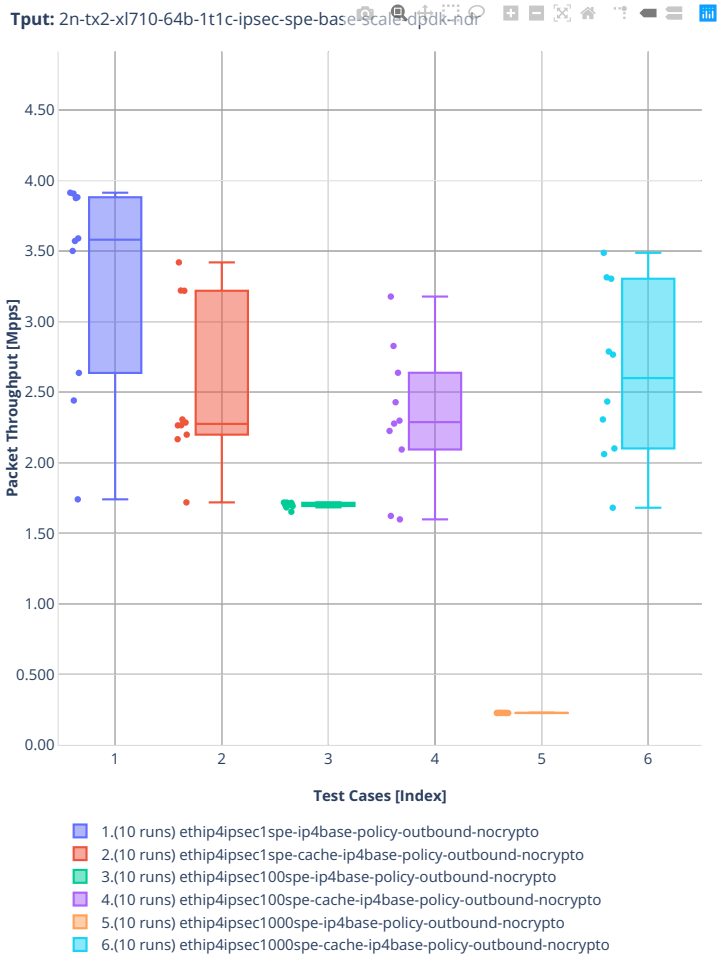
2n-tx2-xl710

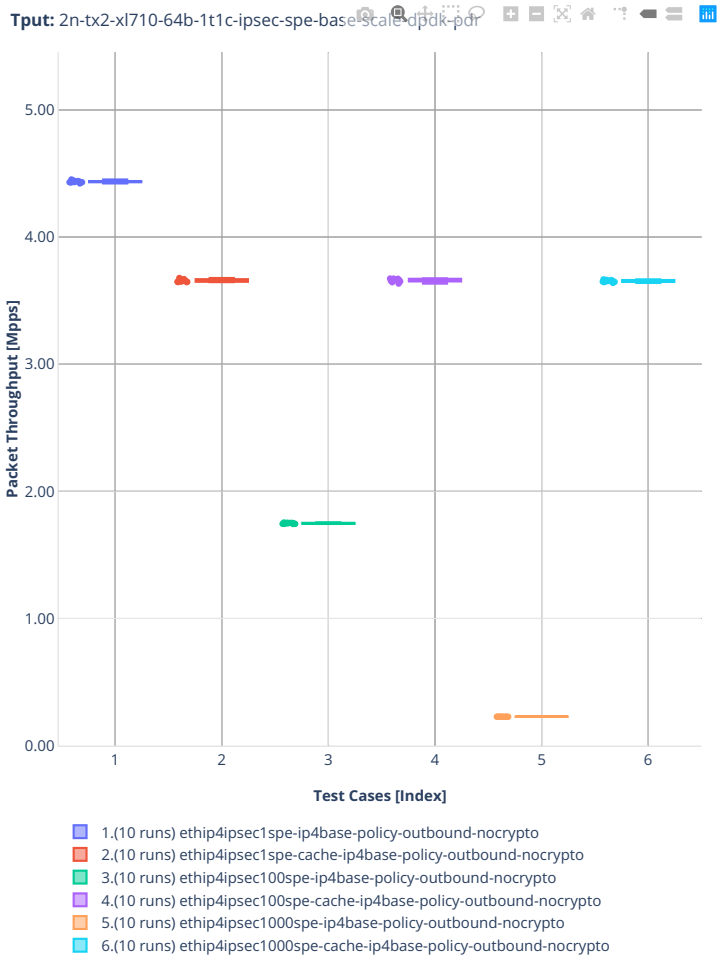
64b-ipsec-spe-ip4routing-base-scale



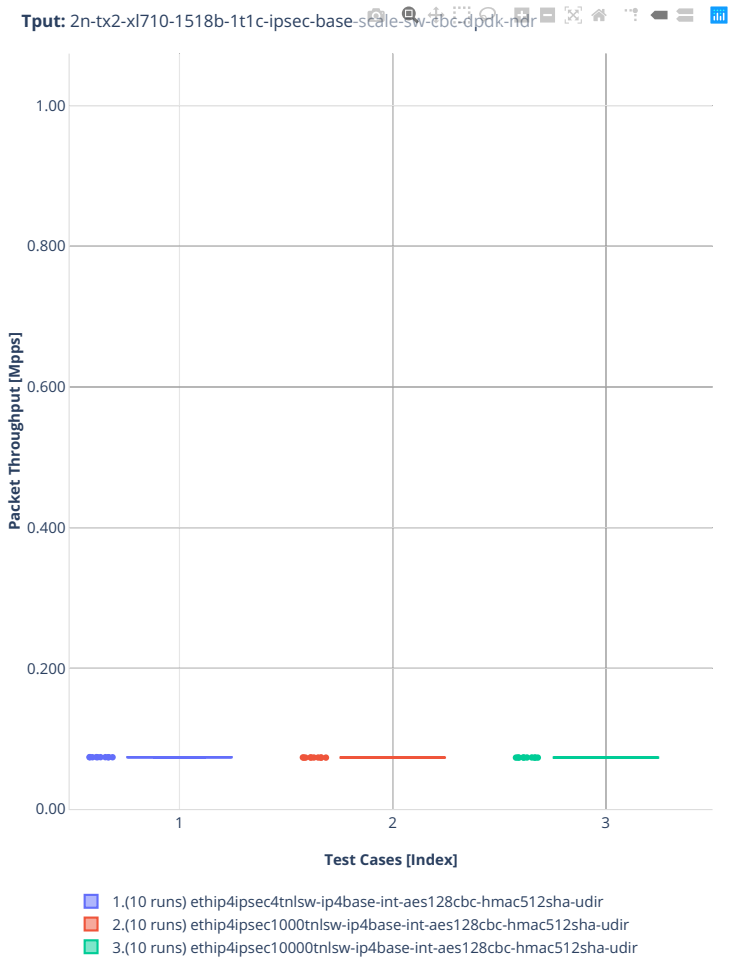

Tput: 2n-tx2-xl710-64b-1t1c-ipsec-base-scale-sw-policy-dpdk-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec1tnlsw-ip4base-policy-aes256gcm-udir
2.(10 runs) ethip4ipsec40tnlsw-ip4base-policy-aes256gcm-udir
3.(10 runs) ethip4ipsec1000tnlsw-ip4base-policy-aes256gcm-udir

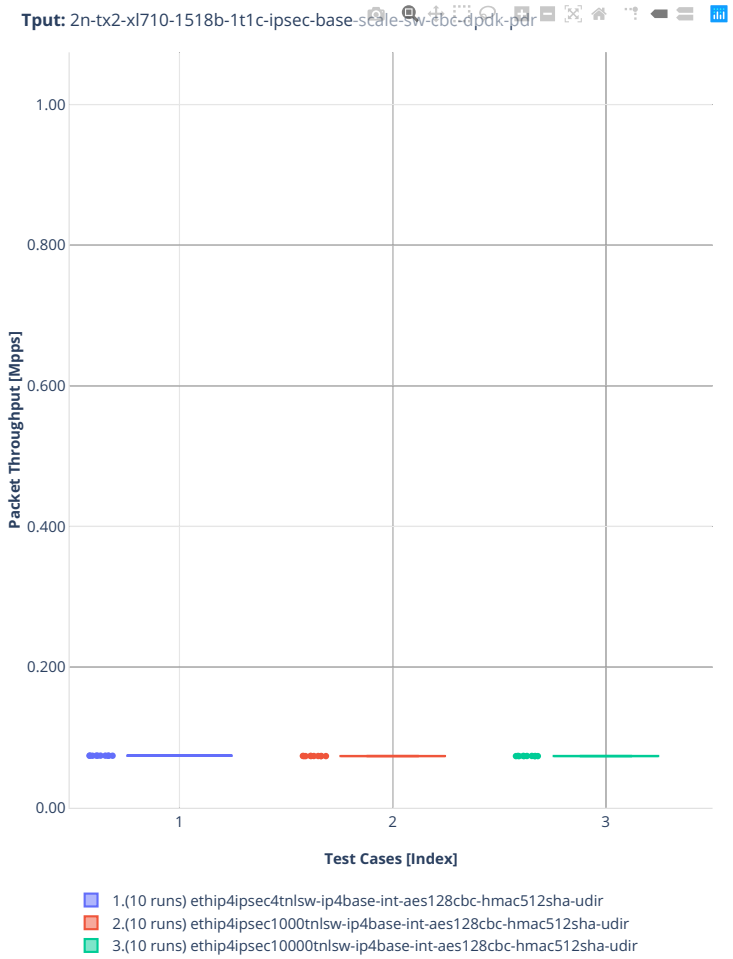
64b-ipsec-ip4routing-base-scale-sw



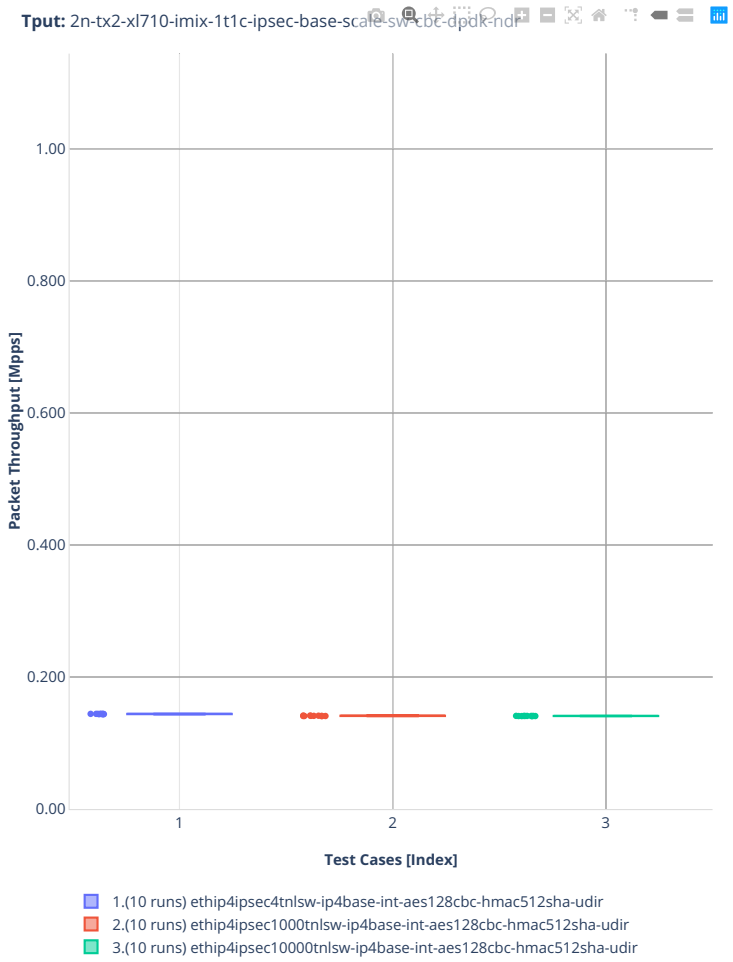

Tput: 2n-tx2-xl710-64b-1t1c-ipsec-spe-base-scale-dpdk-pdr
Packet Throughput [Mpps]
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4ipsec1spe-ip4base-policy-outbound-nocrypto
2.(10 runs) ethip4ipsec1spe-cache-ip4base-policy-outbound-nocrypto
3.(10 runs) ethip4ipsec100spe-ip4base-policy-outbound-nocrypto
4.(10 runs) ethip4ipsec100spe-cache-ip4base-policy-outbound-nocrypto
5.(10 runs) ethip4ipsec1000spe-ip4base-policy-outbound-nocrypto
6.(10 runs) ethip4ipsec1000spe-cache-ip4base-policy-outbound-nocrypto

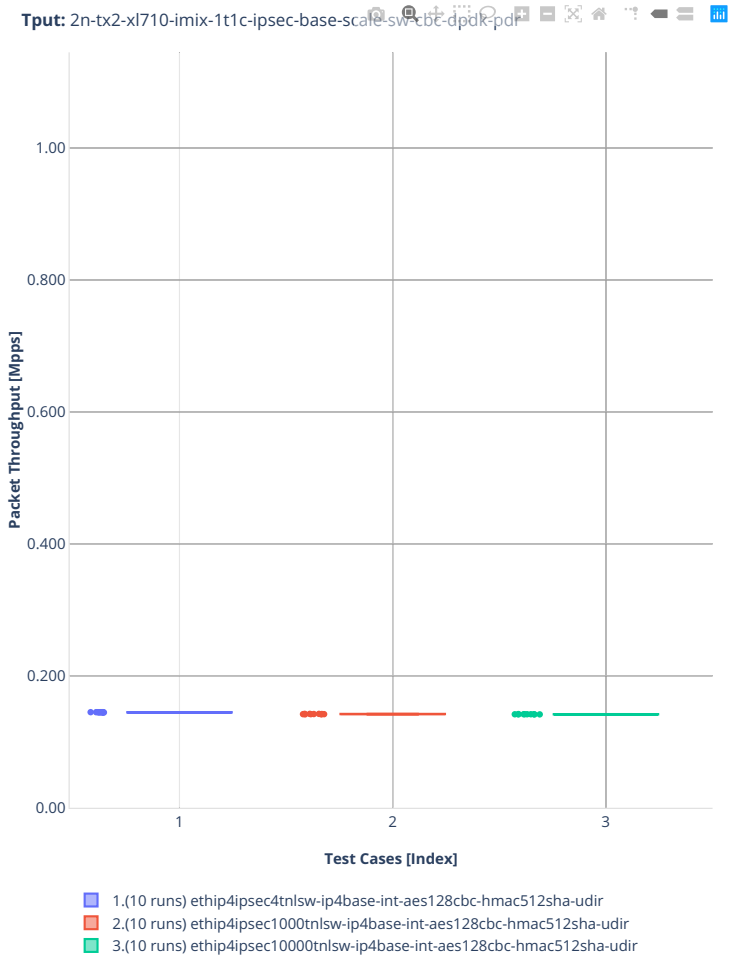
1518b-ipsec-ip4routing-base-scale-sw-cbc



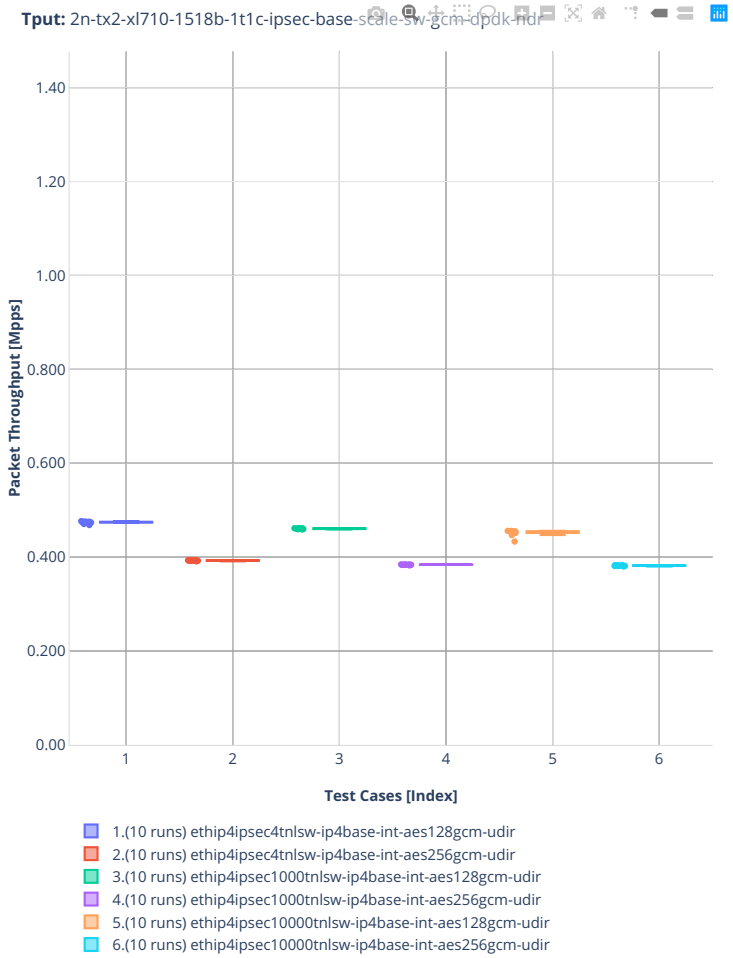

Tput: 2n-tx2-xl710-1518b-1t1c-ipsec-base-scale-sw-cbc-dpdk-pdr
Packet Throughput [Mpps]
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
2.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
3.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir

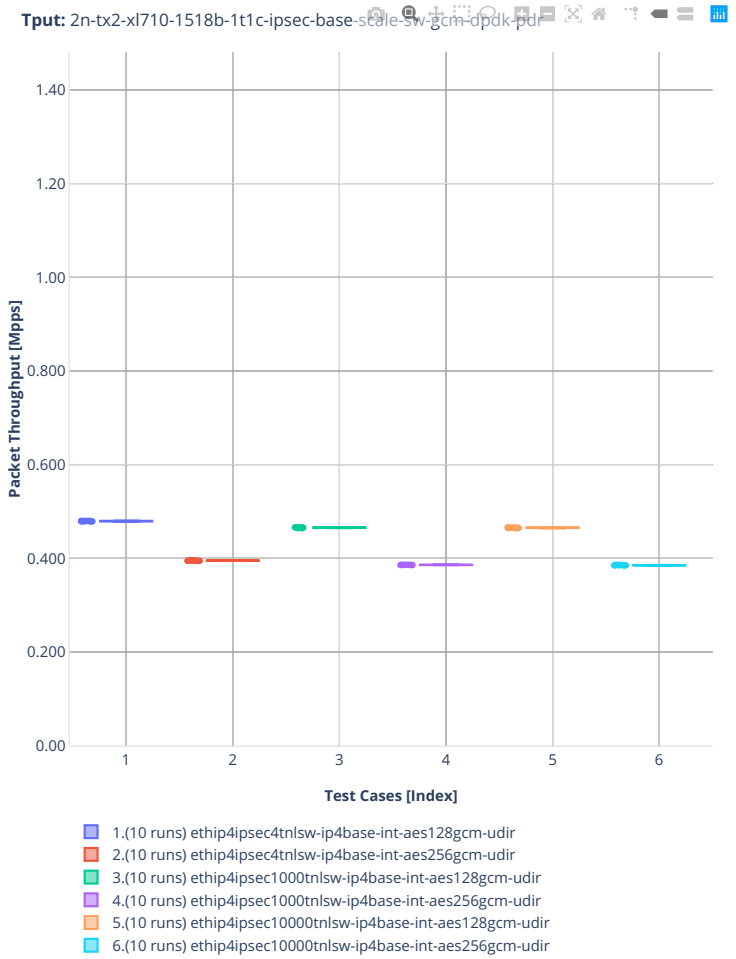
imix-ipsec-ip4routing-base-scale-sw-cbc



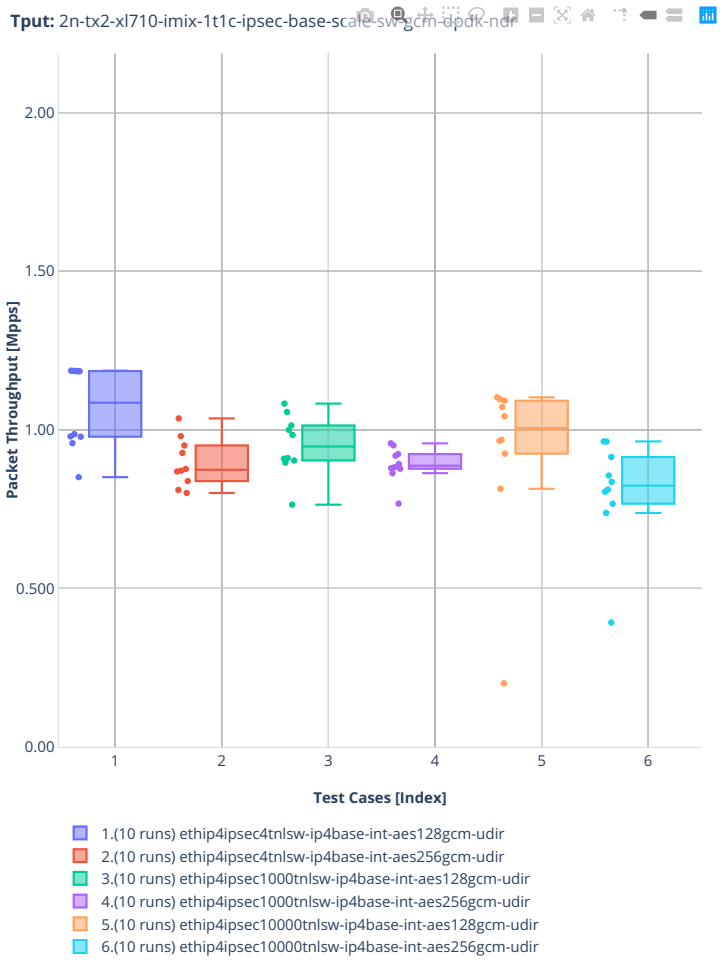

Tput: 2n-tx2-xl710-imix-1t1c-ipsec-base-scale-sw-cbc-dpdk-pdr
1.00
0.800
0.600
0.400
0.200
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
2.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
3.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir

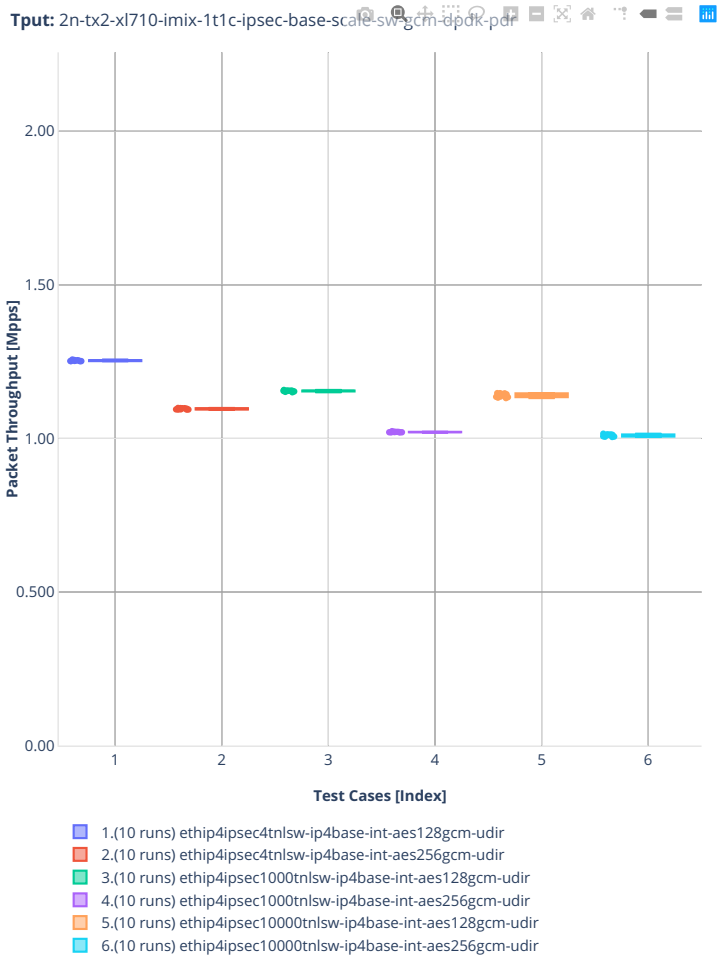
1518b-ipsec-ip4routing-base-scale-sw-gcm




Tput: 2n-tx2-xl710-1518b-1t1c-ipsec-base-scale-sw-gcm-dpdk-pdr
Packet Throughput [Mpps]
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128gcm-udir
2.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm-udir
3.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128gcm-udir
4.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm-udir
5.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128gcm-udir
6.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes256gcm-udir

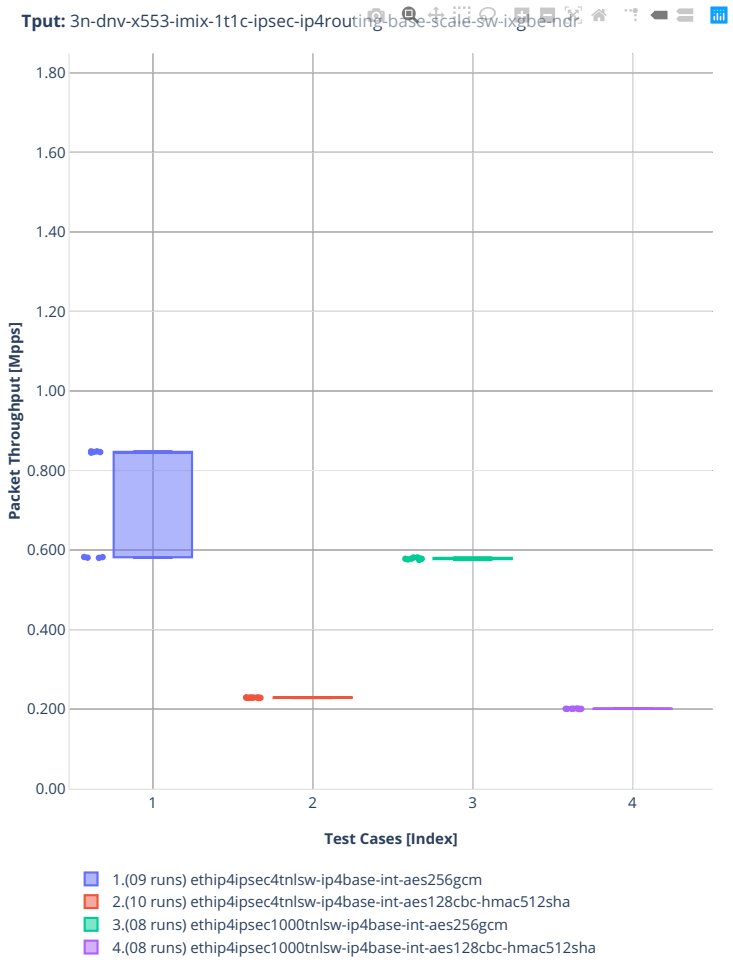
imix-ipsec-ip4routing-base-scale-sw-gcm

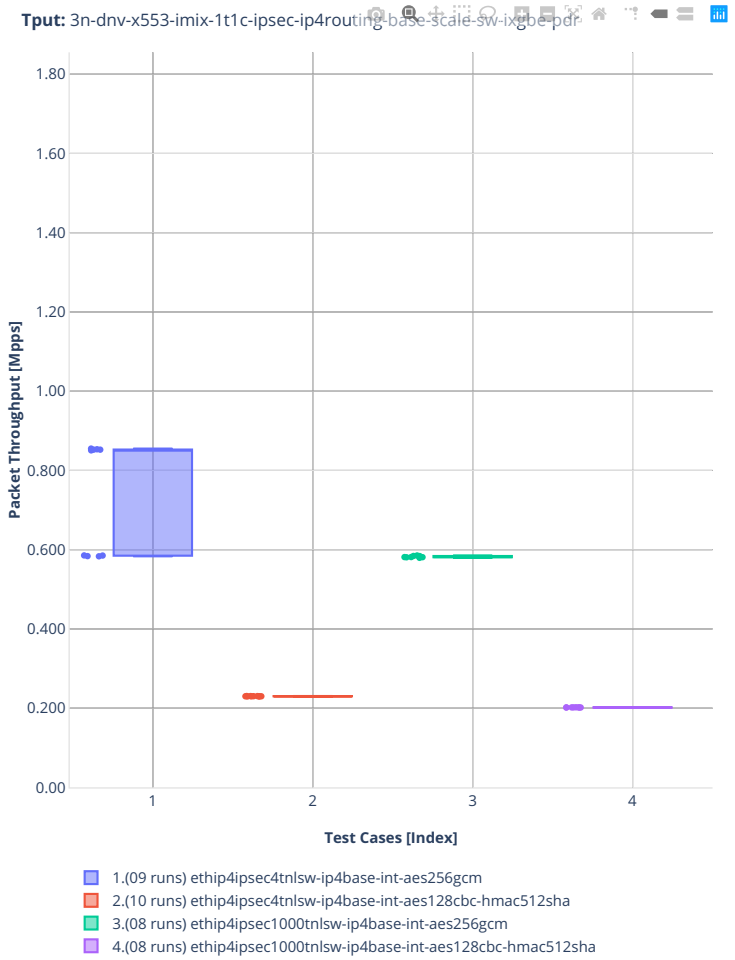



Tput: 2n-tx2-xl710-imix-1t1c-ipsec-base-scale-sw-gcm-dpdk-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128gcm-udir
2.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm-udir
3.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128gcm-udir
4.(10 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm-udir
5.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes128gcm-udir
6.(10 runs) ethip4ipsec10000tnlsw-ip4base-int-aes256gcm-udir

3n-dnv-x553

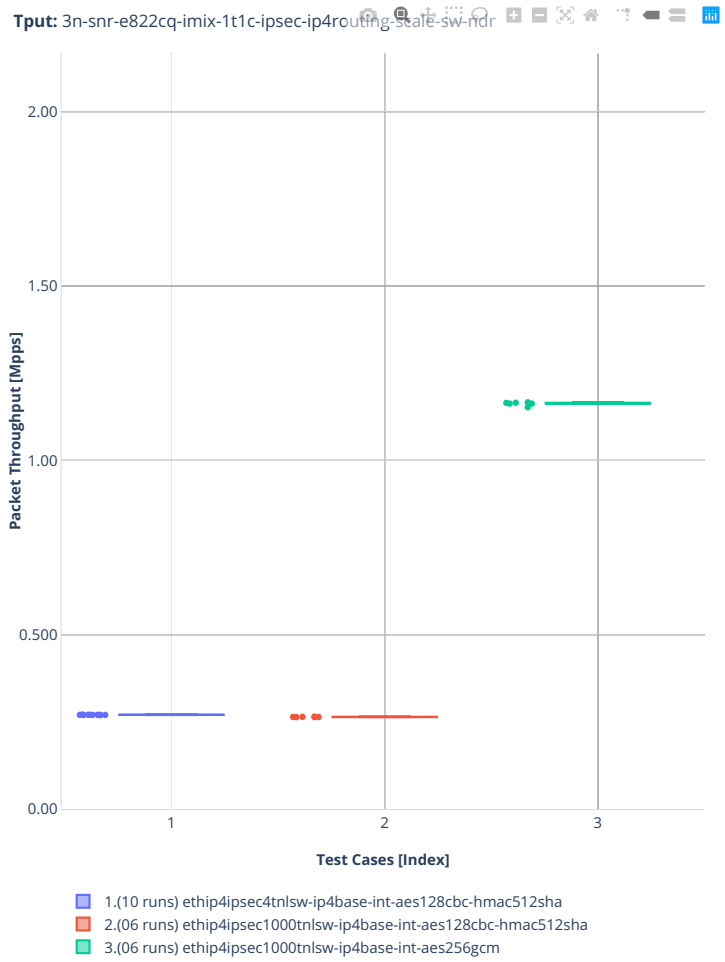
imix-1t1c-ipsec-ip4routing-base-scale-sw-ixgbe

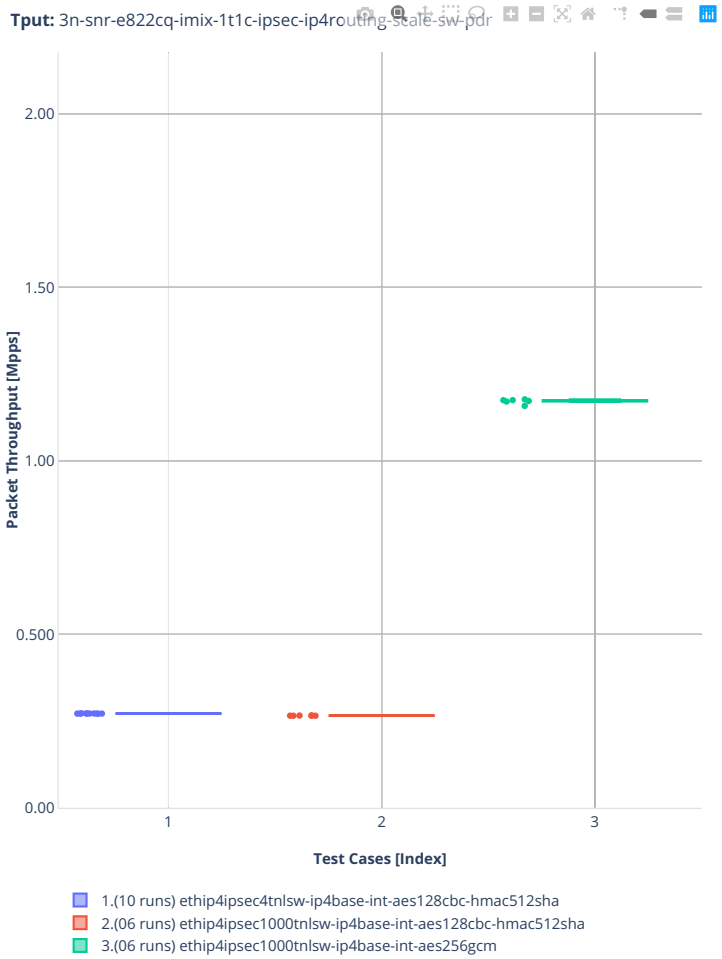



Tput: 3n-dnv-x553-imix-1t1c-ipsec-ip4routing-base-scale-sw-ixgbe-pdr
Packet Throughput [Mpps]
1.80
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
1
2
3
4
Test Cases [Index]
1.(09 runs) ethip4ipsec4tnlsw-ip4base-int-aes256gcm
2.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
3.(08 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
4.(08 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha

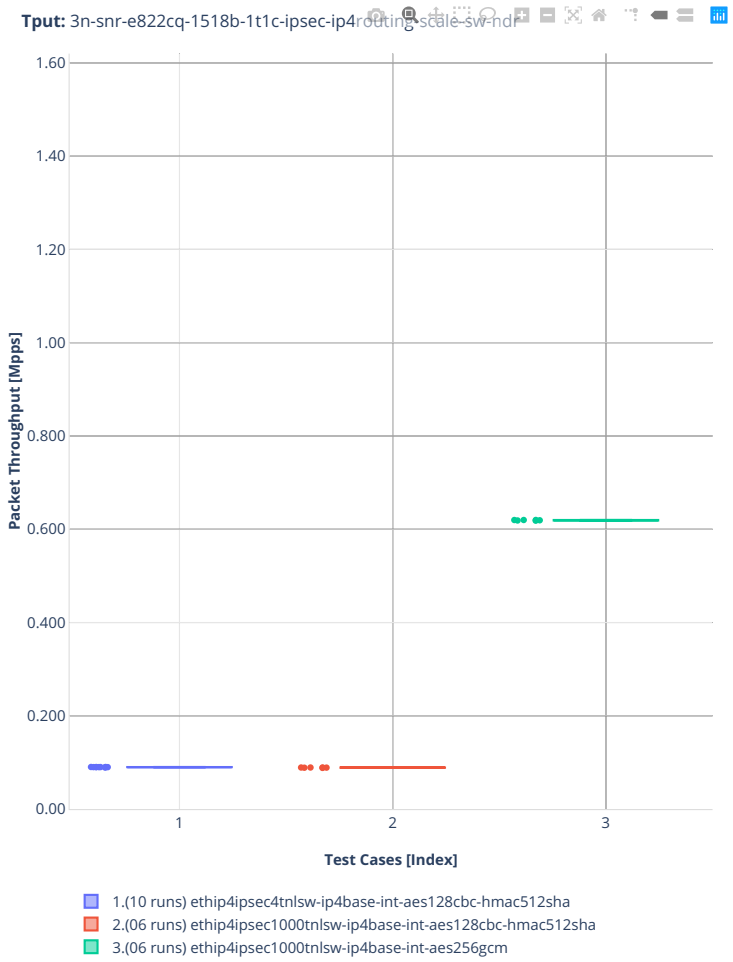
3n-snr-e822cq

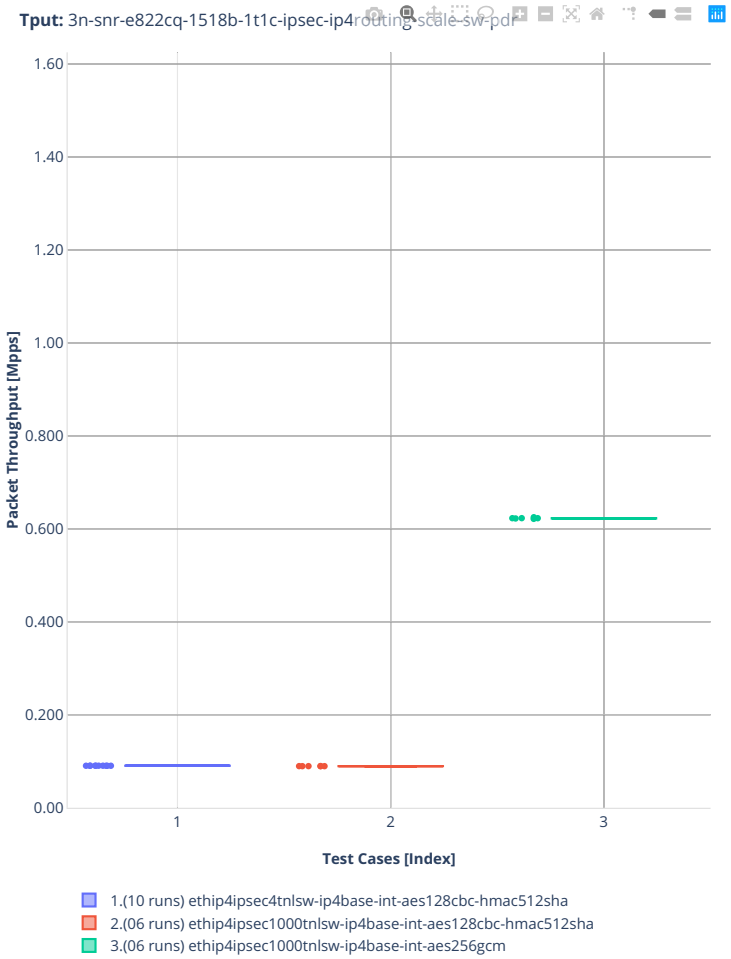
imix-1t1c-ipsec-ip4routing-scale-sw




Tput: 3n-snr-e822cq-imix-1t1c-ipsec-ip4routing-scale-sw-pdr
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
2.(06 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
3.(06 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm

1518b-1t1c-ipsec-ip4routing-scale-sw




Tput: 3n-snr-e822cq-1518b-1t1c-ipsec-ip4routing-scale-sw-pdr
1.60
1.40
1.20
1.00
0.800
0.600
0.400
0.200
0.00
Packet Throughput [Mpps]
1
2
3
Test Cases [Index]
1.(10 runs) ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
2.(06 runs) ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
3.(06 runs) ethip4ipsec1000tnlsw-ip4base-int-aes256gcm

2.4 Speedup Multi-Core

Speedup Multi-Core throughput graphs are generated by multiple executions of the same performance tests across physical testbeds hosted LF FD.io labs: 2n-icx, 3n-icx, 2n-aws, 2n-clx, 2n-zn2, 3n-alt, 3n-tsh, 2n-tx2, 2n-dnv, 3n-dnv. Grouped bars illustrate the 64B/78B packet throughput speedup ratio for 2- and 4-core multi-threaded VPP configurations relative to 1-core configurations.

Additional information about graph data:

1. **Graph Title**: describes tested packet path, testbed topology, processor model, NIC model, packet size used by data plane workers and indication of VPP DUT configuration.
2. **X-axis Labels**: number of cores.
3. **Y-axis Labels**: measured Packets Per Second [pps] throughput values.
4. **Graph Legend**: lists CSIT test suites executed to generate graphed test results.
5. **Hover Information**: lists number of runs executed, specific test substring, mean value of the measured packet throughput, calculated perfect throughput value, difference between measured and perfect values and relative speedup value.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[123], build logs from FD.io vpp performance job 3n-icx[124], build logs from FD.io vpp performance job 2n-aws[125], build logs from FD.io vpp performance job 2n-clx[126], build logs from FD.io vpp performance job 2n-zn2[127], build logs from FD.io vpp performance job 3n-alt[128], build logs from FD.io vpp performance job 3n-tsh[129], build logs from FD.io vpp performance job 2n-tx2[130], **`build logs from FD.io vpp performance job 3n-snr`_**, build logs from FD.io vpp performance job 2n-dnv[131] and build logs from FD.io vpp performance job 3n-dnv[132] with RF result files csit-vpp-perf-2210-*.zip archived here. Required per test case data set size is **10**, but for VPP tests the actual size varies per test case and is <=10.

---

[123] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[124] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-icx
[125] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-aws
[126] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx
[127] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-zn2
[128] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-alt
[129] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-tsh
[130] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-tx2
[131] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-dnv
[132] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-dnv

2.4.1 L2 Ethernet Switching

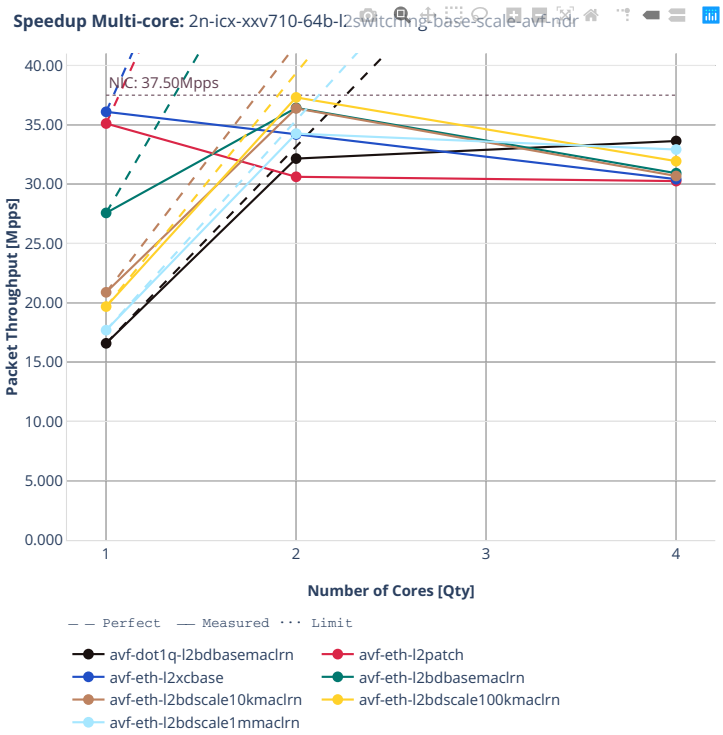
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 64B performance tests with VPP L2 Ethernet switching, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

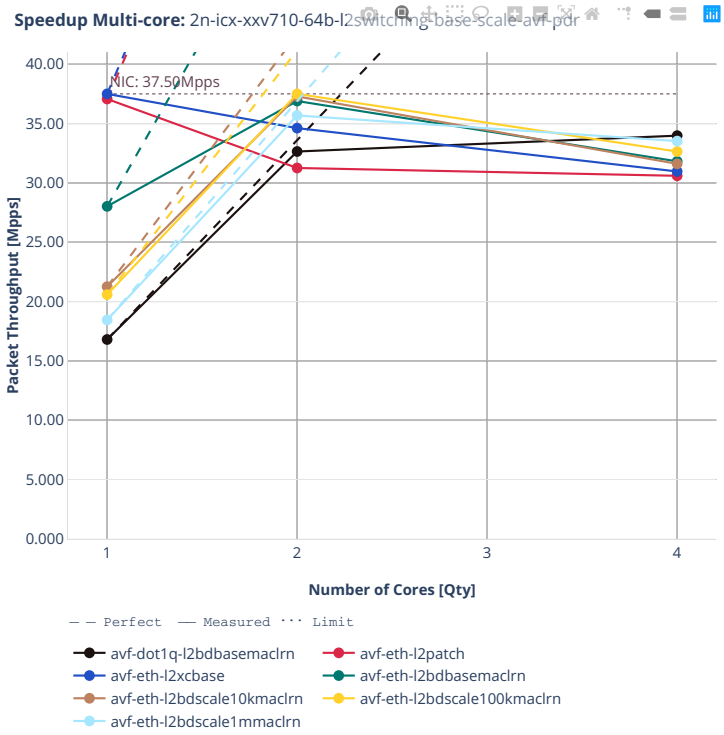
CSIT source code for the test cases used for plots can be found in CSIT git repository[133].

[133] https://git.fd.io/csit/tree/tests/vpp/perf/l2?h=rls2210

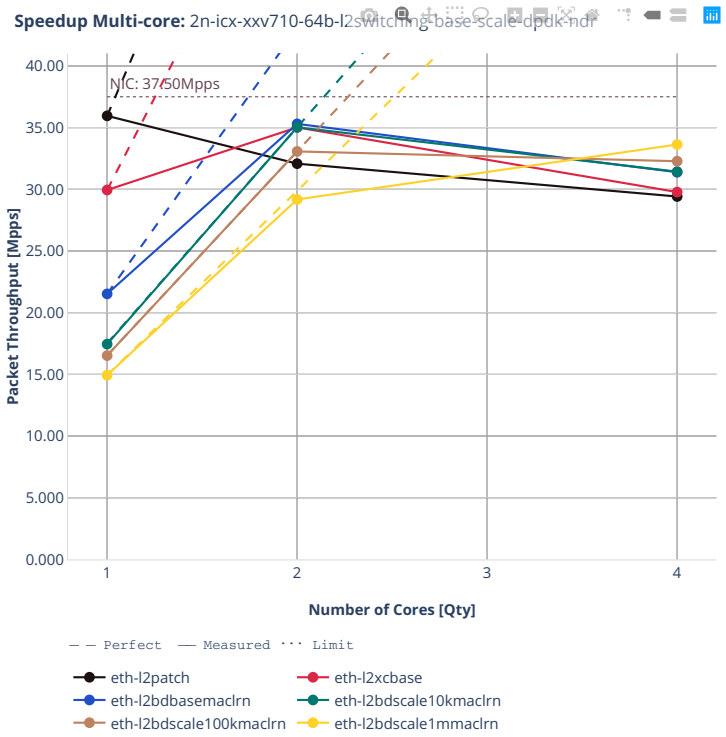
2n-icx-xxv710

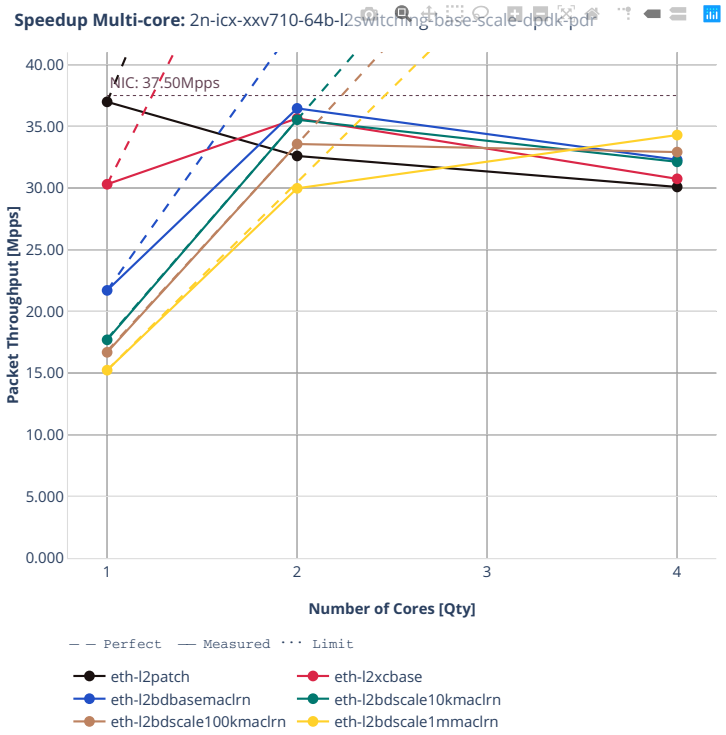
64b-l2switching-base-scale-avf




Speedup Multi-core: 2n-icx-xxv710-64b-l2switching-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-dot1q-l2bdbasemaclrn
avf-eth-l2patch
avf-eth-l2xcbase
avf-eth-l2bdbasemaclrn
avf-eth-l2bdscale10kmaclrn
avf-eth-l2bdscale100kmaclrn
avf-eth-l2bdscale1mmaclrn

64b-l2switching-base-scale-dpdk



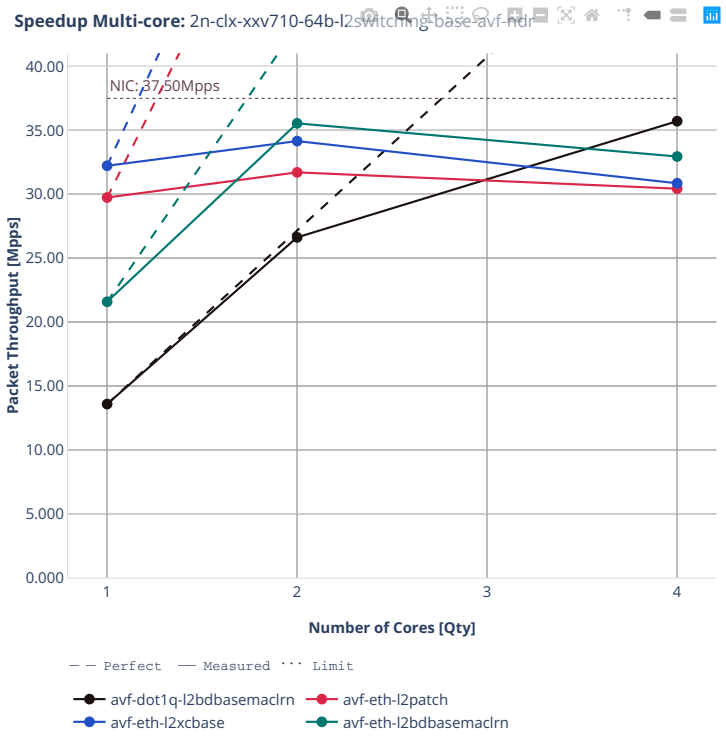

Speedup Multi-core: 2n-icx-xxv710-64b-l2switching-base-scale-dpdk-pdr
MIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn
eth-l2bdscale100kmaclrn
eth-l2bdscale1mmaclrn

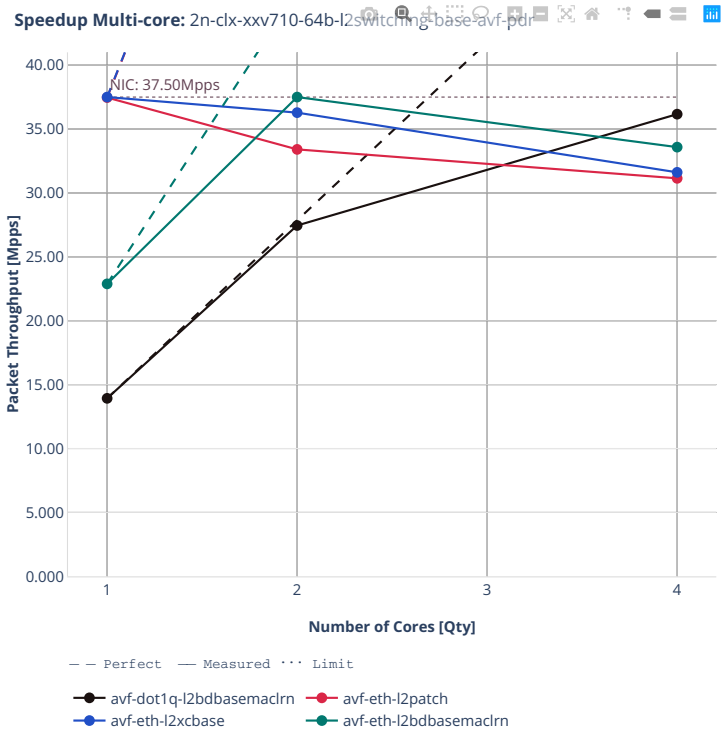
3n-icx-xxv710

64b-l2switching-base

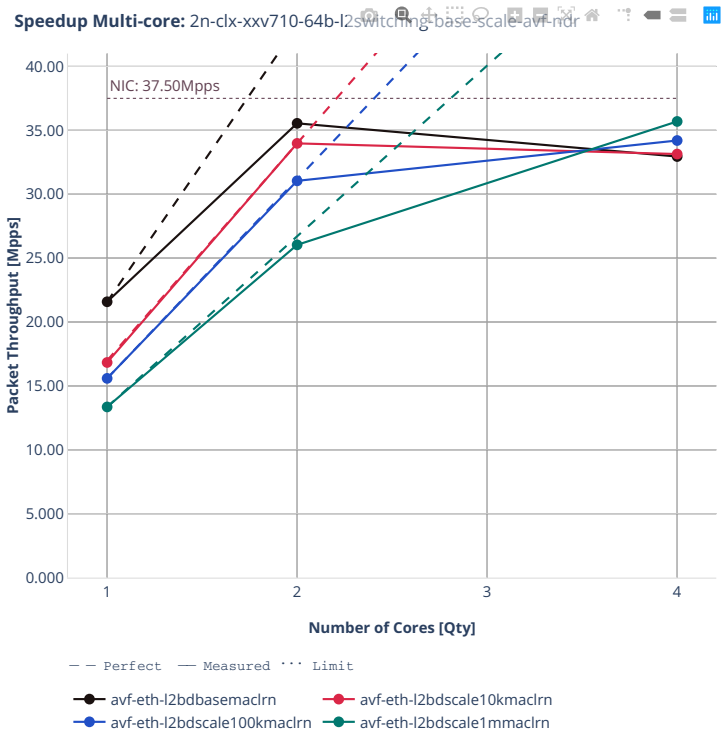
2n-clx-xxv710

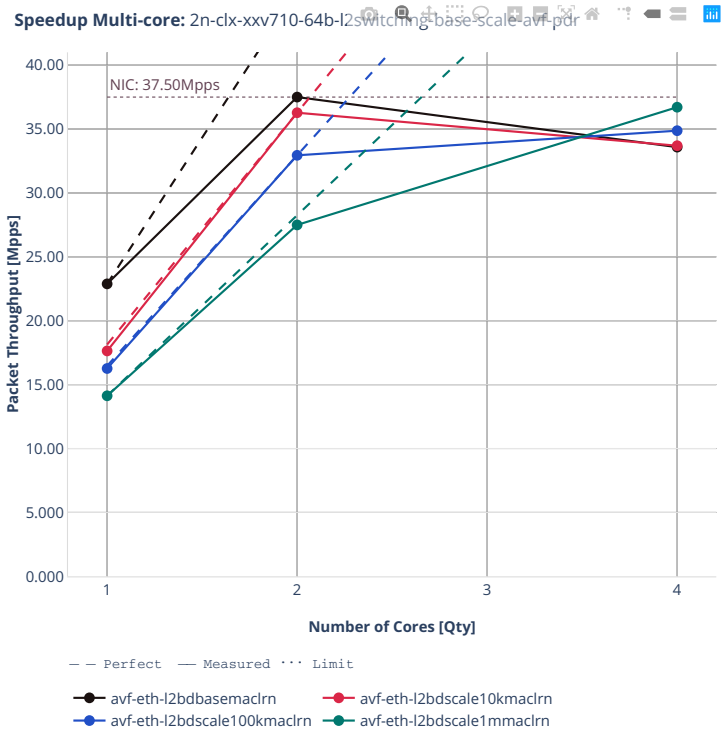
64b-l2switching-base-avf



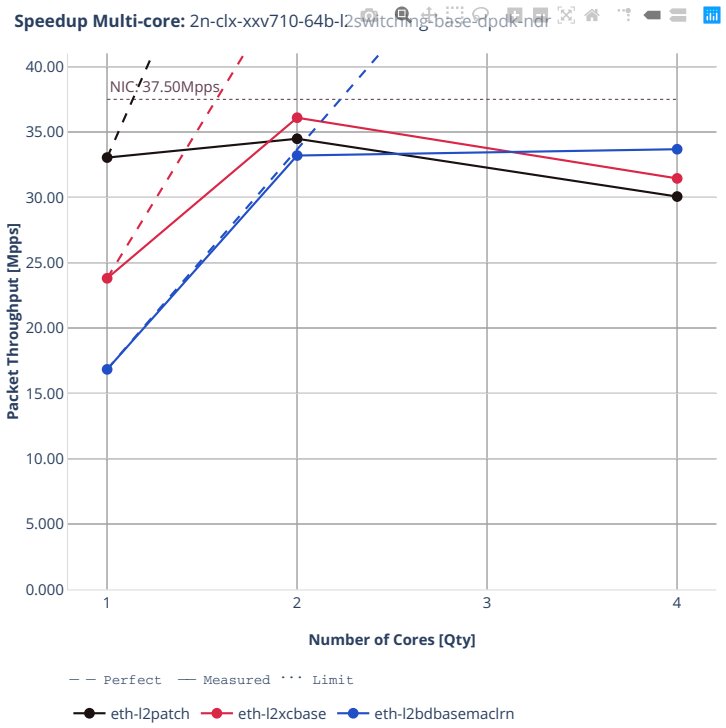

Speedup Multi-core: 2n-clx-xxv710-64b-l2switching-base-avf-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-dot1q-l2bdbasemaclrn
avf-eth-l2patch
avf-eth-l2xcbase
avf-eth-l2bdbasemaclrn

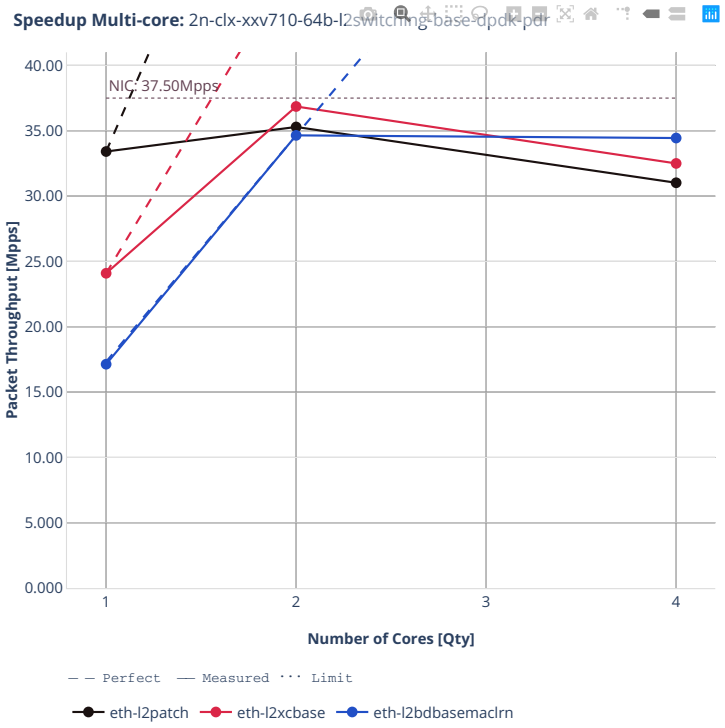
64b-l2switching-base-scale-avf



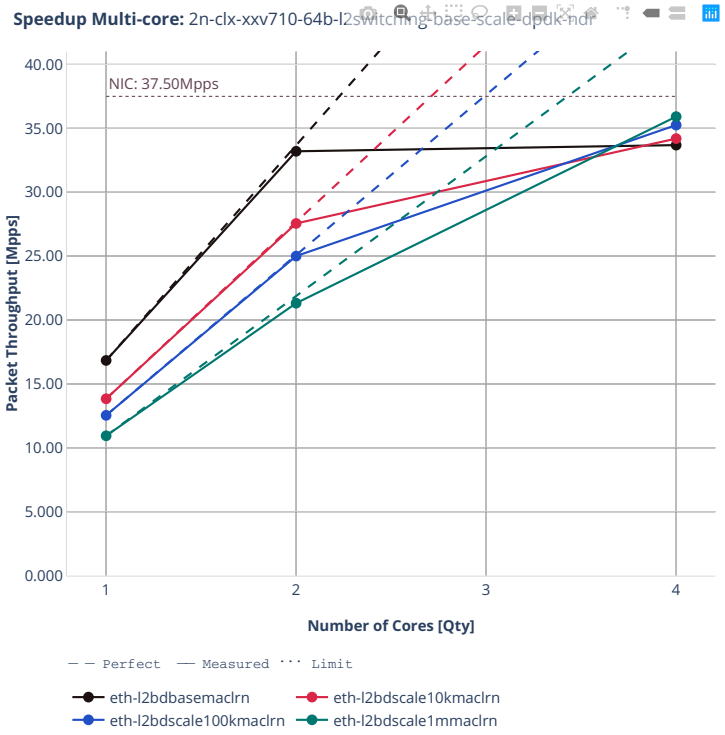

Speedup Multi-core: 2n-clx-xxv710-64b-l2switching-base-scale-avf-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2bdbasemaclrn
avf-eth-l2bdscale10kmaclrn
avf-eth-l2bdscale100kmaclrn
avf-eth-l2bdscale1mmaclrn

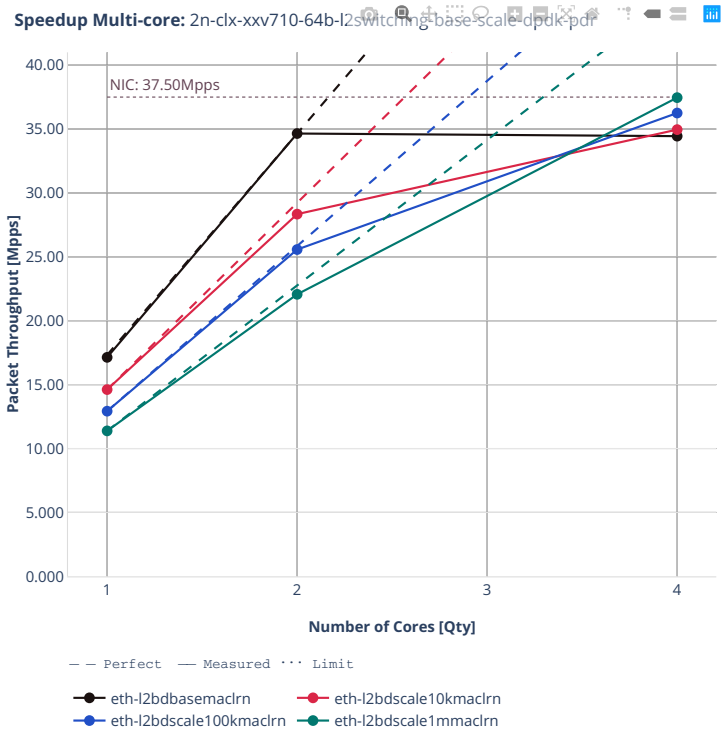
64b-l2switching-base-dpdk




Speedup Multi-core: 2n-clx-xxv710-64b-l2switching-base-dpdk-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn

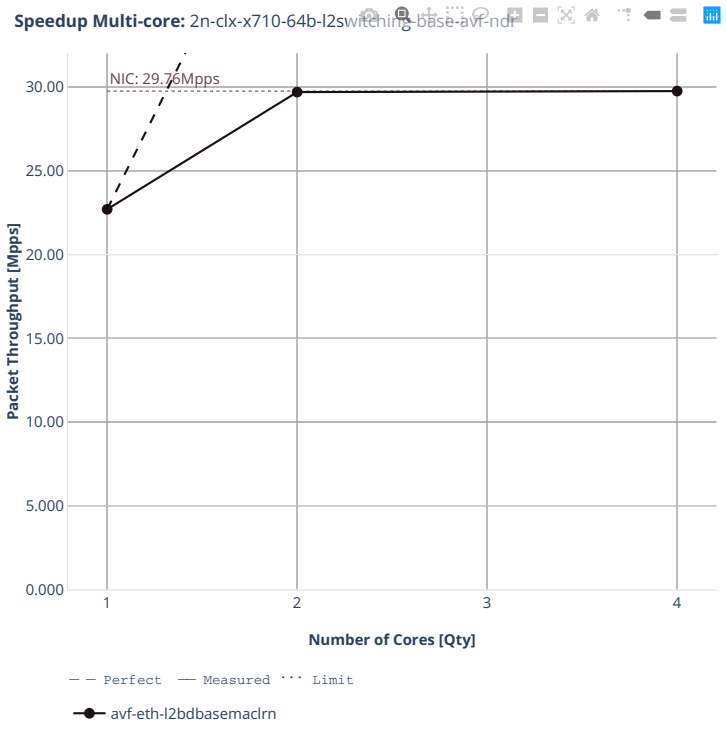
64b-l2switching-base-scale-dpdk

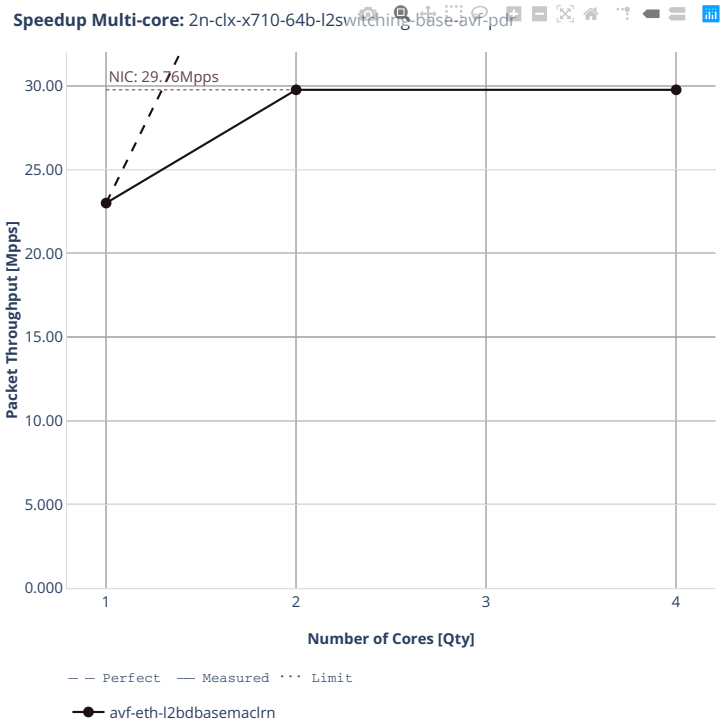



Speedup Multi-core: 2n-clx-xxv710-64b-l2switching-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn
eth-l2bdscale100kmaclrn
eth-l2bdscale1mmaclrn

2n-clx-x710

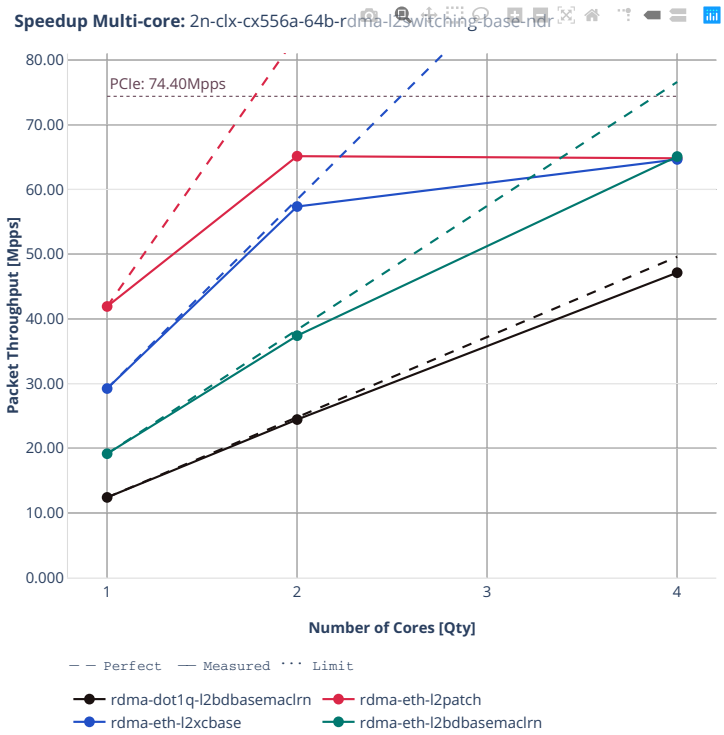
64b-l2switching-base-avf

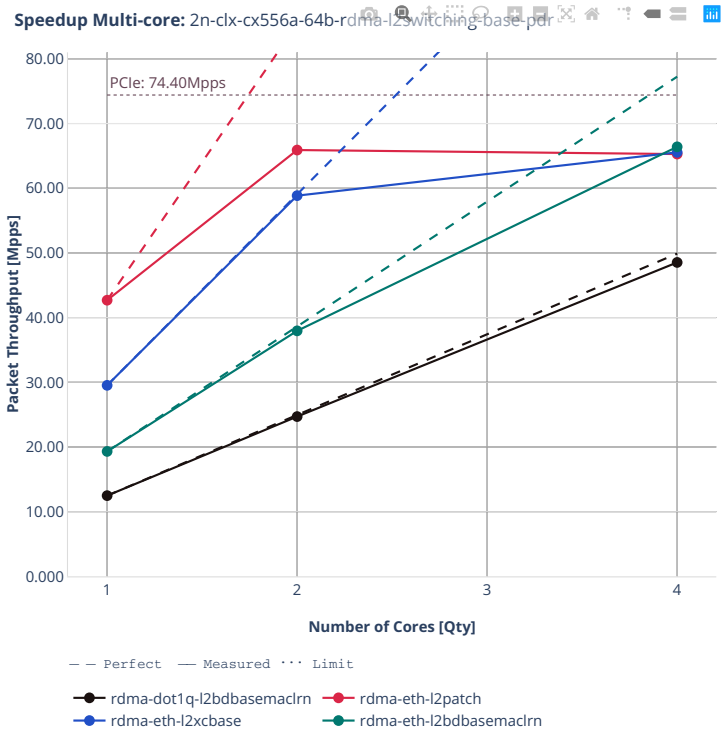



Speedup Multi-core: 2n-clx-x710-64b-l2switching-base-avf-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2bdbasemaclrn

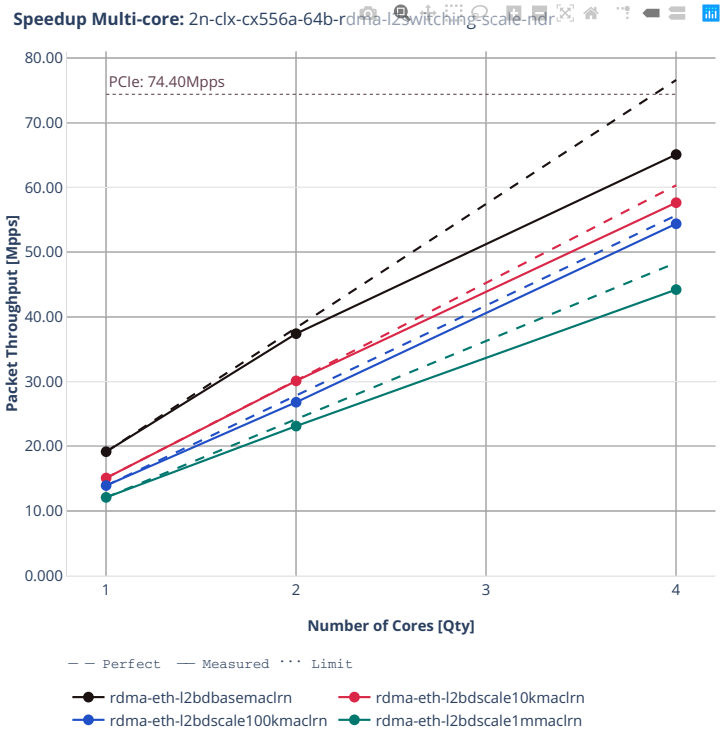
2n-clx-cx556a

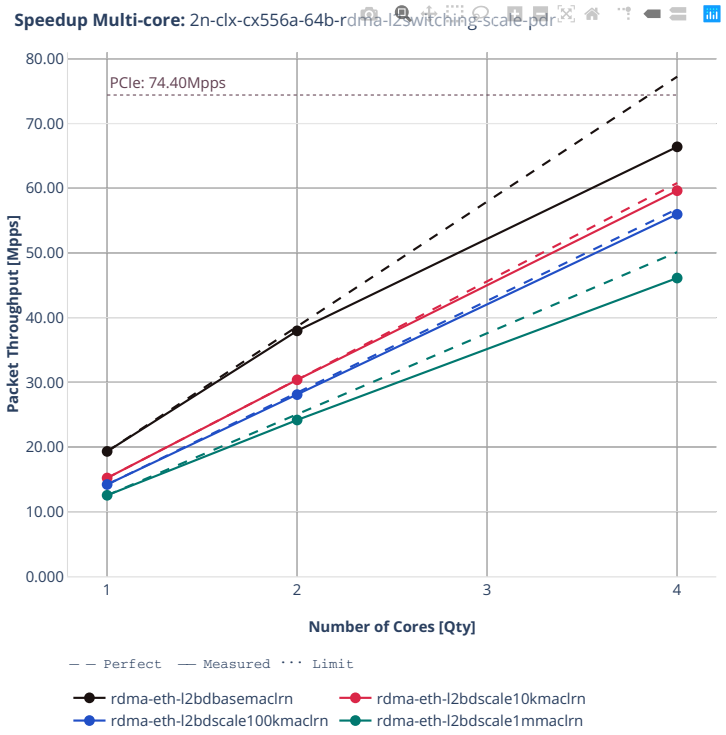
64b-l2switching-base-rdma-core




Speedup Multi-core: 2n-clx-cx556a-64b-rdma-l2switching-base-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-dot1q-l2bdbasemaclrn
rdma-eth-l2patch
rdma-eth-l2xcbase
rdma-eth-l2bdbasemaclrn

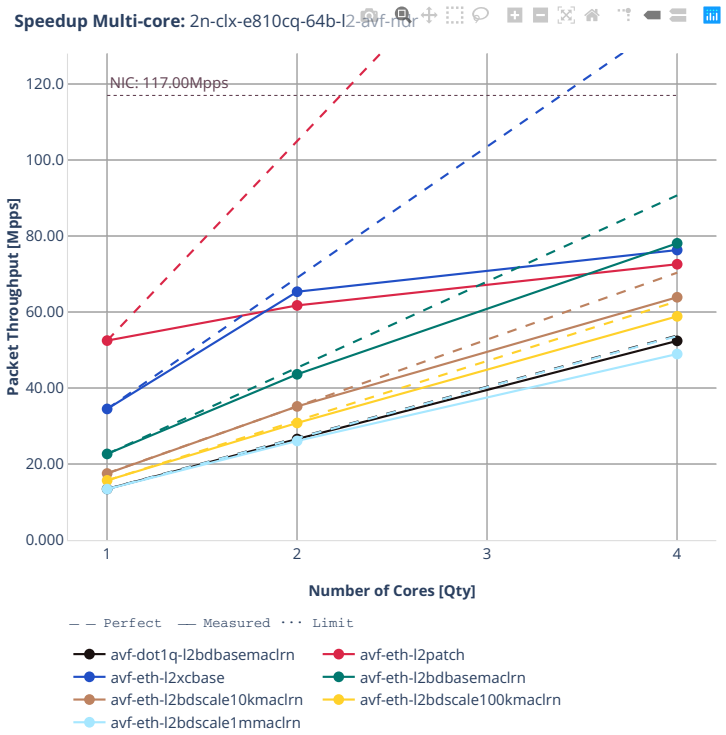
64b-l2switching-scale

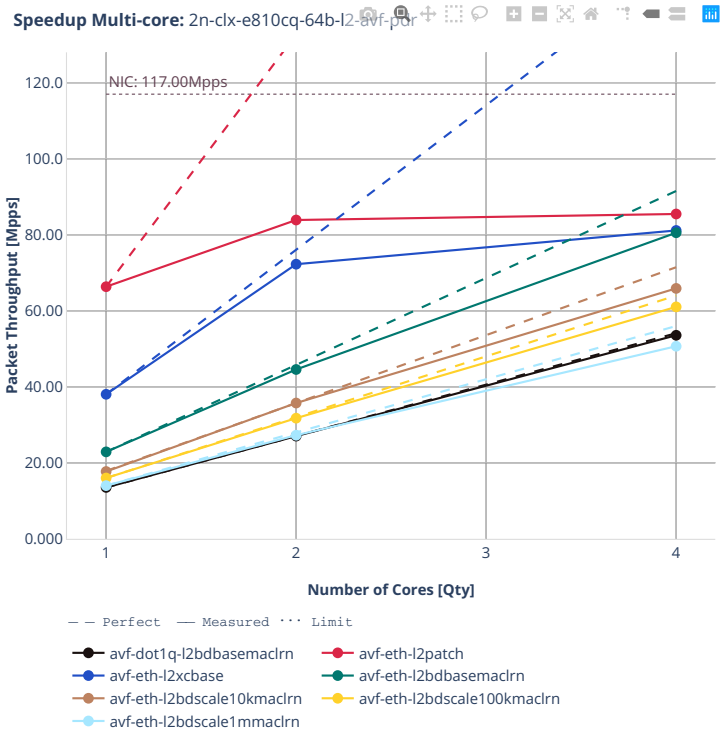



Speedup Multi-core: 2n-clx-cx556a-64b-rdma-l2switching-scale-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-eth-l2bdbasemaclrn
rdma-eth-l2bdscale10kmaclrn
rdma-eth-l2bdscale100kmaclrn
rdma-eth-l2bdscale1mmaclrn

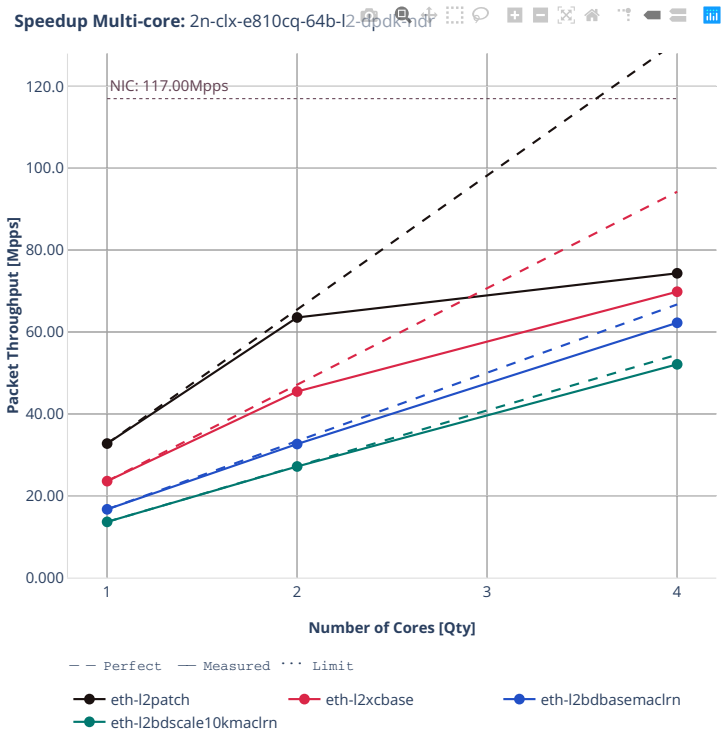
2n-clx-e810cq

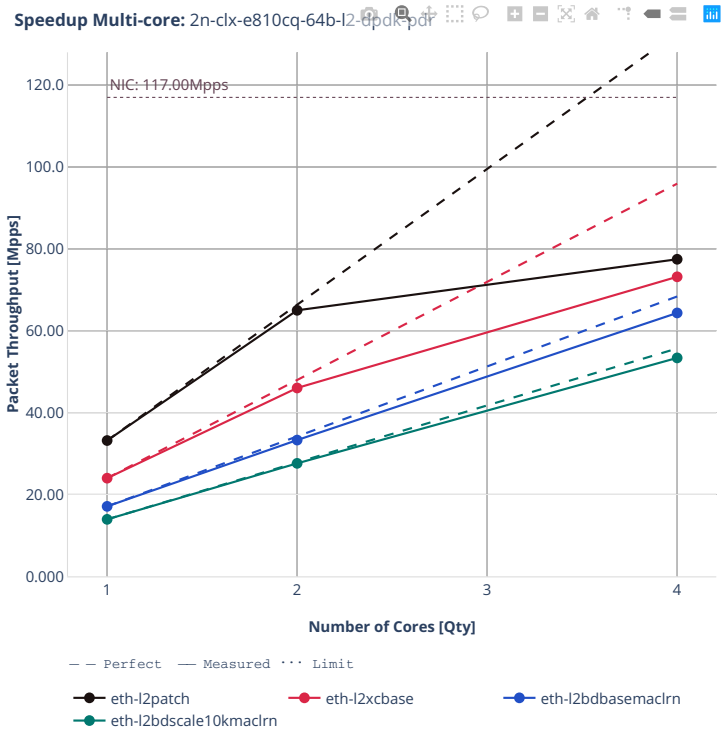
64b-l2switching-avf




Speedup Multi-core: 2n-clx-e810cq-64b-l2-avf-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-dot1q-l2bdbasemaclrn
avf-eth-l2patch
avf-eth-l2xcbase
avf-eth-l2bdbasemaclrn
avf-eth-l2bdscale10kmaclrn
avf-eth-l2bdscale100kmaclrn
avf-eth-l2bdscale1mmaclrn

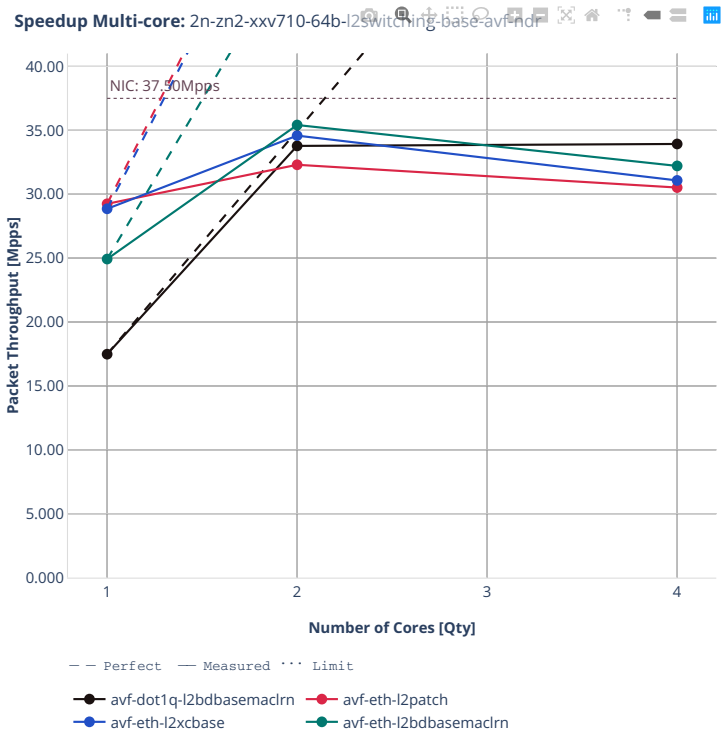
64b-l2switching-dpdk

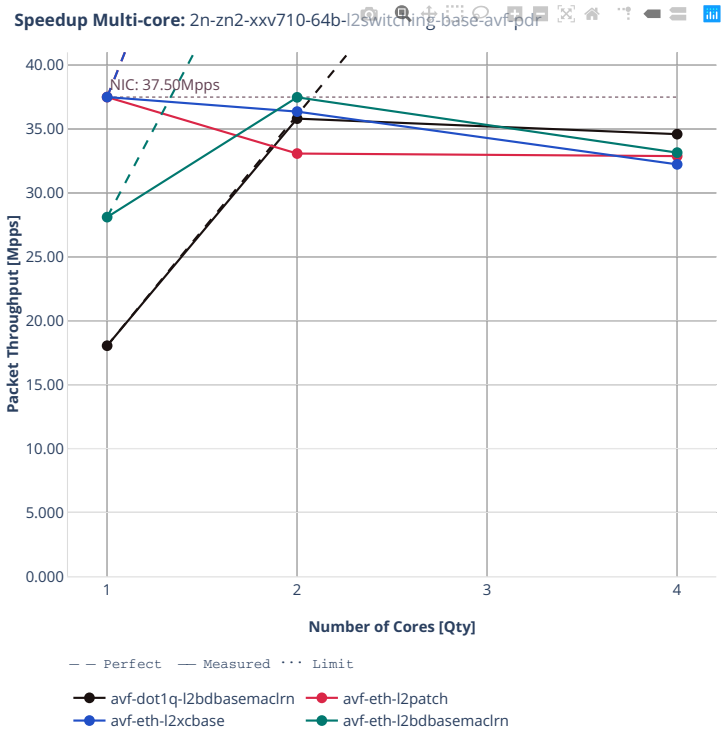



Speedup Multi-core: 2n-clx-e810cq-64b-l2-dpdk-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn

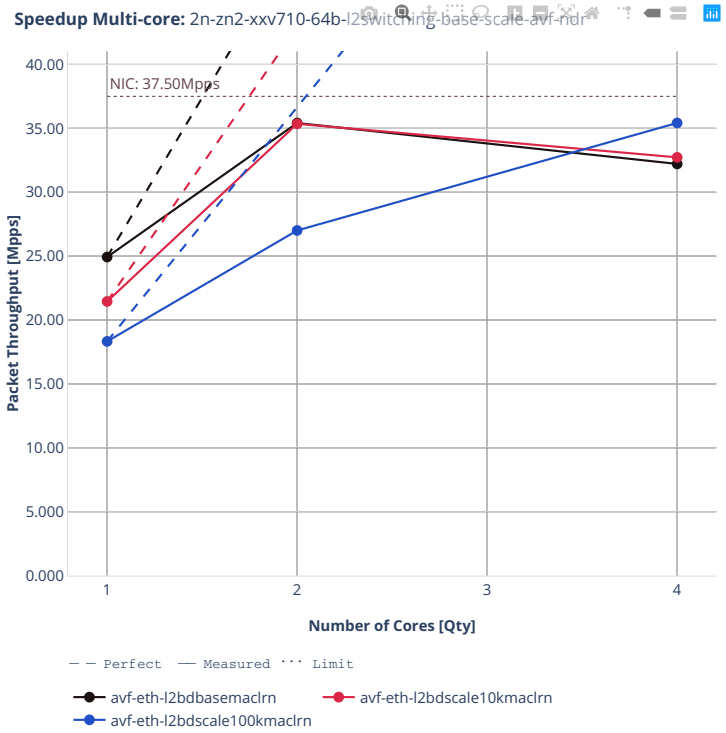
2n-zn2-xxv710

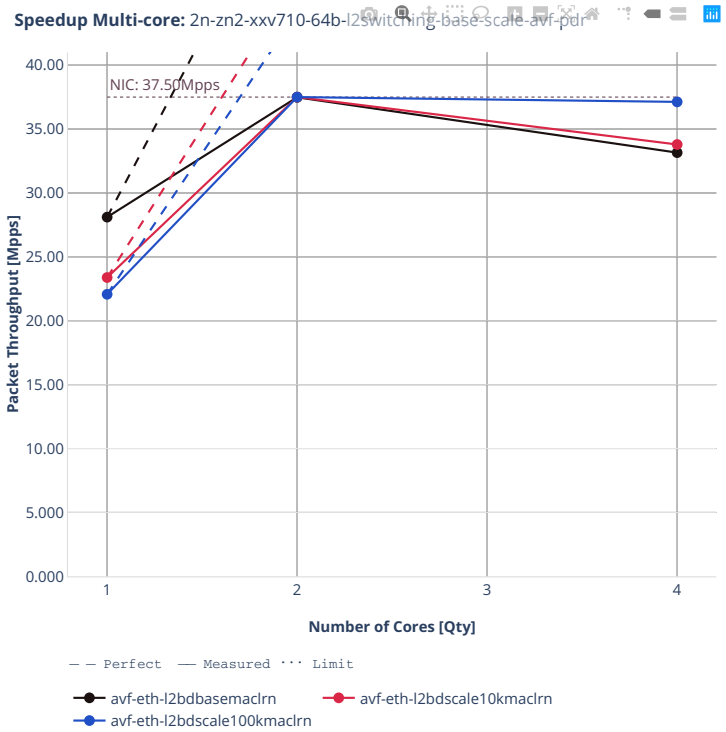
64b-l2switching-base-avf



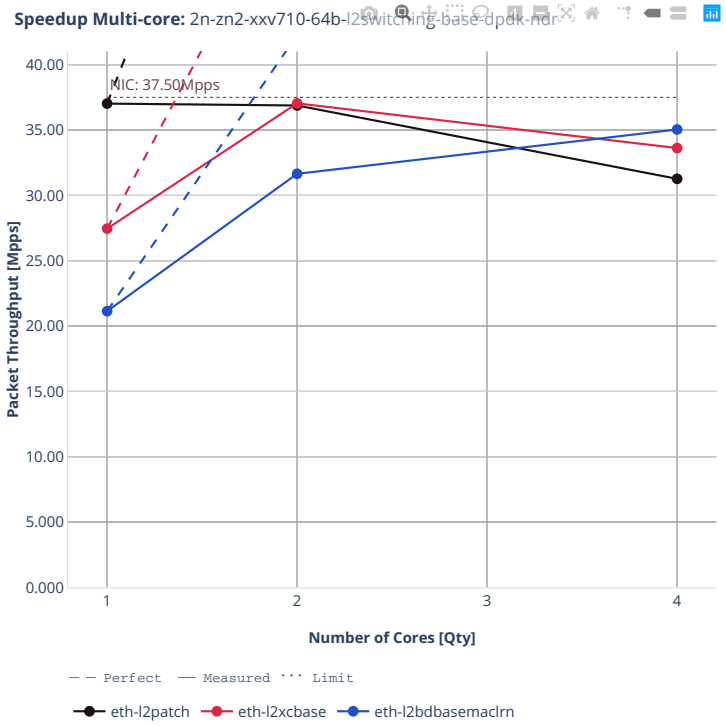

Speedup Multi-core: 2n-zn2-xxv710-64b-l2switching-base-avf-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-dot1q-l2bdbasemaclrn
avf-eth-l2patch
avf-eth-l2xcbase
avf-eth-l2bdbasemaclrn

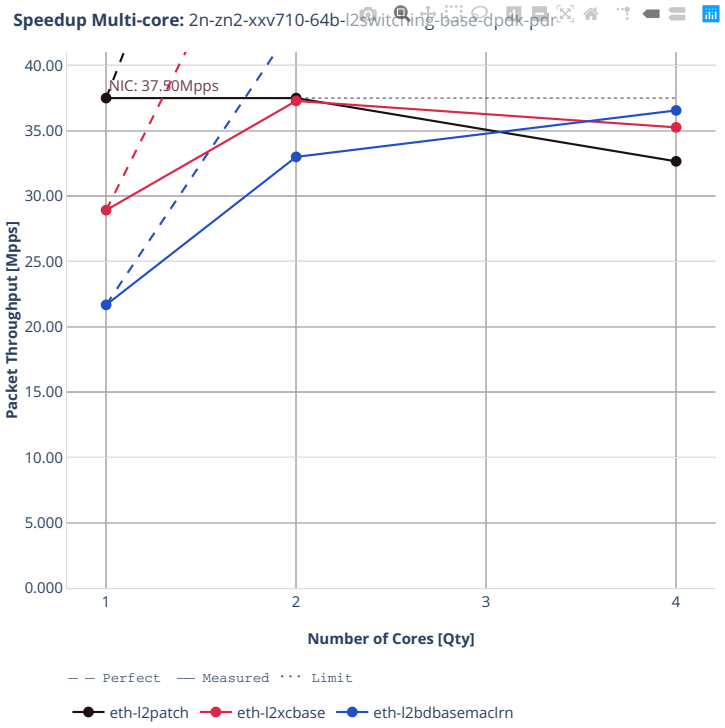
64b-l2switching-base-scale-avf



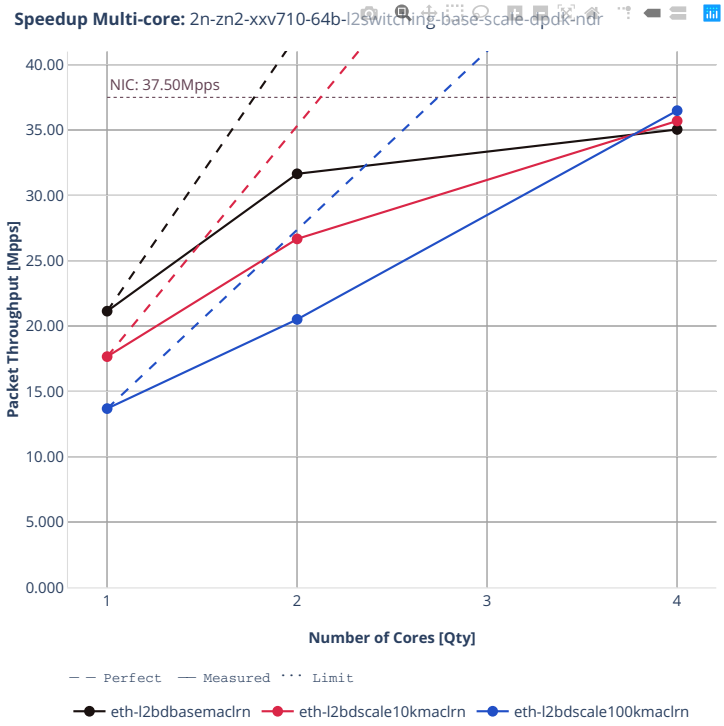

Speedup Multi-core: 2n-zn2-xxv710-64b-l2switching-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2bdbasemaclrn
avf-eth-l2bdscale10kmaclrn
avf-eth-l2bdscale100kmaclrn

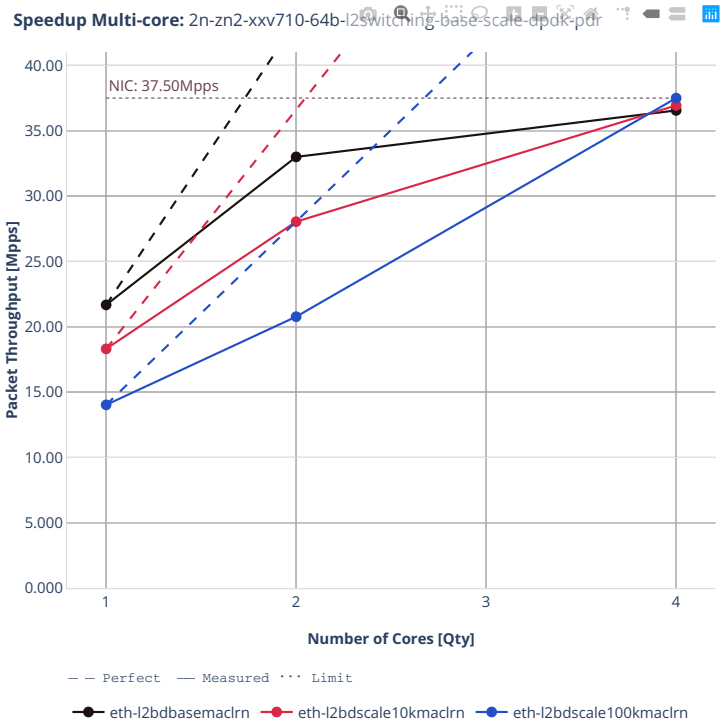
64b-l2switching-base-dpdk




Speedup Multi-core: 2n-zn2-xxv710-64b-l2switching-base-dpdk-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn

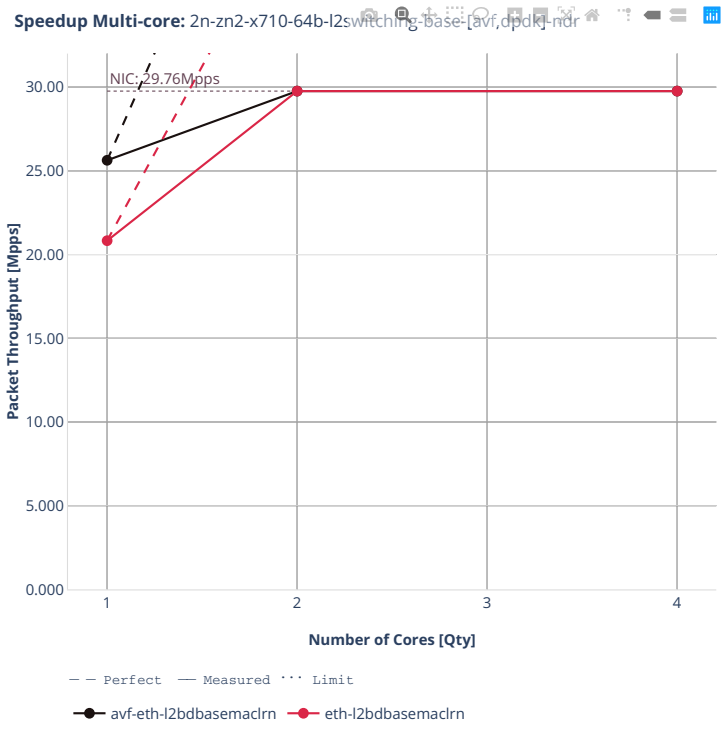
64b-l2switching-base-scale-dpdk

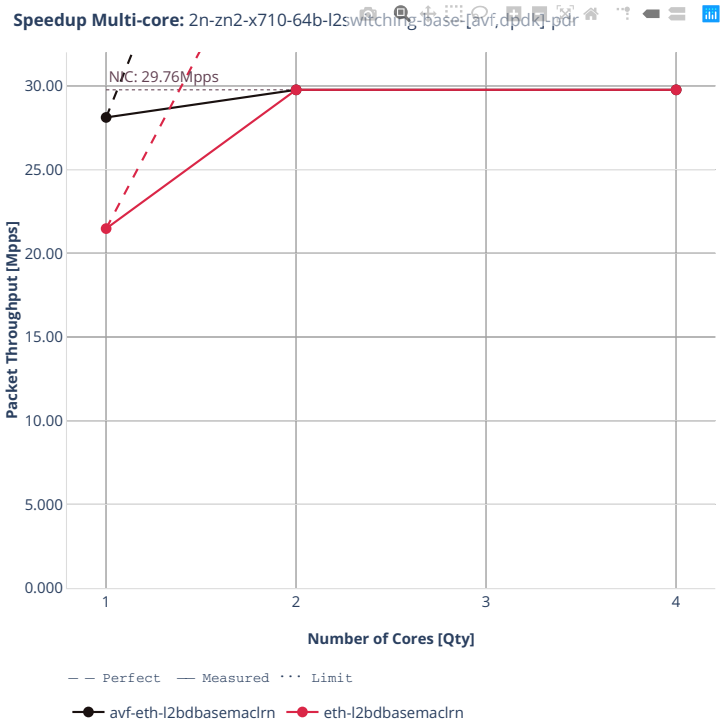



Speedup Multi-core: 2n-zn2-xxv710-64b-l2switching-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn
eth-l2bdscale100kmaclrn

2n-zn2-x710

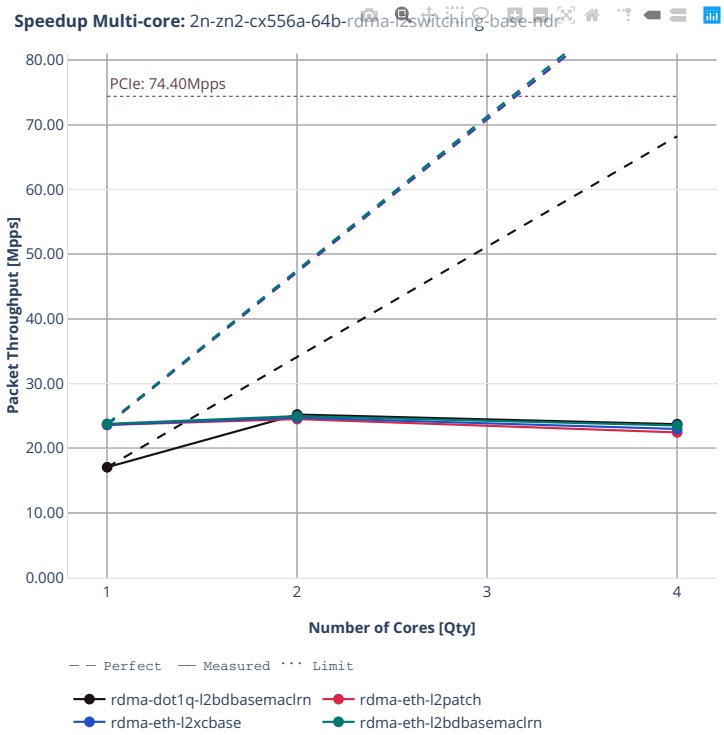
64b-l2switching-base

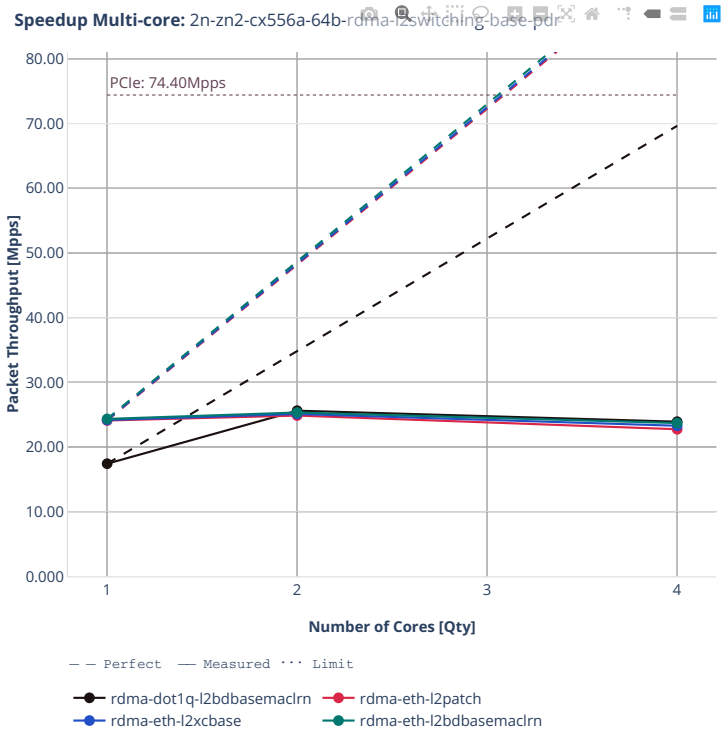



Speedup Multi-core: 2n-zn2-x710-64b-l2switching-base-[avf,dpdk]-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2bdbasemaclrn
eth-l2bdbasemaclrn

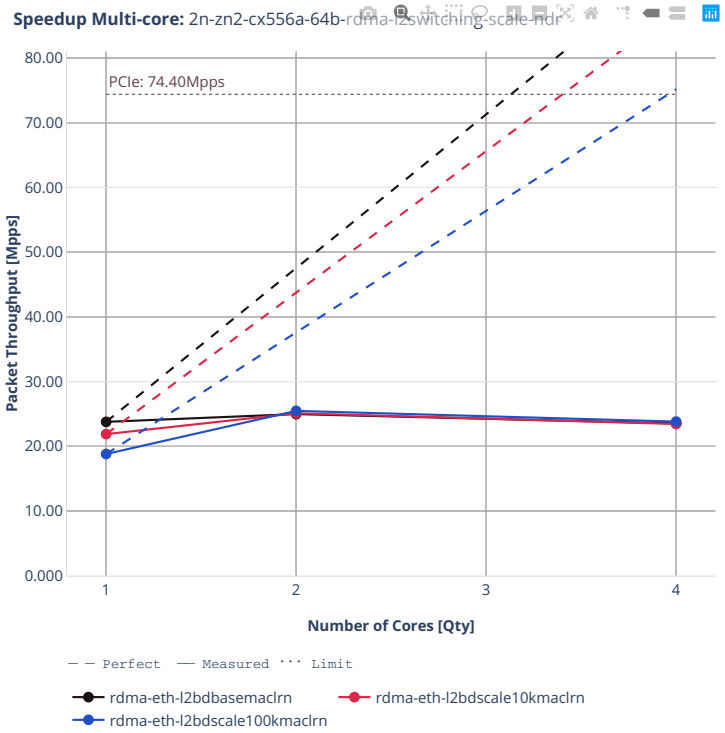
2n-zn2-cx556a

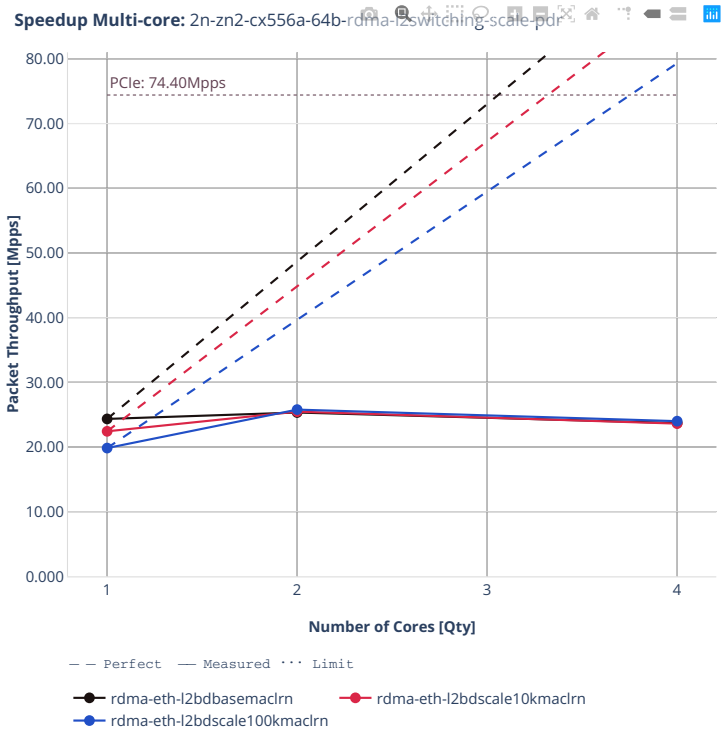
64b-l2switching-base-rdma-core




Speedup Multi-core: 2n-zn2-cx556a-64b-rdma-l2switching-base-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-dot1q-l2bdbasemaclrn
rdma-eth-l2patch
rdma-eth-l2xcbase
rdma-eth-l2bdbasemaclrn

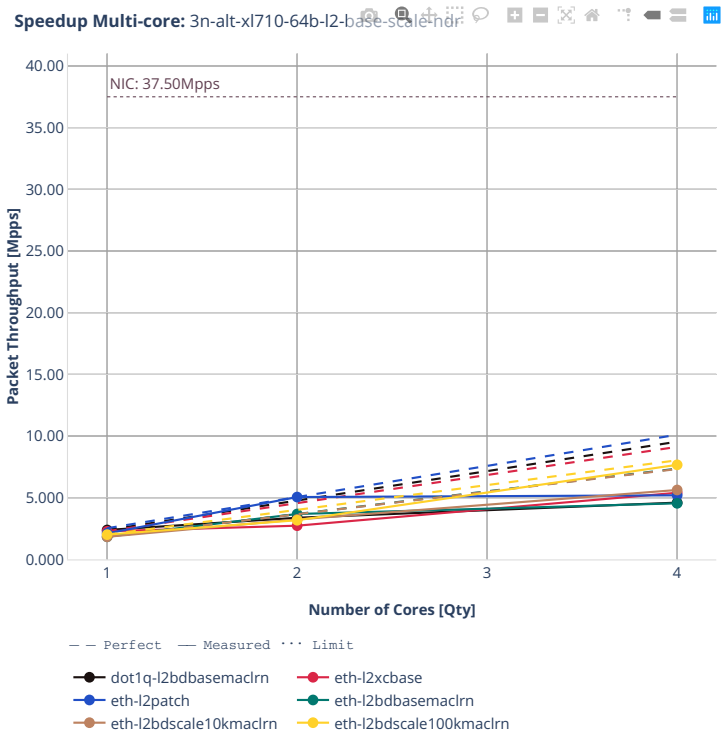
64b-l2switching-scale

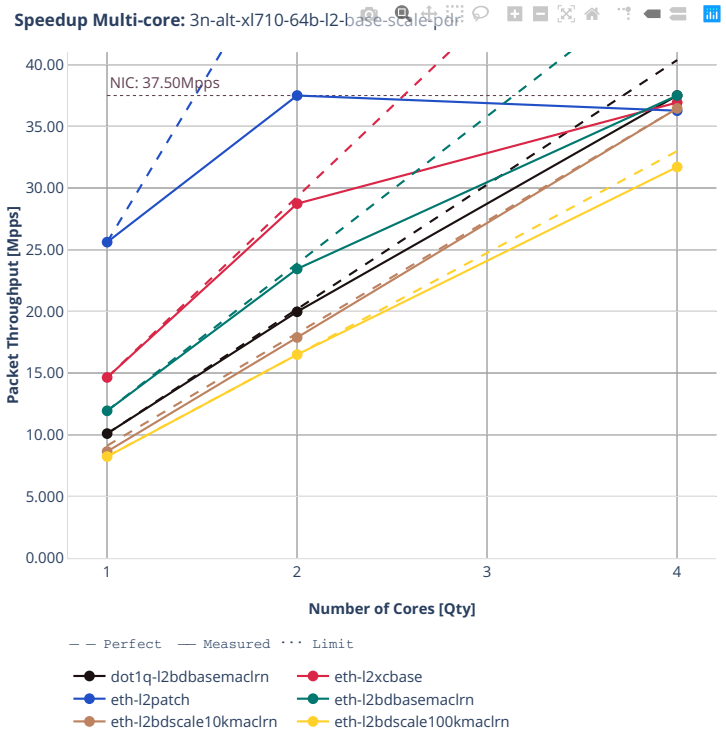



Speedup Multi-core: 2n-zn2-cx556a-64b-rdma-l2switching-scale-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-eth-l2bdbasemaclrn
rdma-eth-l2bdscale10kmaclrn
rdma-eth-l2bdscale100kmaclrn

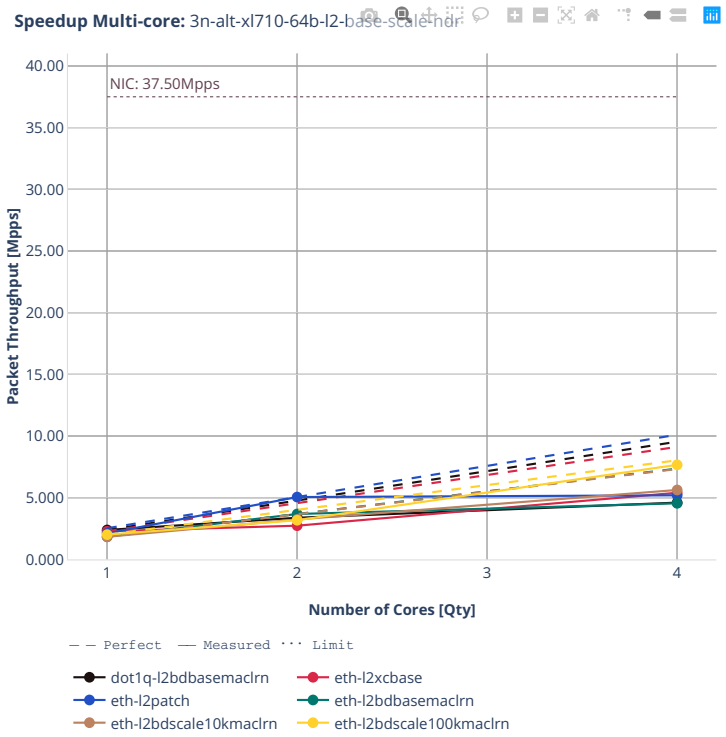
3n-alt-xl710

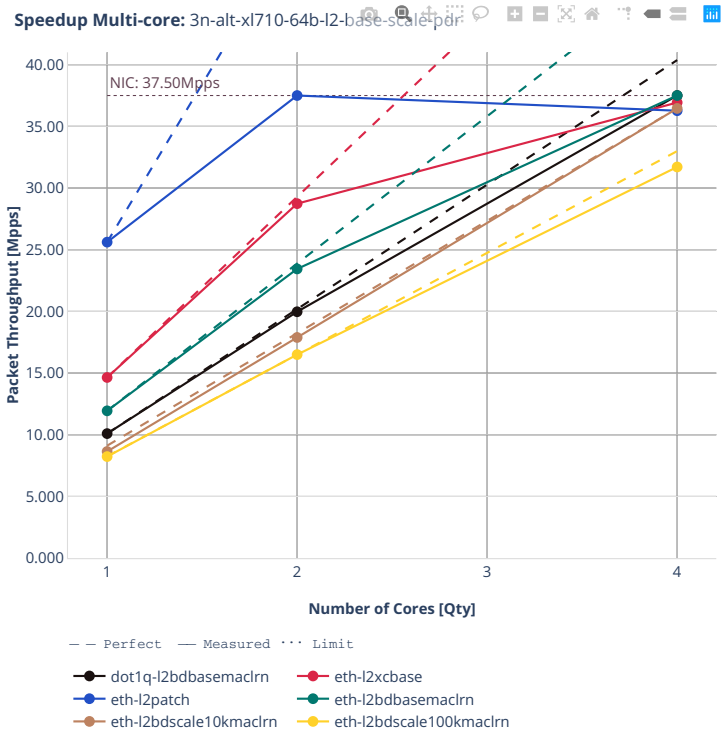
64b-l2switching-base-scale




Speedup Multi-core: 3n-alt-xl710-64b-l2-base-scale-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
dot1q-l2bdbasemaclrn
eth-l2xcbase
eth-l2patch
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn
eth-l2bdscale100kmaclrn

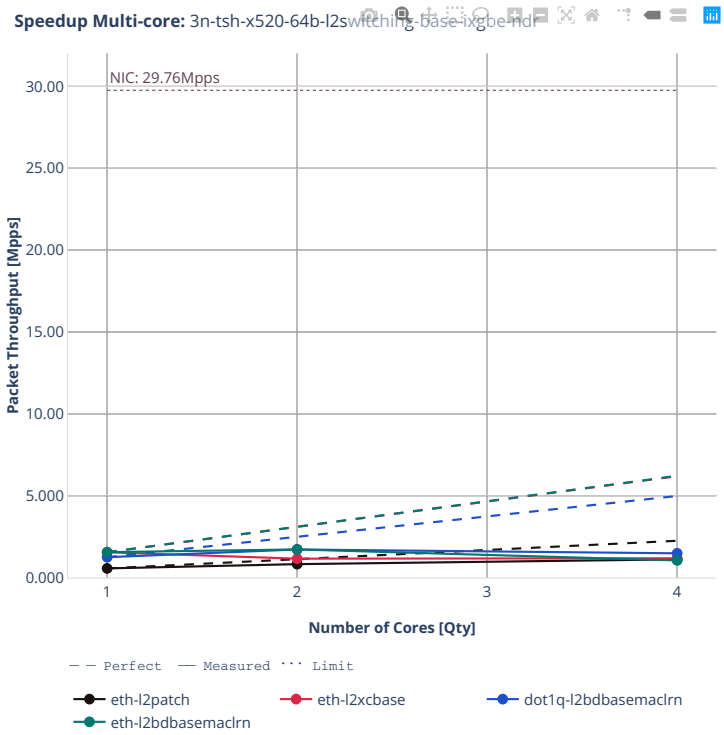
64b-l2switching-features

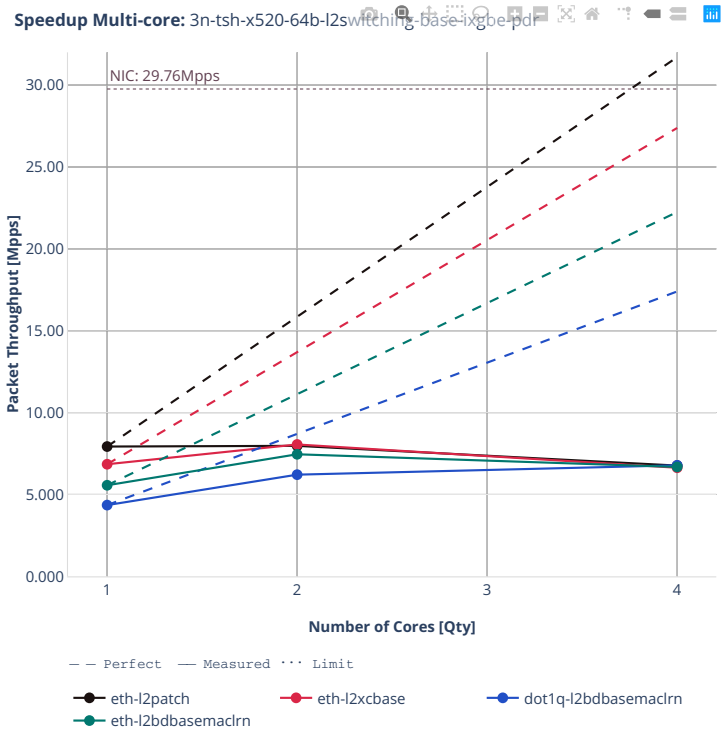



Speedup Multi-core: 3n-alt-xl710-64b-l2-base-scale-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
dot1q-l2bdbasemaclrn
eth-l2xcbase
eth-l2patch
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn
eth-l2bdscale100kmaclrn

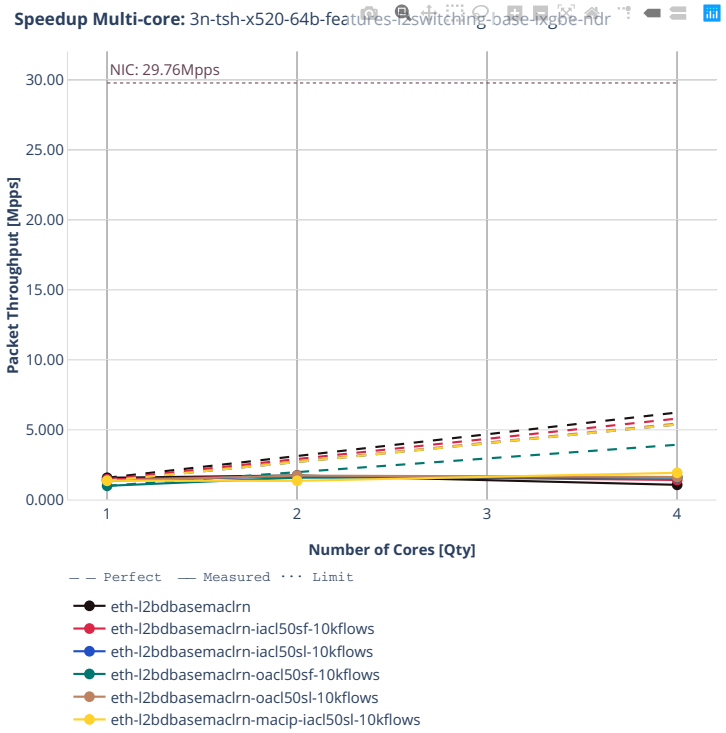
3n-tsh-x520

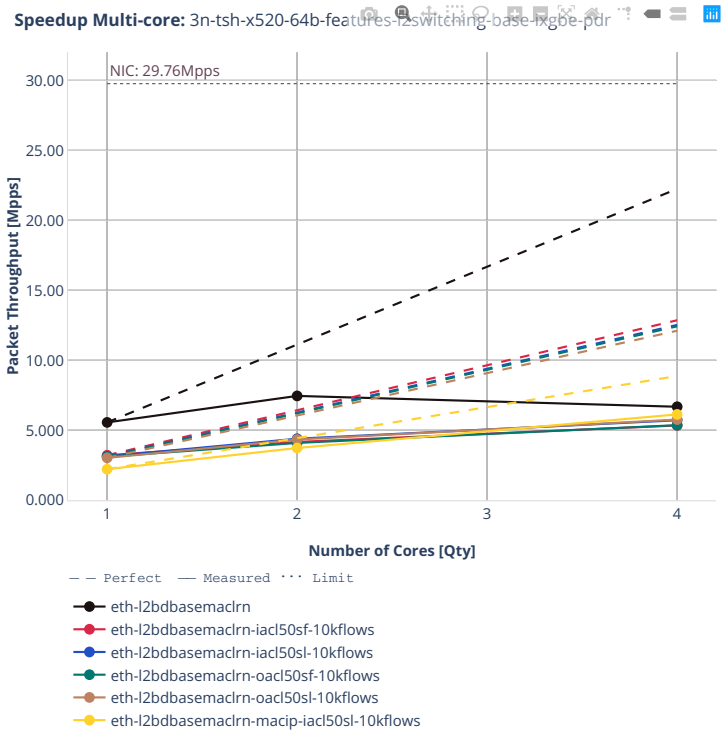
64b-l2switching-base-ixgbe




Speedup Multi-core: 3n-tsh-x520-64b-l2switching-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
dot1q-l2bdbasemaclrn
eth-l2bdbasemaclrn

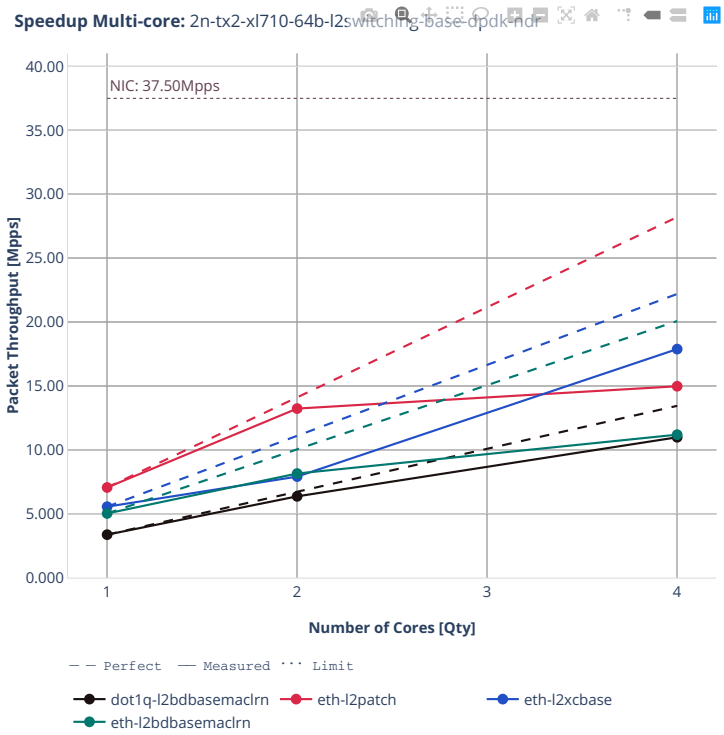
64b-features-l2switching-base-ixgbe




Speedup Multi-core: 3n-tsh-x520-64b-features-l2switching-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2bdbasemaclrn
eth-l2bdbasemaclrn-iacl50sf-10kflows
eth-l2bdbasemaclrn-iacl50sl-10kflows
eth-l2bdbasemaclrn-oacl50sf-10kflows
eth-l2bdbasemaclrn-oacl50sl-10kflows
eth-l2bdbasemaclrn-macip-iacl50sl-10kflows

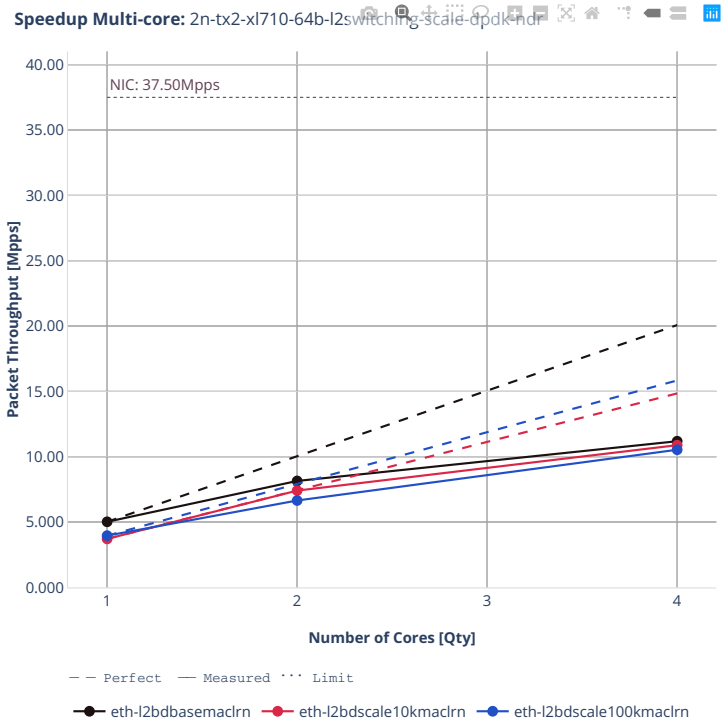
2n-tx2-xl710

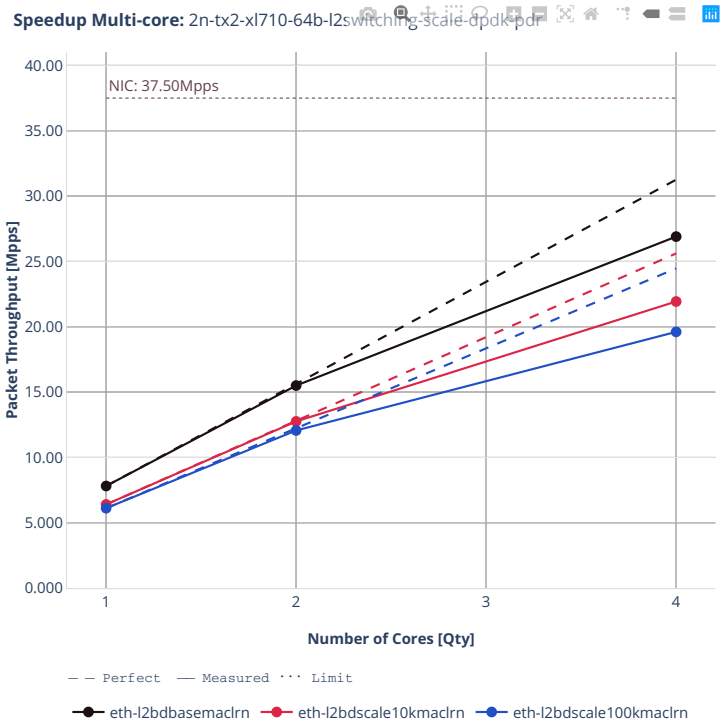
64b-l2switching-base-dpdk



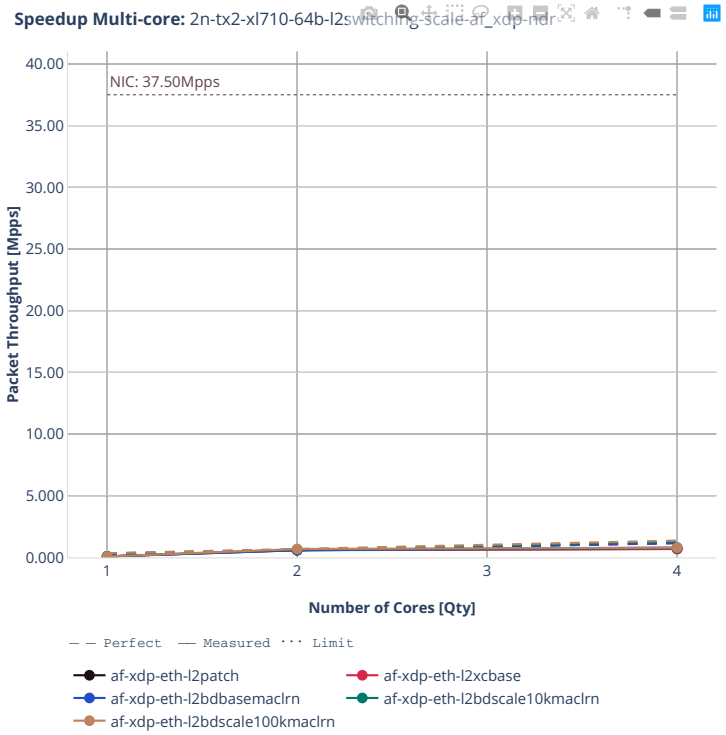

Speedup Multi-core: 2n-tx2-xl710-64b-l2switching-base-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
dot1q-l2bdbasemaclrn
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn

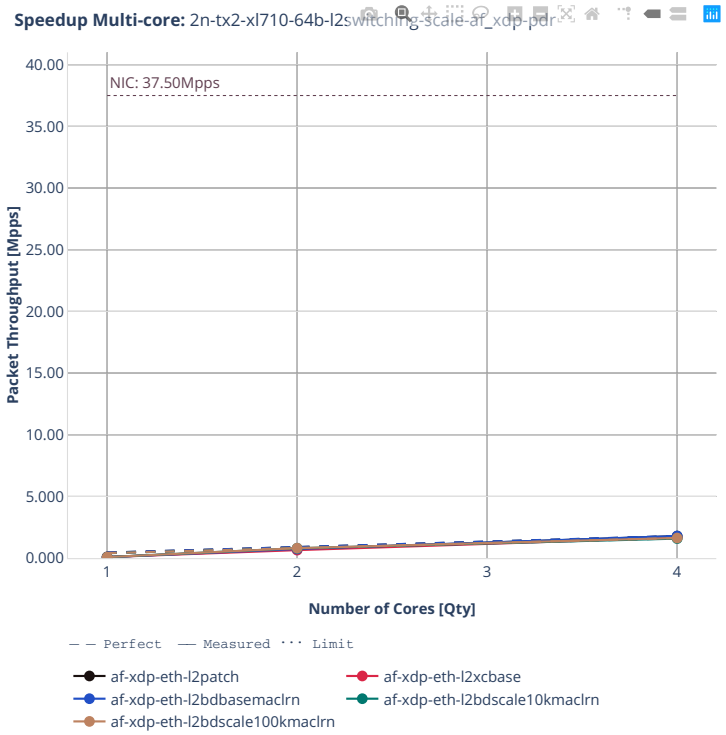
64b-l2switching-scale-dpdk



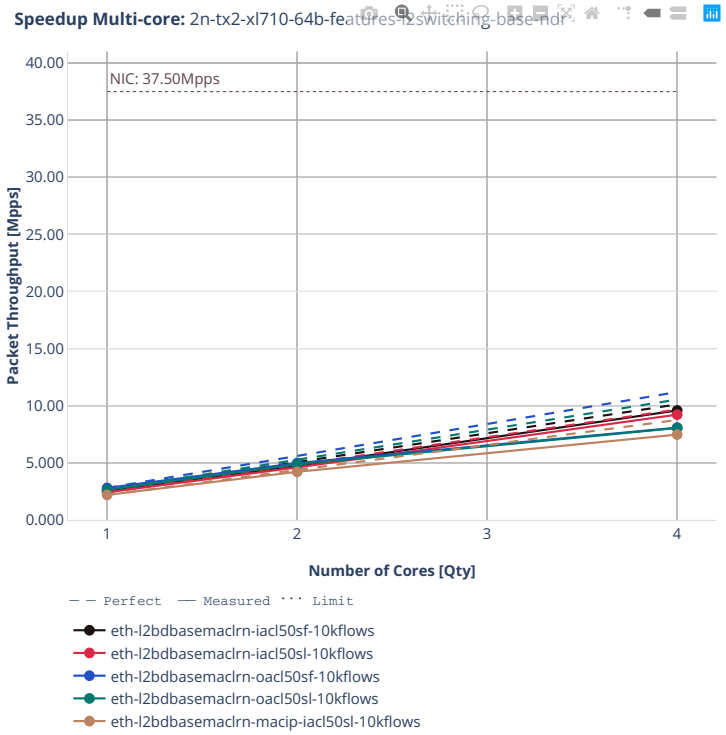

Speedup Multi-core: 2n-tx2-xl710-64b-l2switching-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn
eth-l2bdscale100kmaclrn

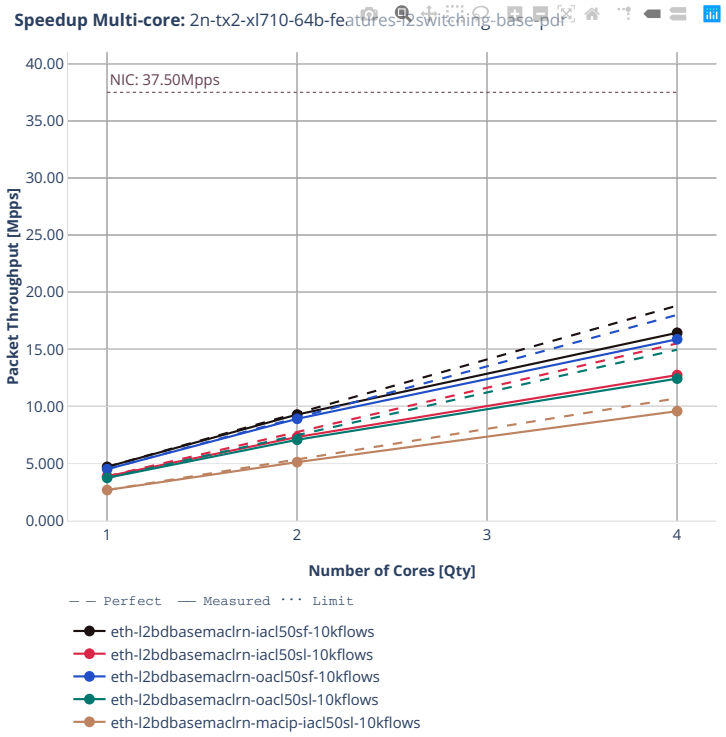
64b-l2switching-scale-af-xdp




Speedup Multi-core: 2n-tx2-xl710-64b-l2switching-scale-af_xdp-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
af-xdp-eth-l2patch
af-xdp-eth-l2xcbase
af-xdp-eth-l2bdbasemaclrn
af-xdp-eth-l2bdscale10kmaclrn
af-xdp-eth-l2bdscale100kmaclrn

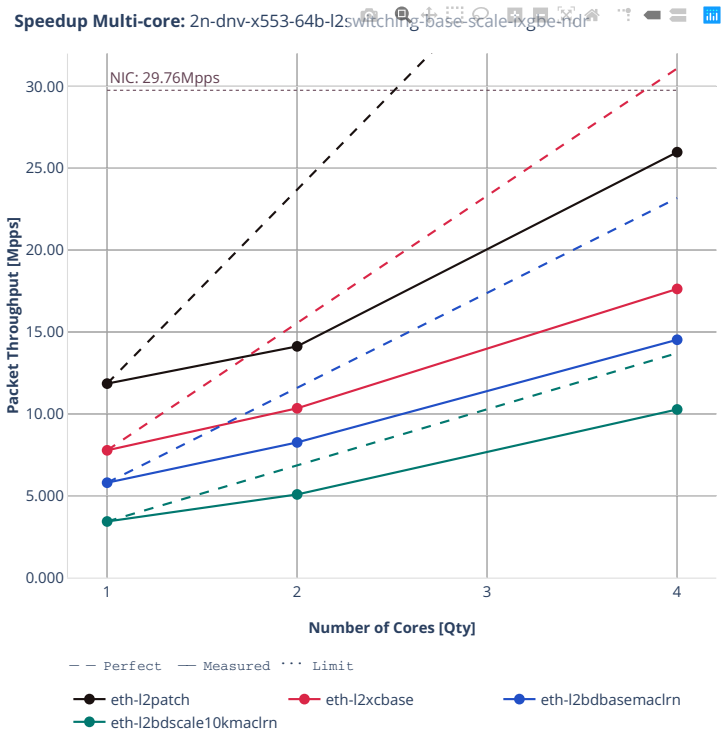
64b-features-l2switching-base-dpdk

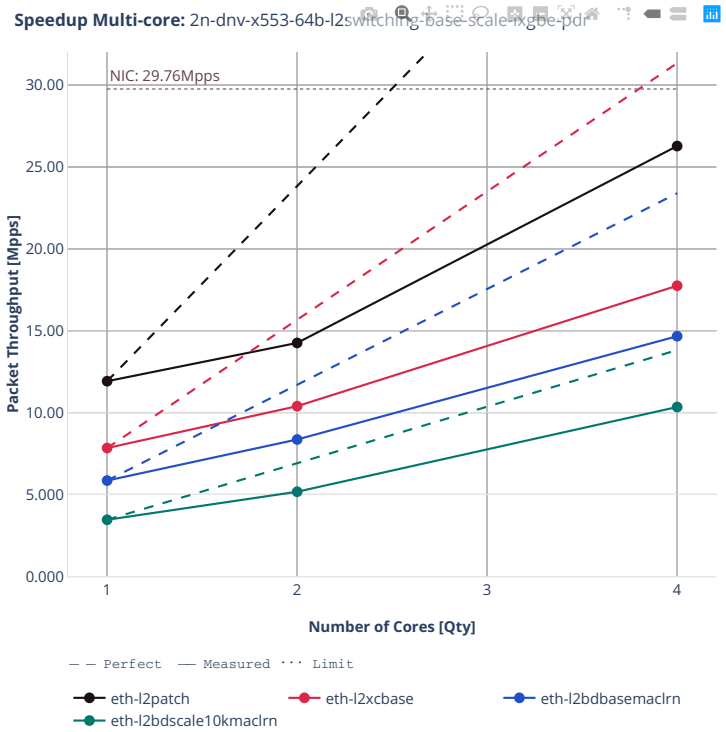



Speedup Multi-core: 2n-tx2-xl710-64b-features-l2switching-base-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2bdbasemaclrn-iacl50sf-10kflows
eth-l2bdbasemaclrn-iacl50sl-10kflows
eth-l2bdbasemaclrn-oacl50sf-10kflows
eth-l2bdbasemaclrn-oacl50sl-10kflows
eth-l2bdbasemaclrn-macip-iacl50sl-10kflows

2n-dnv-x553

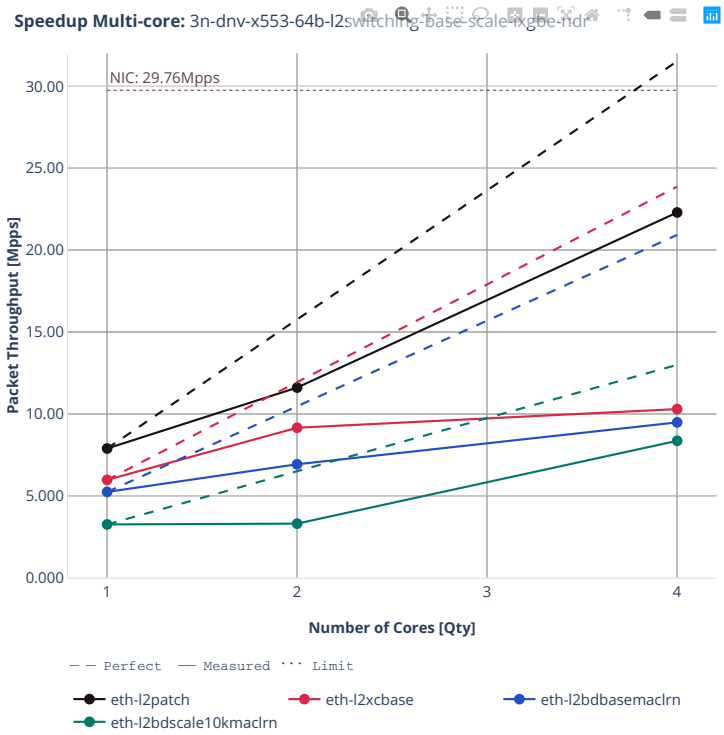
64b-l2switching-base-scale-ixgbe

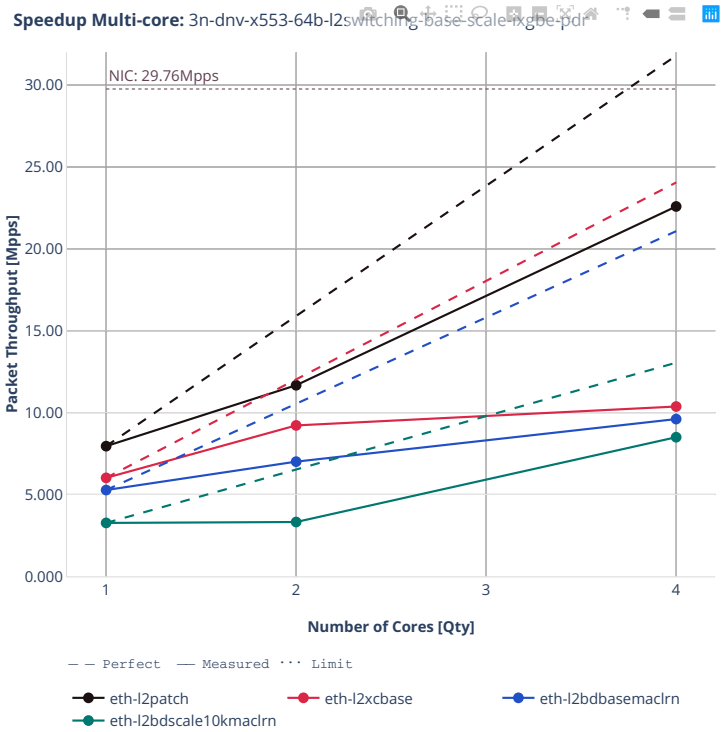



Speedup Multi-core: 2n-dnv-x553-64b-l2switching-base-scale-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn

3n-dnv-x553

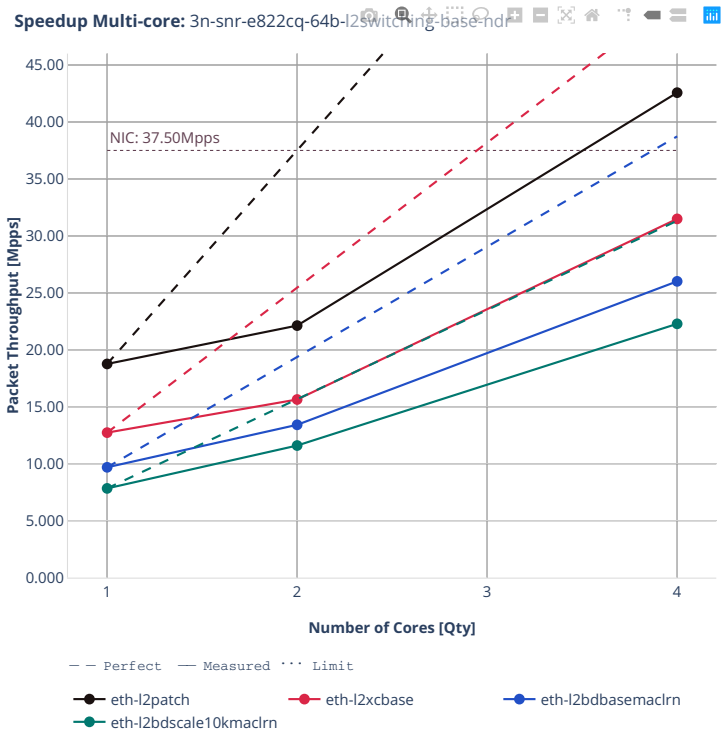
64b-l2switching-base-scale-ixgbe

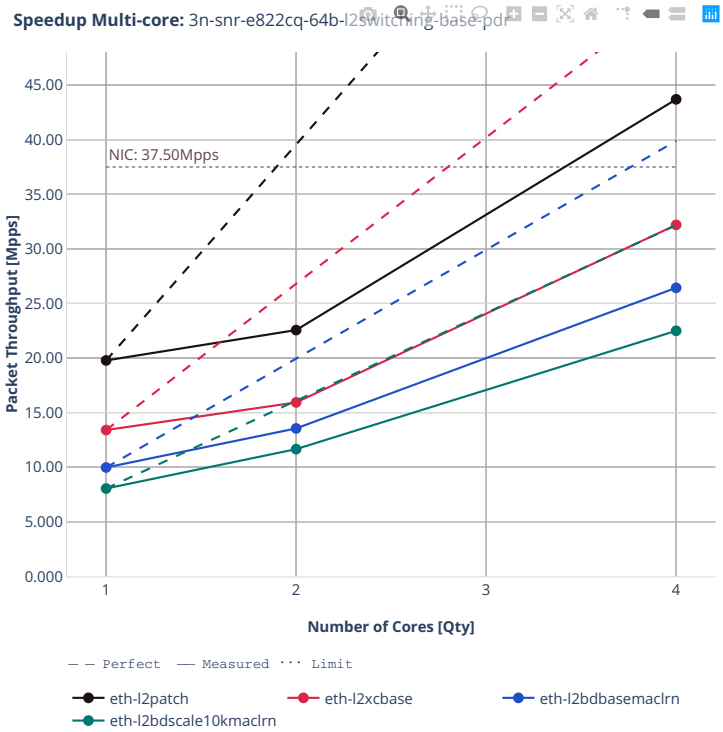



Speedup Multi-core: 3n-dnv-x553-64b-l2switching-base-scale-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn

3n-snr-e822cq

64b-l2switching-base




Speedup Multi-core: 3n-snr-e822cq-64b-l2switching-base-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2patch
eth-l2xcbase
eth-l2bdbasemaclrn
eth-l2bdscale10kmaclrn

2.4.2 IPv4 Routing

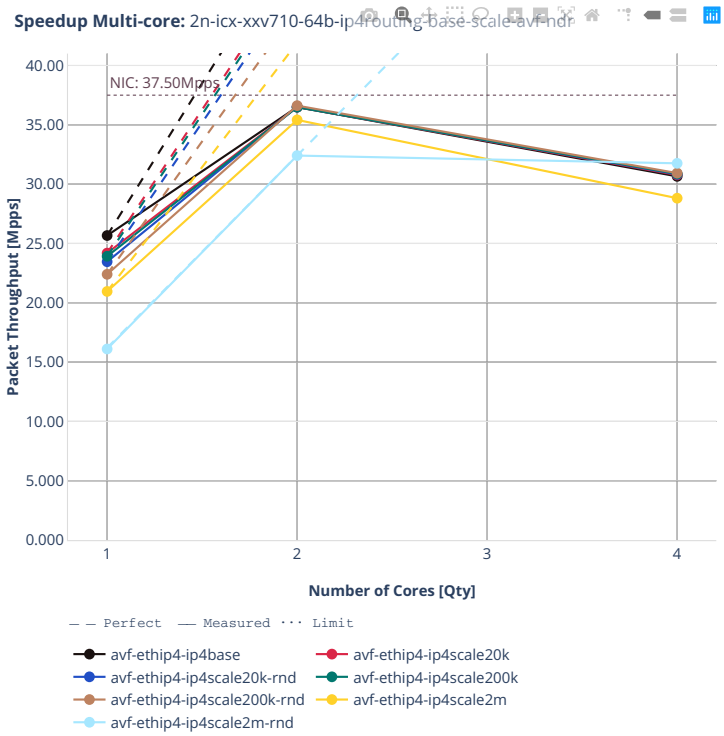
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 64B performance tests with VPP IPv4 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

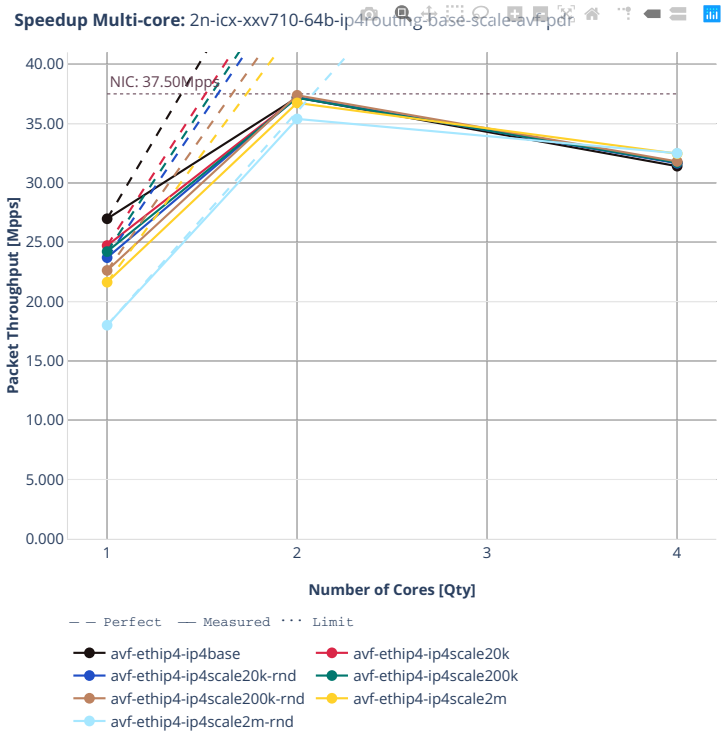
CSIT source code for the test cases used for plots can be found in CSIT git repository[134].

[134] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls2210

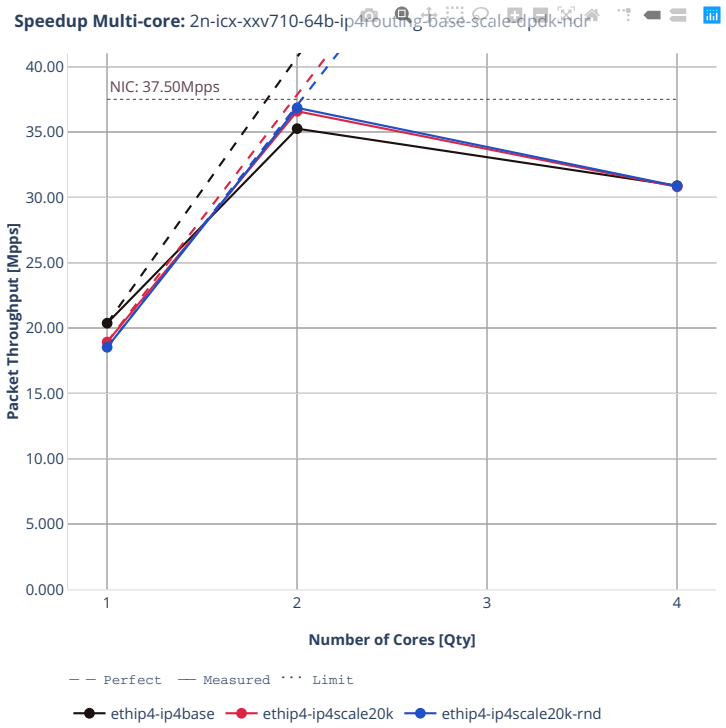
2n-icx-xxv710

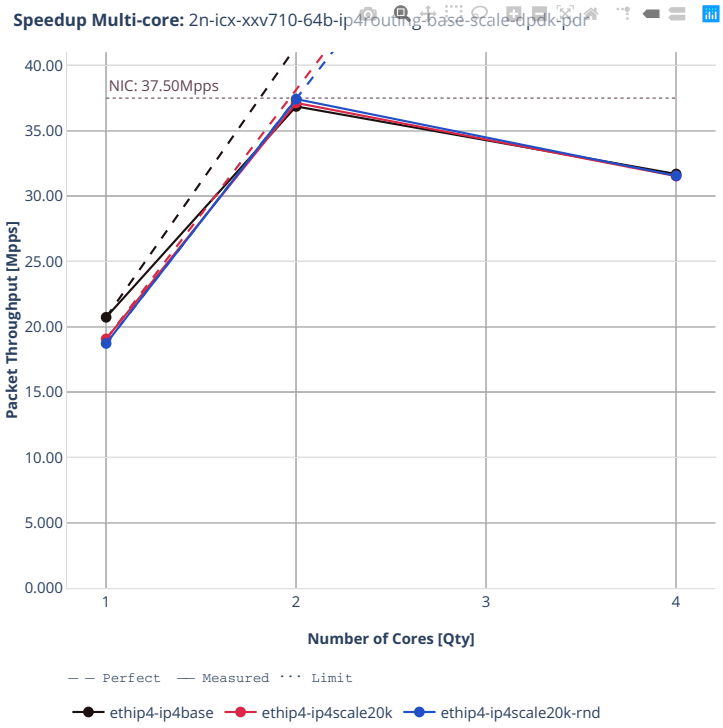
64b-ip4routing-base-scale-avf



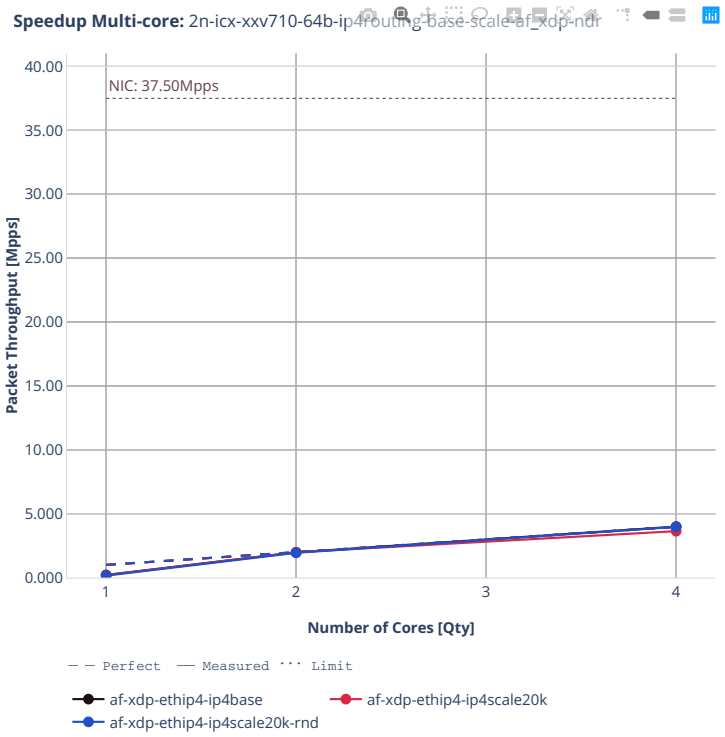

Speedup Multi-core: 2n-icx-xxv710-64b-ip4routing-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4-ip4base
avf-ethip4-ip4scale20k
avf-ethip4-ip4scale20k-rnd
avf-ethip4-ip4scale200k
avf-ethip4-ip4scale200k-rnd
avf-ethip4-ip4scale2m
avf-ethip4-ip4scale2m-rnd

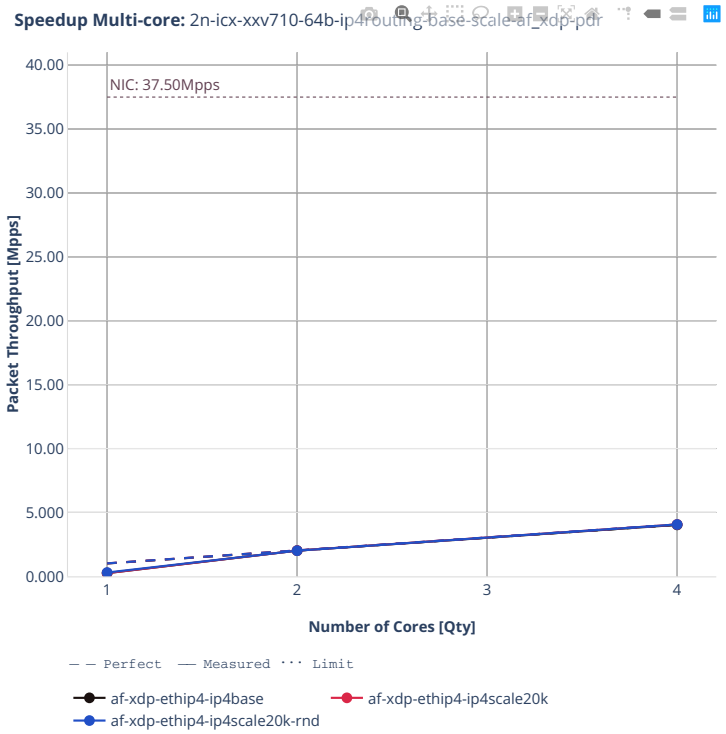
64b-ip4routing-base-scale-dpdk



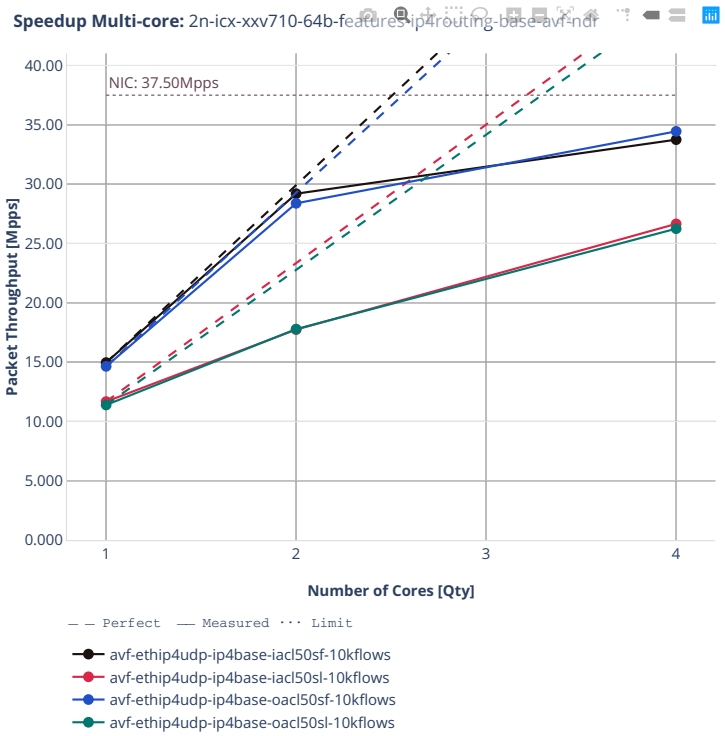

Speedup Multi-core: 2n-icx-xxv710-64b-ip4routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale20k-rnd

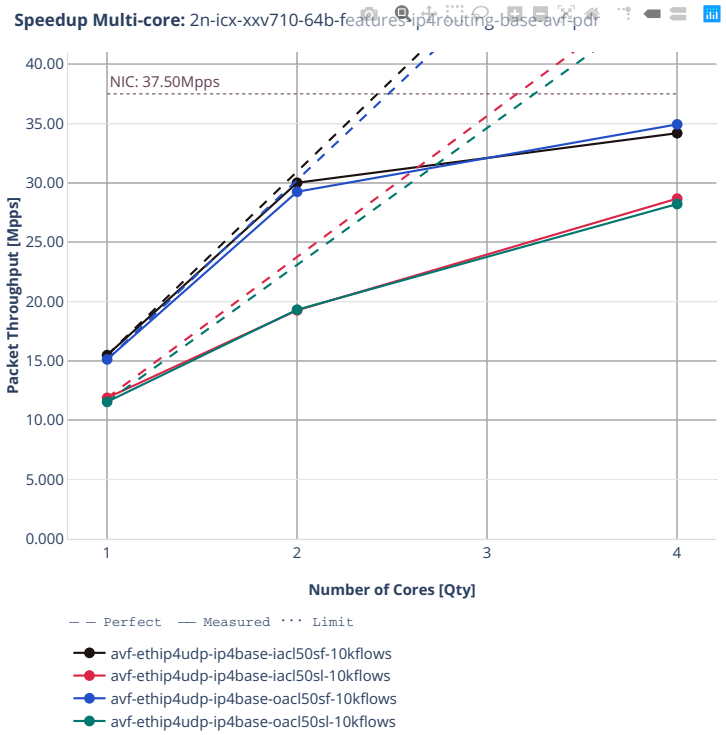
64b-ip4routing-base-scale-af_xdp




Speedup Multi-core: 2n-icx-xxv710-64b-ip4routing-base-scale-af_xdp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
af-xdp-ethip4-ip4base
af-xdp-ethip4-ip4scale20k
af-xdp-ethip4-ip4scale20k-rnd

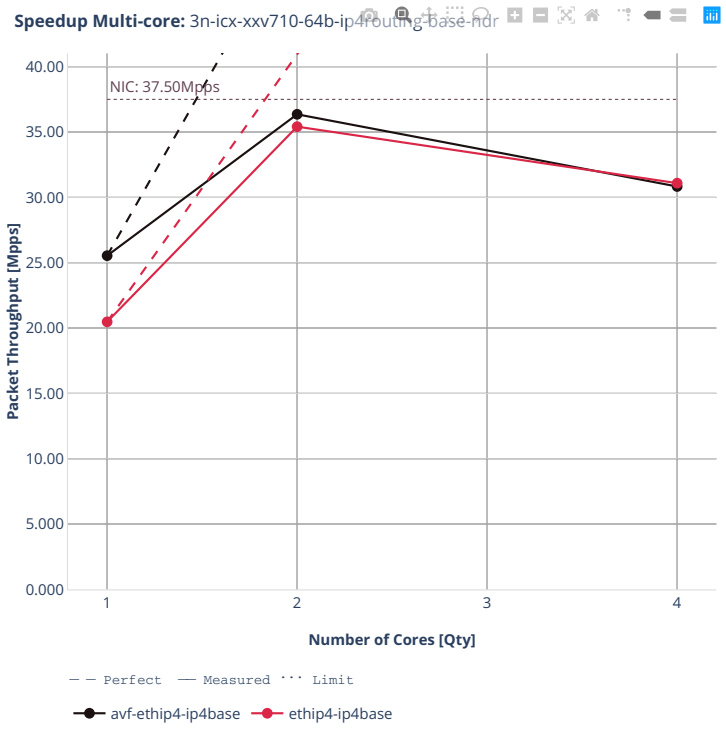
64b-features-ip4routing-base-avf

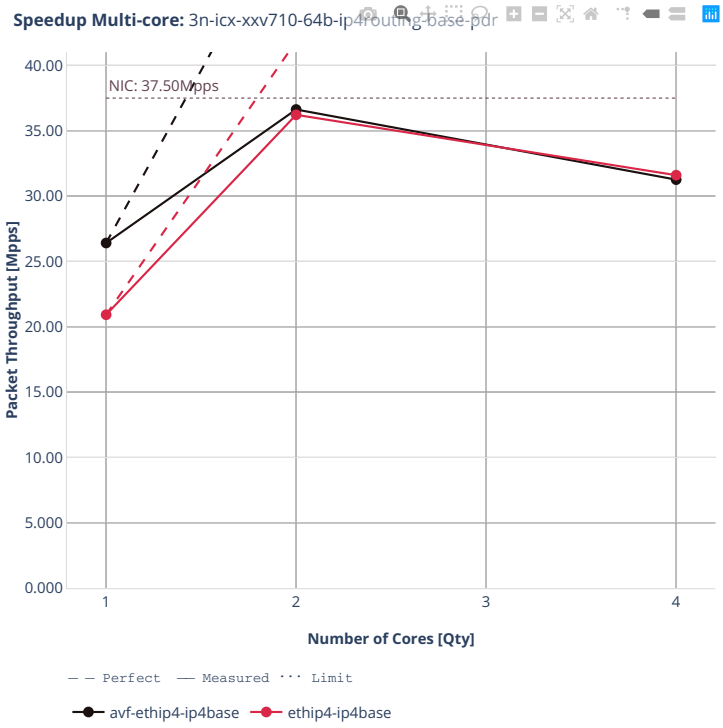



Speedup Multi-core: 2n-icx-xxv710-64b-features-ip4routing-base-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-ip4base-iacl50sf-10kflows
avf-ethip4udp-ip4base-iacl50sl-10kflows
avf-ethip4udp-ip4base-oacl50sf-10kflows
avf-ethip4udp-ip4base-oacl50sl-10kflows

3n-icx-xxv710

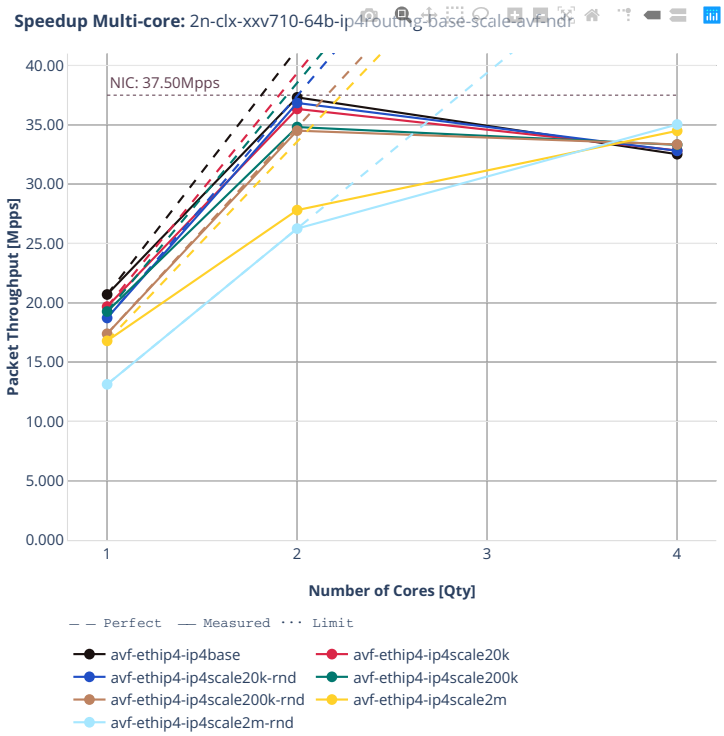
64b-ip4routing-base

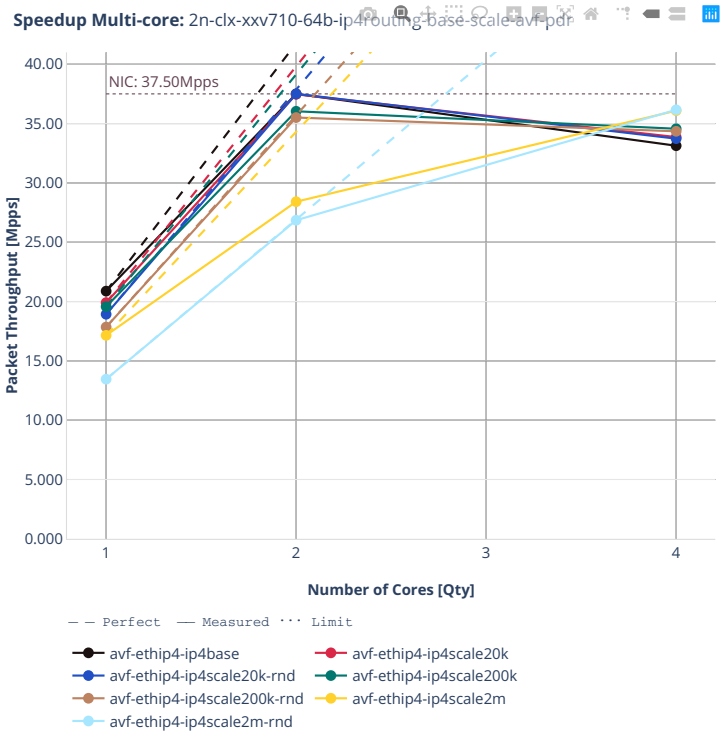



Speedup Multi-core: 3n-icx-xxv710-64b-ip4routing-base-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4-ip4base
ethip4-ip4base

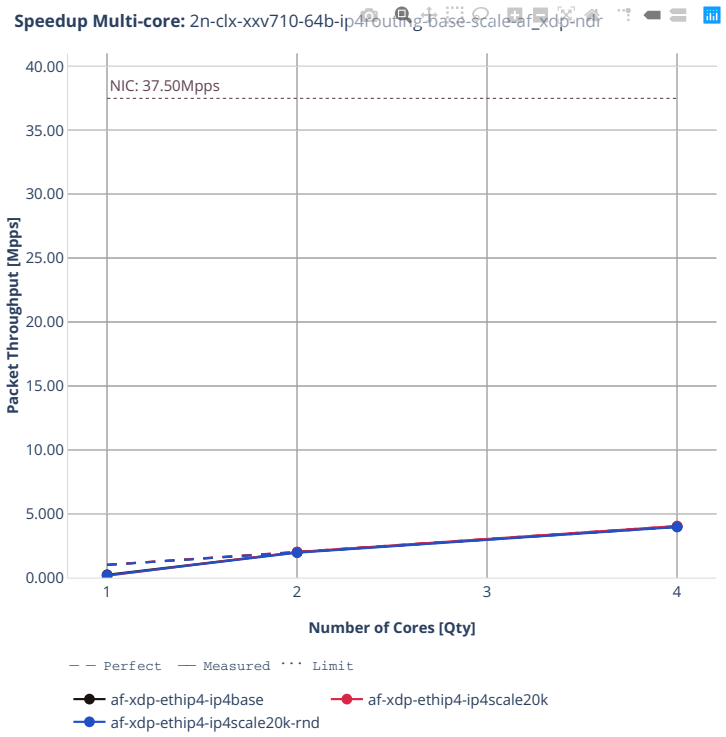
2n-clx-xxv710

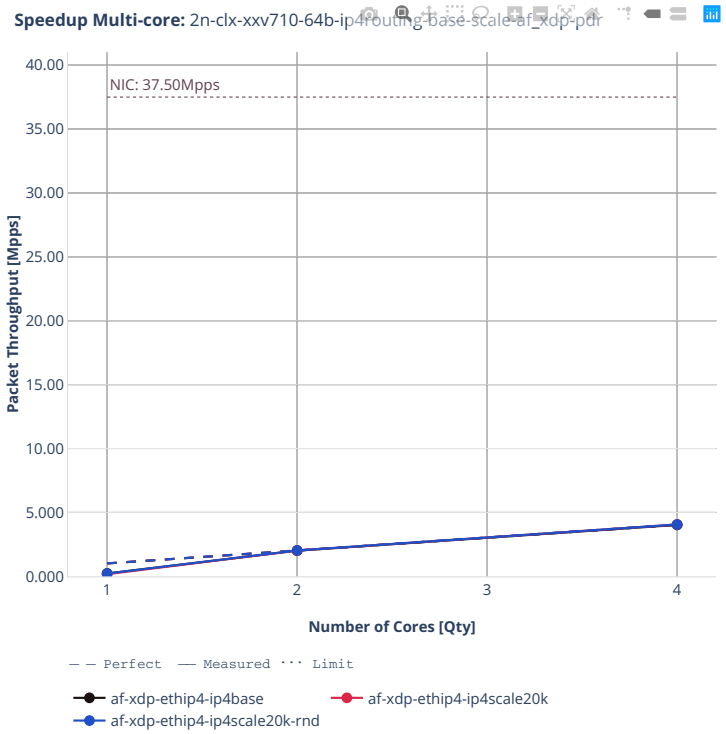
64b-ip4routing-base-scale-avf



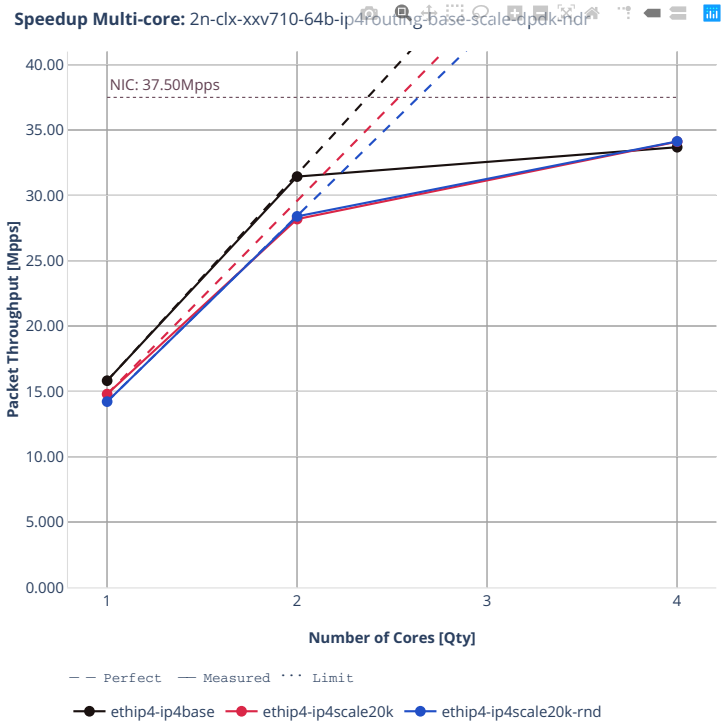

Speedup Multi-core: 2n-clx-xxv710-64b-ip4routing-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4-ip4base
avf-ethip4-ip4scale20k
avf-ethip4-ip4scale20k-rnd
avf-ethip4-ip4scale200k
avf-ethip4-ip4scale200k-rnd
avf-ethip4-ip4scale2m
avf-ethip4-ip4scale2m-rnd

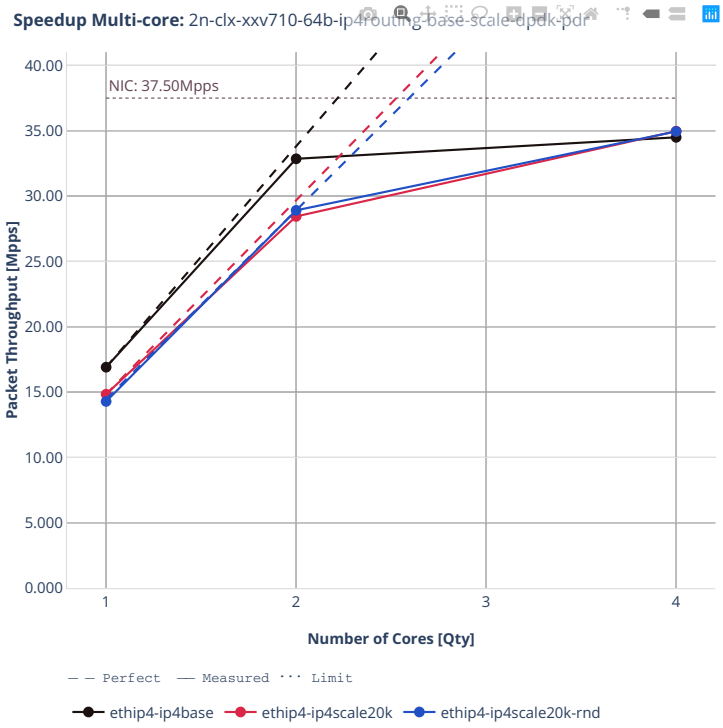
64b-ip4routing-base-scale-af-xdp



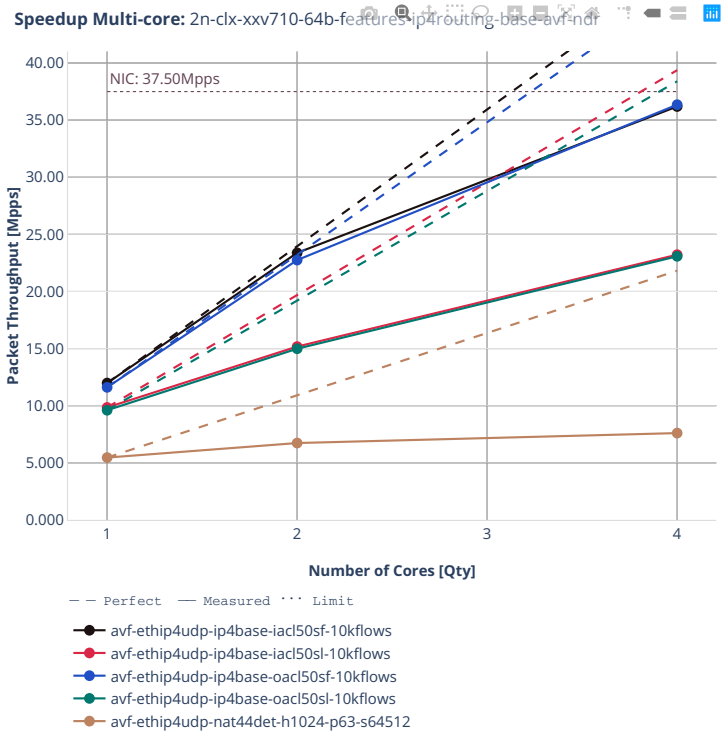

Speedup Multi-core: 2n-clx-xxv710-64b-ip4routing-base-scale-af_xdp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
af-xdp-ethip4-ip4base
af-xdp-ethip4-ip4scale20k
af-xdp-ethip4-ip4scale20k-rnd

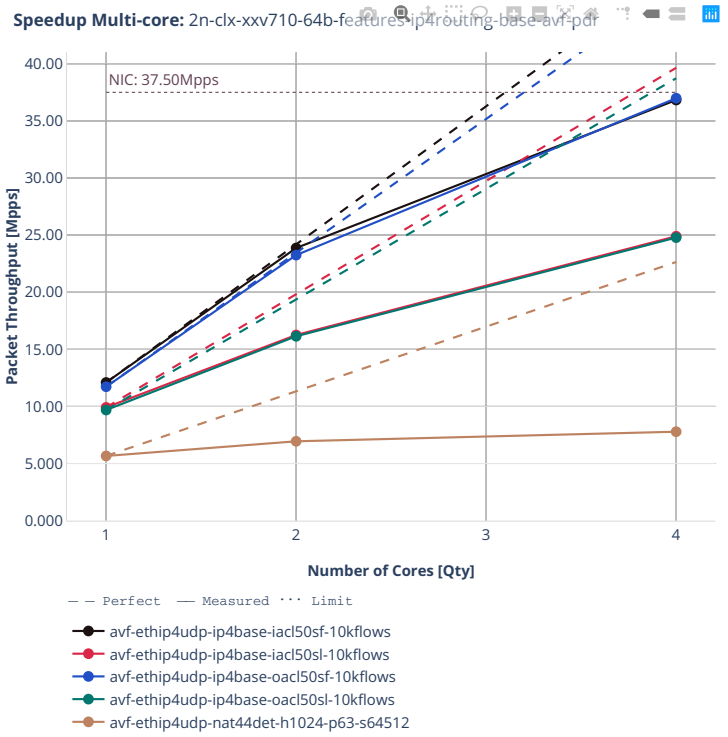
64b-ip4routing-base-scale-dpdk




Speedup Multi-core: 2n-clx-xxv710-64b-ip4routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale20k-rnd

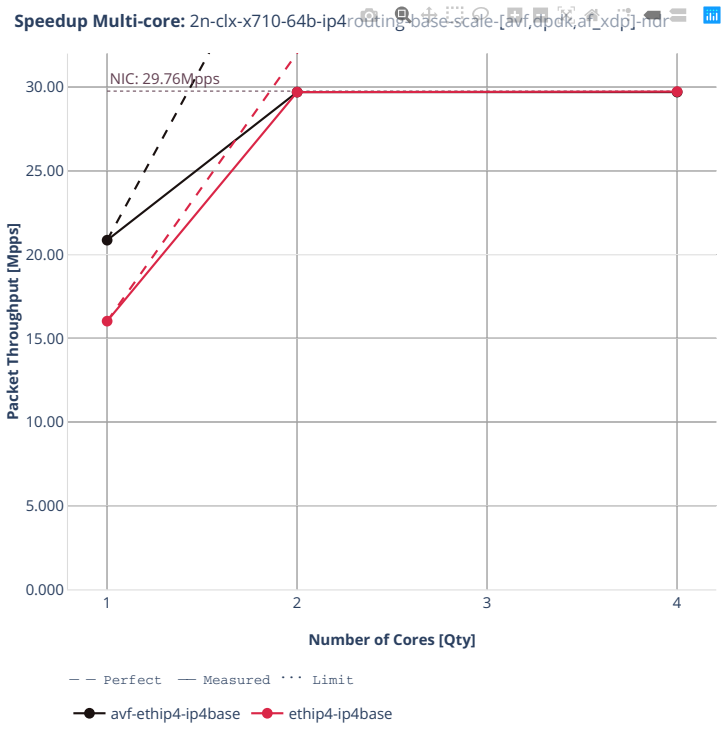
64b-features-ip4routing-base-avf

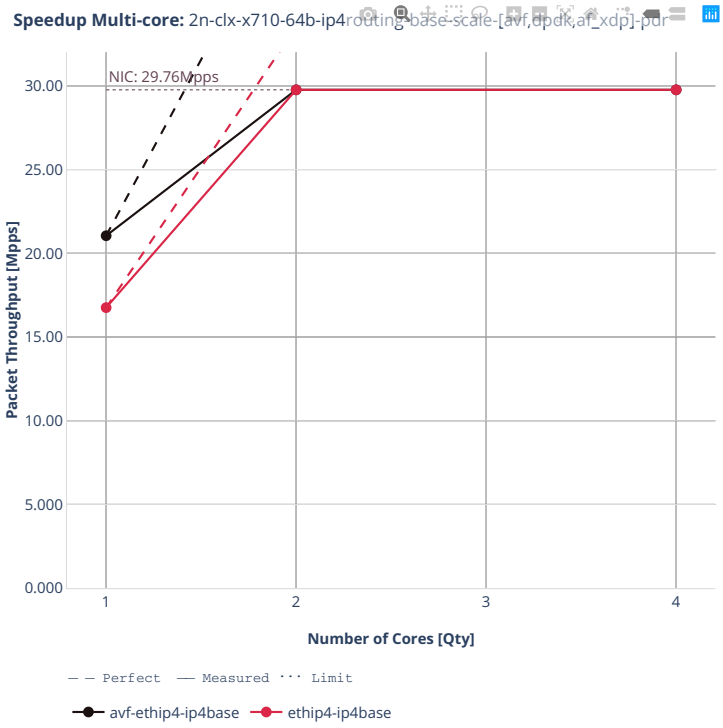



Speedup Multi-core: 2n-clx-xxv710-64b-features-ip4routing-base-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-ip4base-iacl50sf-10kflows
avf-ethip4udp-ip4base-iacl50sl-10kflows
avf-ethip4udp-ip4base-oacl50sf-10kflows
avf-ethip4udp-ip4base-oacl50sl-10kflows
avf-ethip4udp-nat44det-h1024-p63-s64512

2n-clx-x710

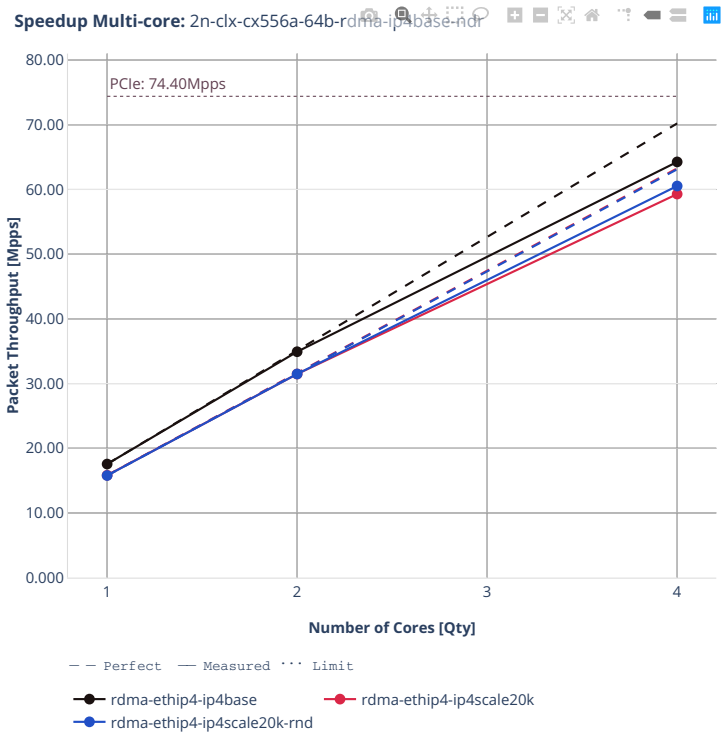
64b-ip4routing-base-scale-[avf,dpdk]

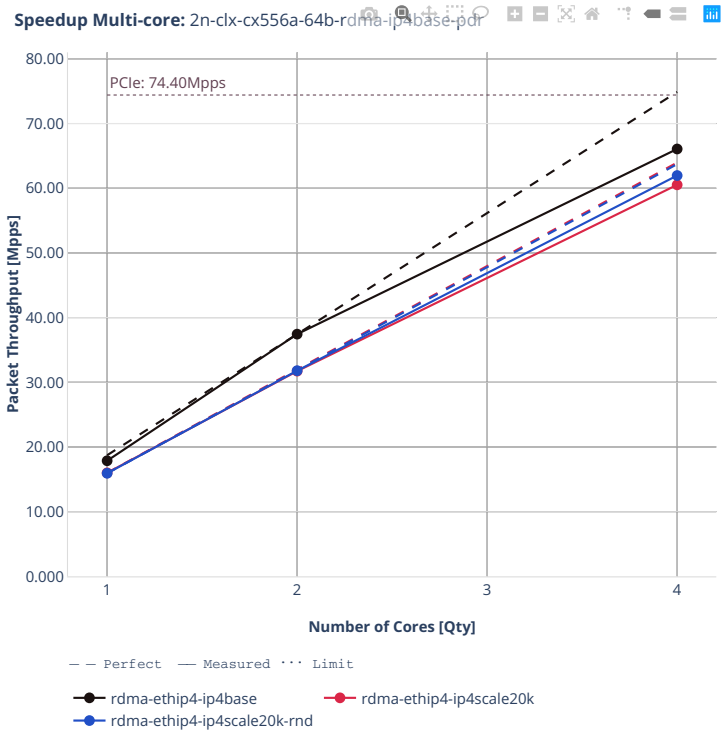



Speedup Multi-core: 2n-clx-x710-64b-ip4routing-base-scale-[avf,dpdk,af_xdp]-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4-ip4base
ethip4-ip4base

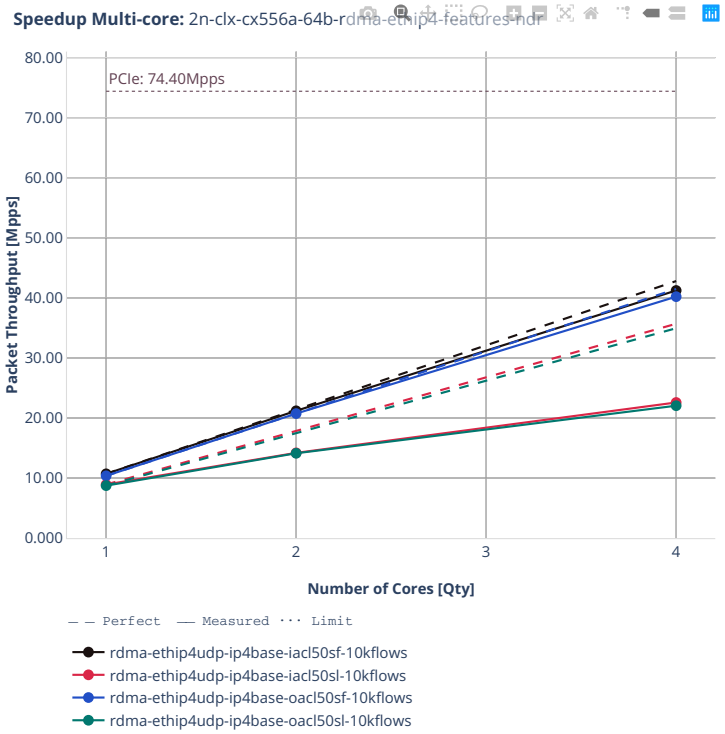
2n-clx-cx556a

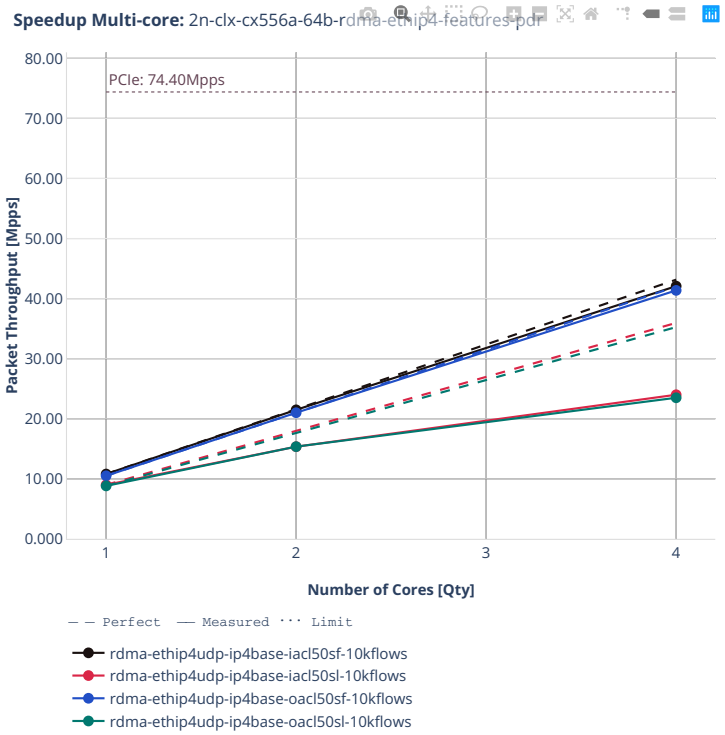
64b-ip4routing-base-scale-rdma-core




Speedup Multi-core: 2n-clx-cx556a-64b-rdma-ip4base-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-ethip4-ip4base
rdma-ethip4-ip4scale20k
rdma-ethip4-ip4scale20k-rnd

64b-ip4routing-features



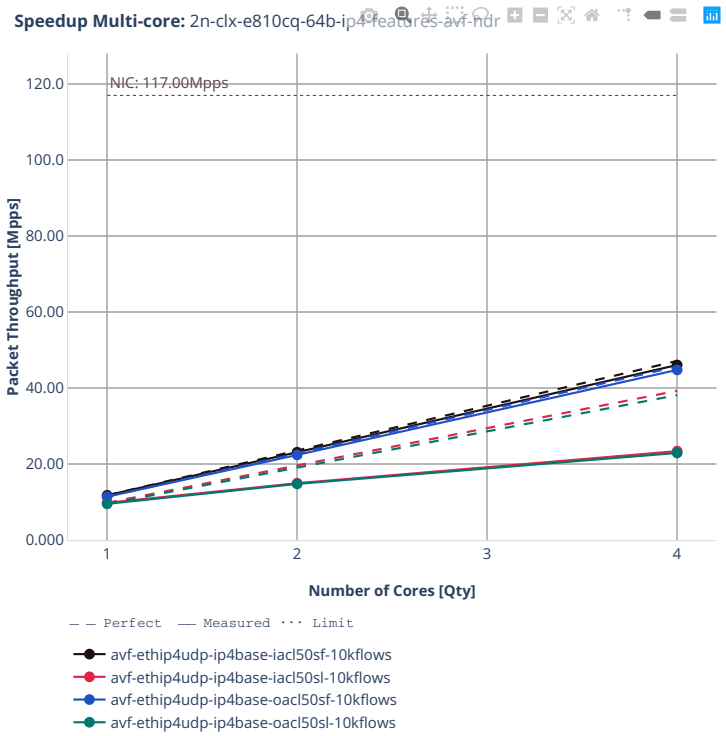

Speedup Multi-core: 2n-clx-cx556a-64b-rdma-ethip4-features-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-ethip4udp-ip4base-iacl50sf-10kflows
rdma-ethip4udp-ip4base-iacl50sl-10kflows
rdma-ethip4udp-ip4base-oacl50sf-10kflows
rdma-ethip4udp-ip4base-oacl50sl-10kflows

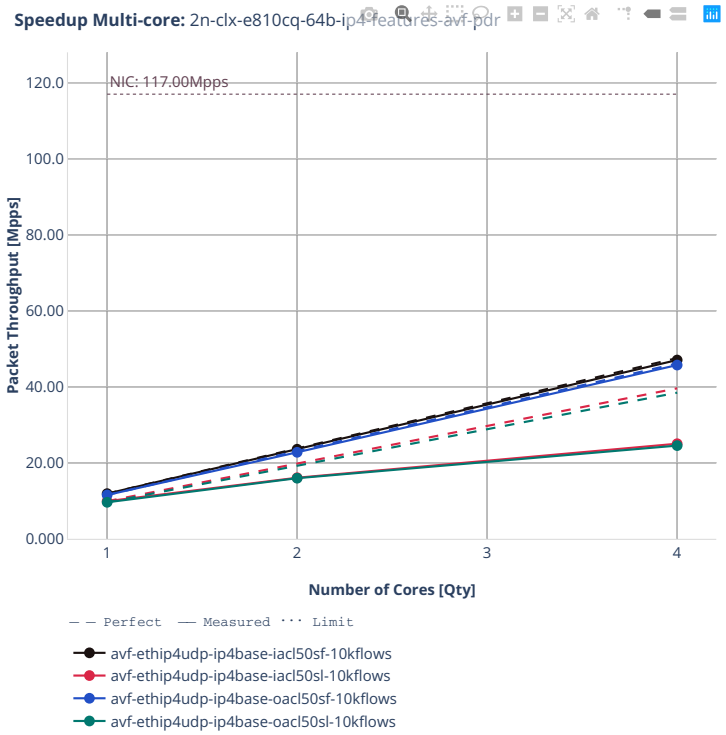
2n-clx-e810cq

64b-ip4routing-base-scale-avf

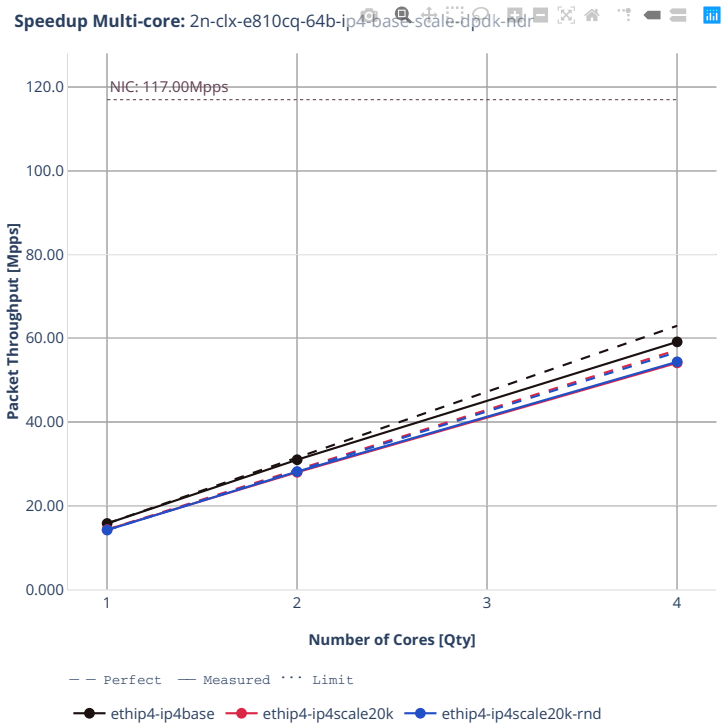

Speedup Multi-core: 2n-clx-e810cq-64b-ip4-base-scale-avf-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4-ip4base
avf-ethip4-ip4scale20k
avf-ethip4-ip4scale20k-rnd

64b-ip4routing-features-avf




Speedup Multi-core: 2n-clx-e810cq-64b-ip4-features-avf-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-ip4base-iacl50sf-10kflows
avf-ethip4udp-ip4base-iacl50sl-10kflows
avf-ethip4udp-ip4base-oacl50sf-10kflows
avf-ethip4udp-ip4base-oacl50sl-10kflows

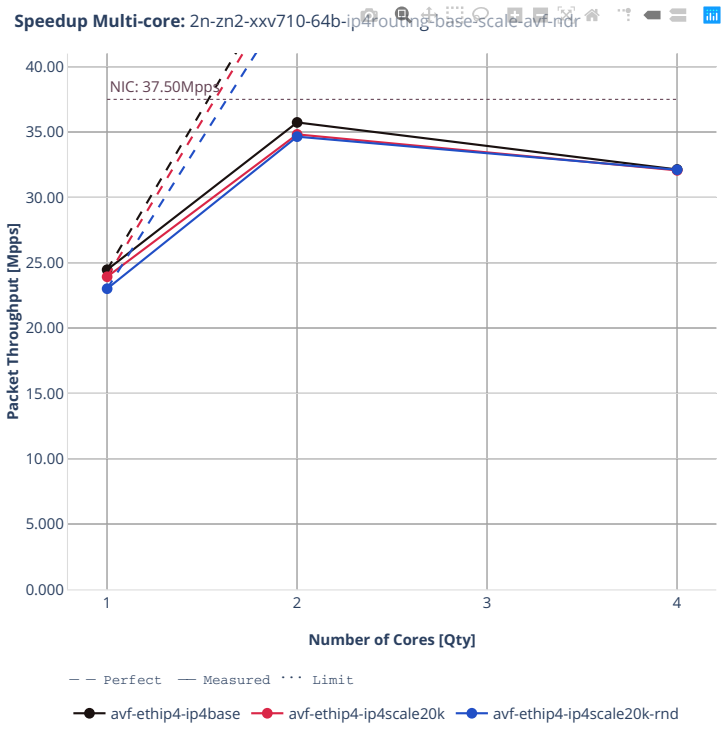
64b-ip4routing-base-scale-dpdk

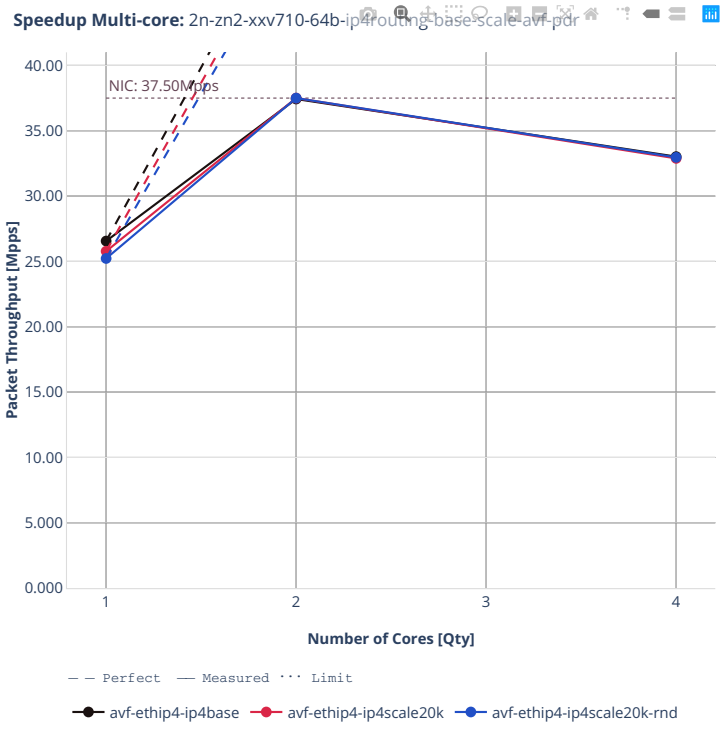



Speedup Multi-core: 2n-clx-e810cq-64b-ip4-base-scale-dpdk-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale20k-rnd

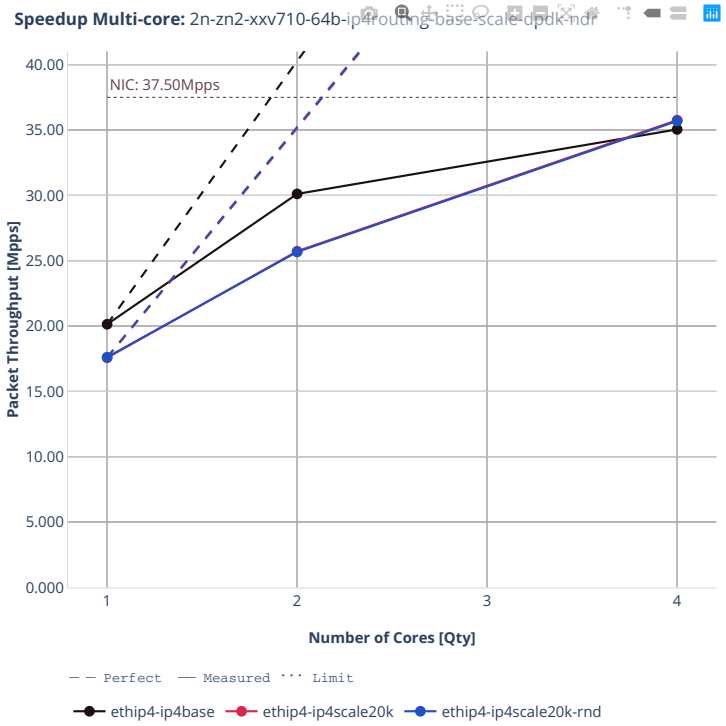
2n-zn2-xxv710

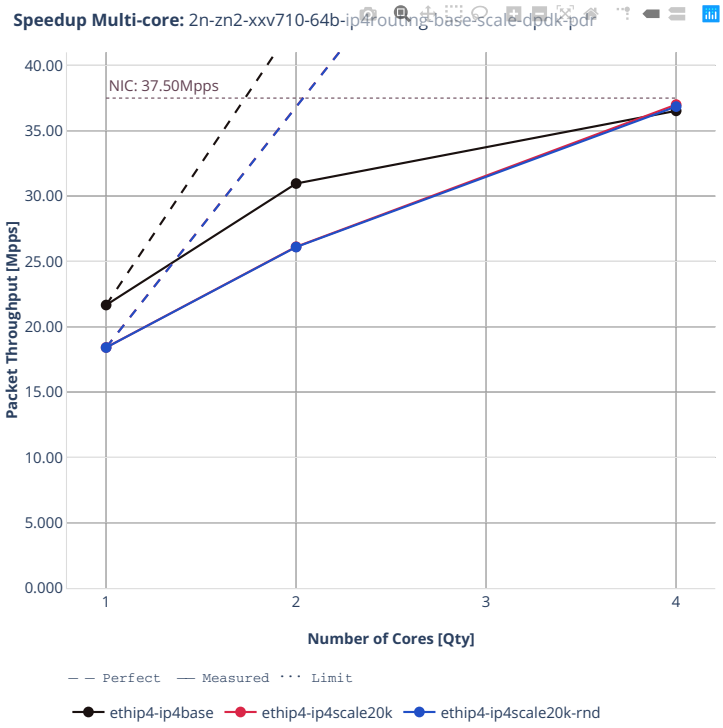
64b-ip4routing-base-scale-avf



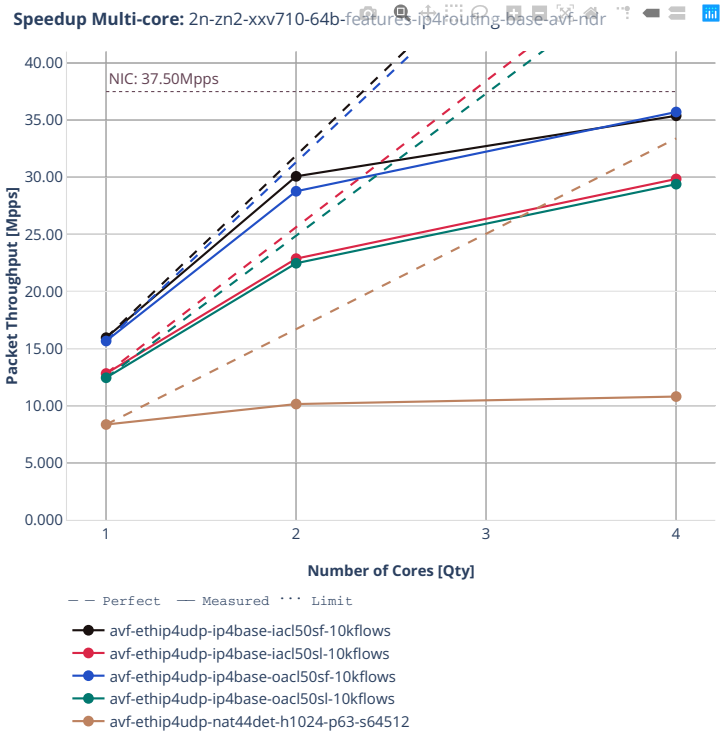

Speedup Multi-core: 2n-zn2-xxv710-64b-ip4routing-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4-ip4base
avf-ethip4-ip4scale20k
avf-ethip4-ip4scale20k-rnd

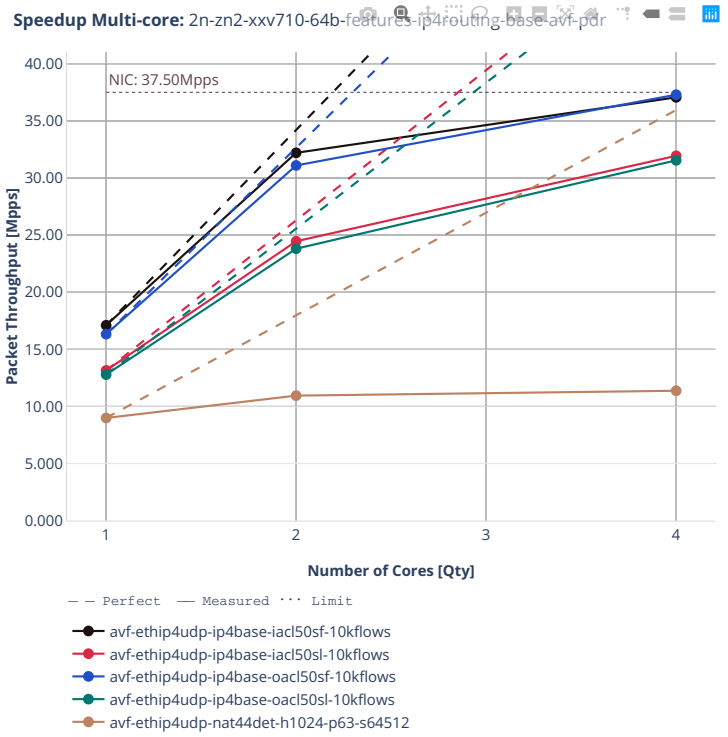
64b-ip4routing-base-scale-dpdk




Speedup Multi-core: 2n-zn2-xxv710-64b-ip4routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale20k-rnd

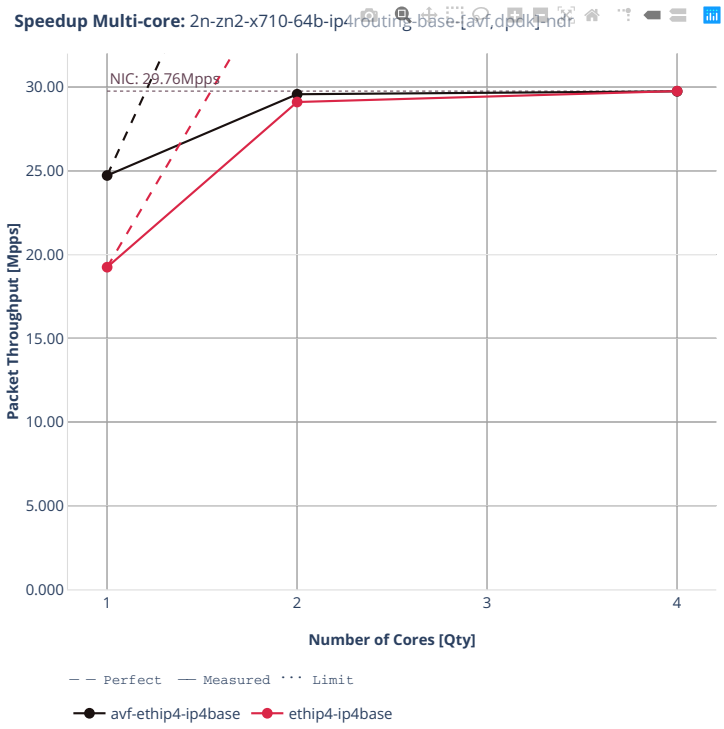
64b-features-ip4routing-base-avf

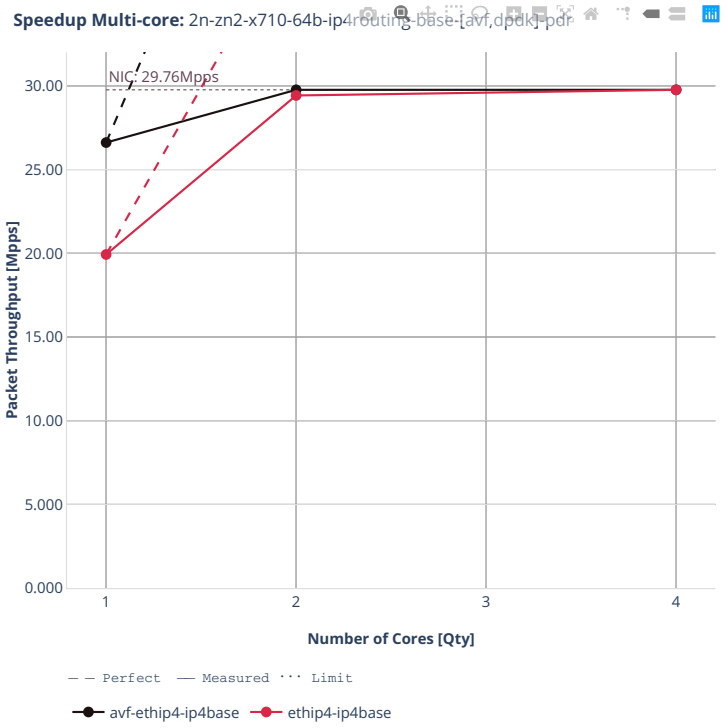



Speedup Multi-core: 2n-zn2-xxv710-64b-features-ip4routing-base-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-ip4base-iacl50sf-10kflows
avf-ethip4udp-ip4base-iacl50sl-10kflows
avf-ethip4udp-ip4base-oacl50sf-10kflows
avf-ethip4udp-ip4base-oacl50sl-10kflows
avf-ethip4udp-nat44det-h1024-p63-s64512

2n-zn2-x710

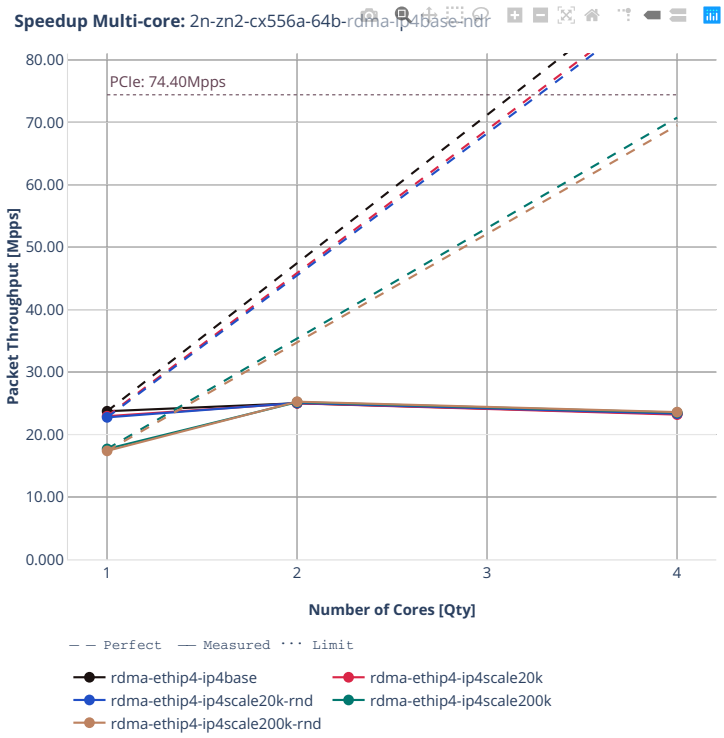
64b-ip4routing-base-[avf,dpdk]

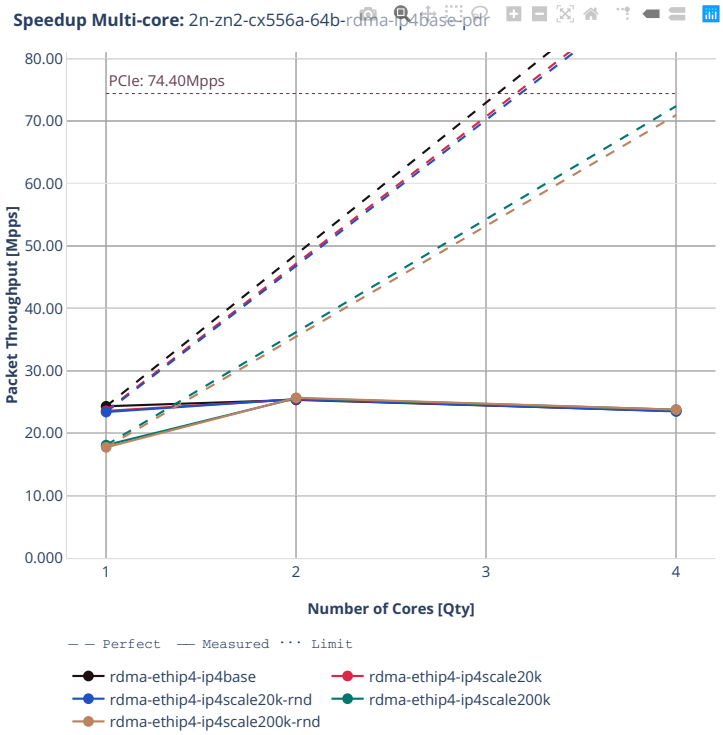



Speedup Multi-core: 2n-zn2-x710-64b-ip4routing-base-[avf,dpdk]-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4-ip4base
ethip4-ip4base

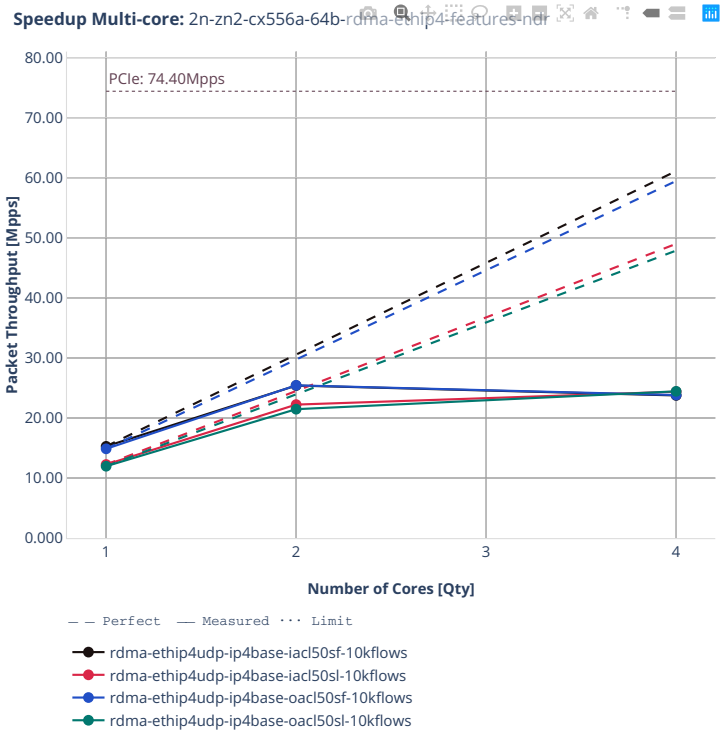
2n-zn2-cx556a

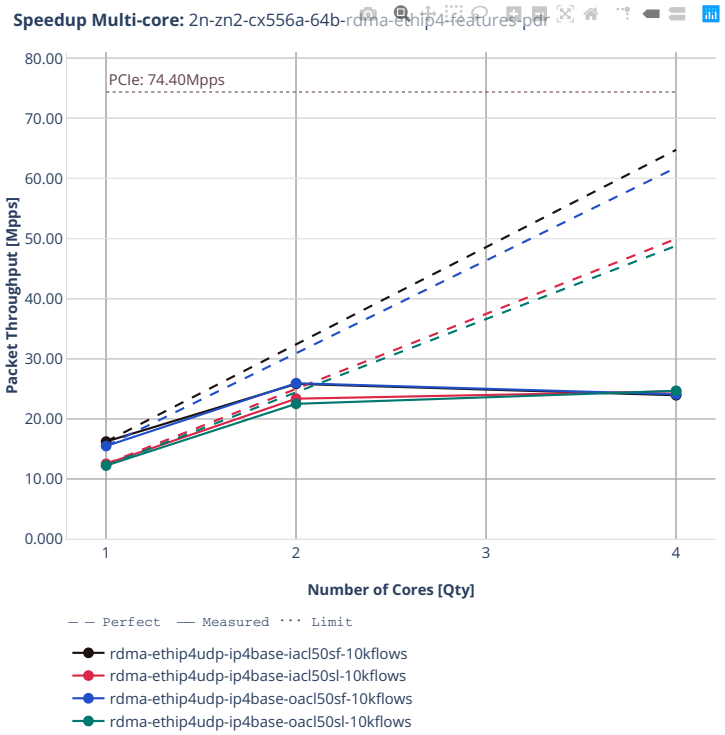
64b-ip4routing-base-scale-rdma-core




Speedup Multi-core: 2n-zn2-cx556a-64b-rdma-ip4base-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-ethip4-ip4base
rdma-ethip4-ip4scale20k
rdma-ethip4-ip4scale20k-rnd
rdma-ethip4-ip4scale200k
rdma-ethip4-ip4scale200k-rnd

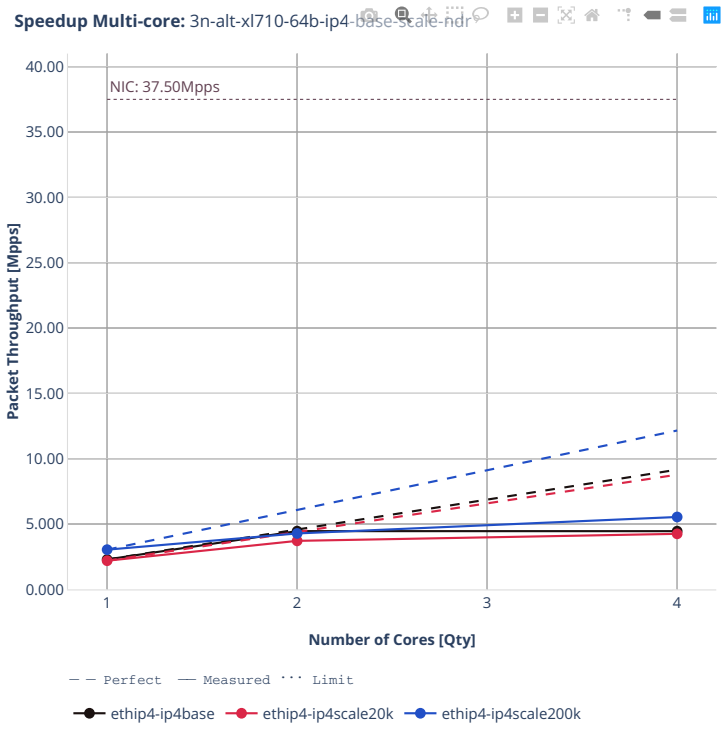
64b-ip4routing-features

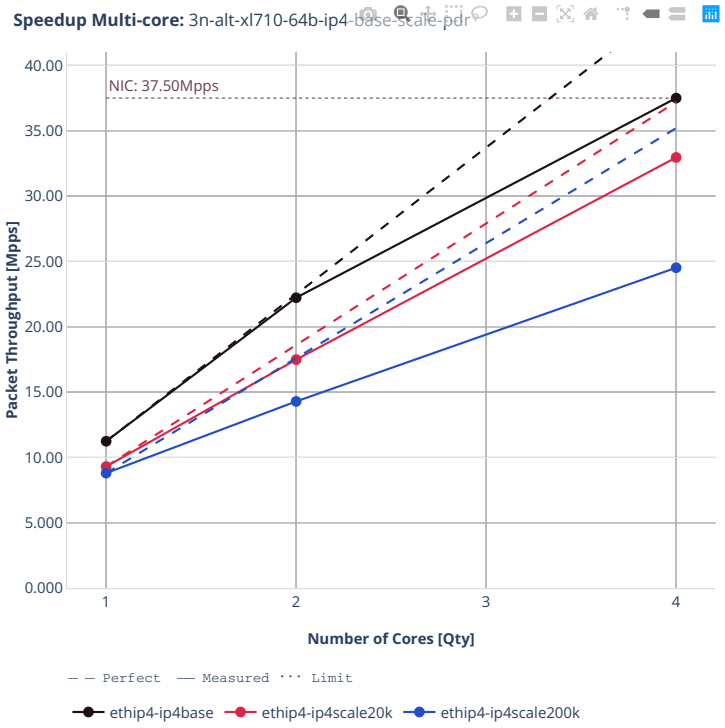



Speedup Multi-core: 2n-zn2-cx556a-64b-rdma-ethip4-features-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-ethip4udp-ip4base-iacl50sf-10kflows
rdma-ethip4udp-ip4base-iacl50sl-10kflows
rdma-ethip4udp-ip4base-oacl50sf-10kflows
rdma-ethip4udp-ip4base-oacl50sl-10kflows

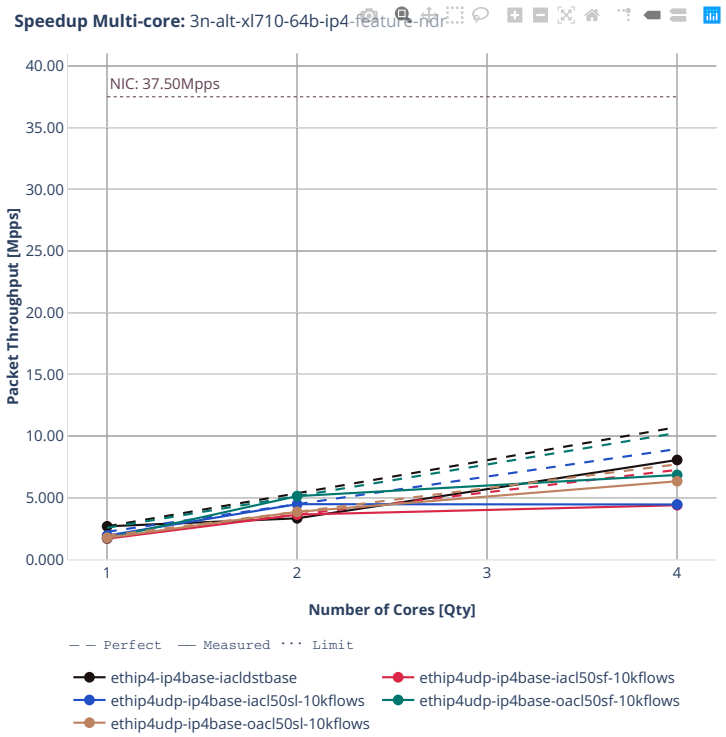
3n-alt-xl710

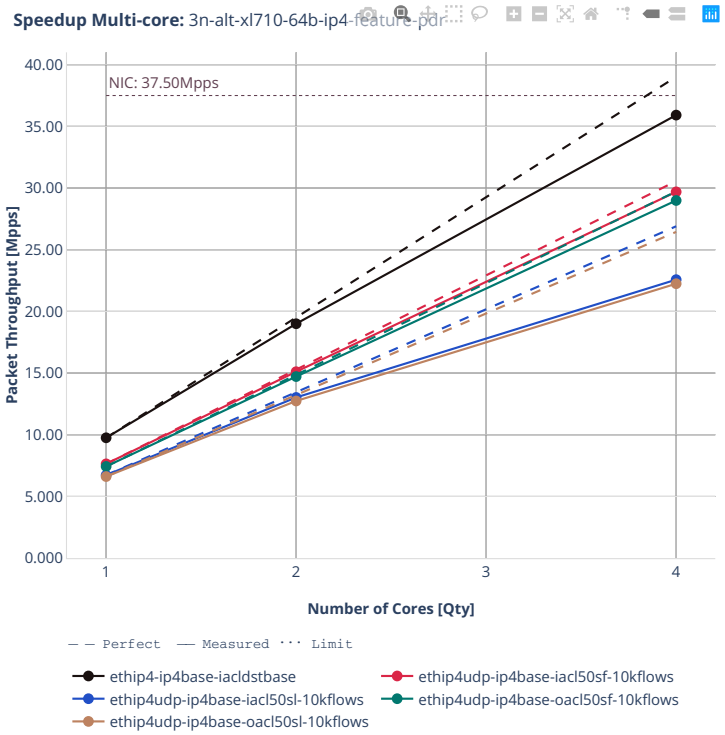
64b-ip4routing-base-scale




Speedup Multi-core: 3n-alt-xl710-64b-ip4-base-scale-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale200k

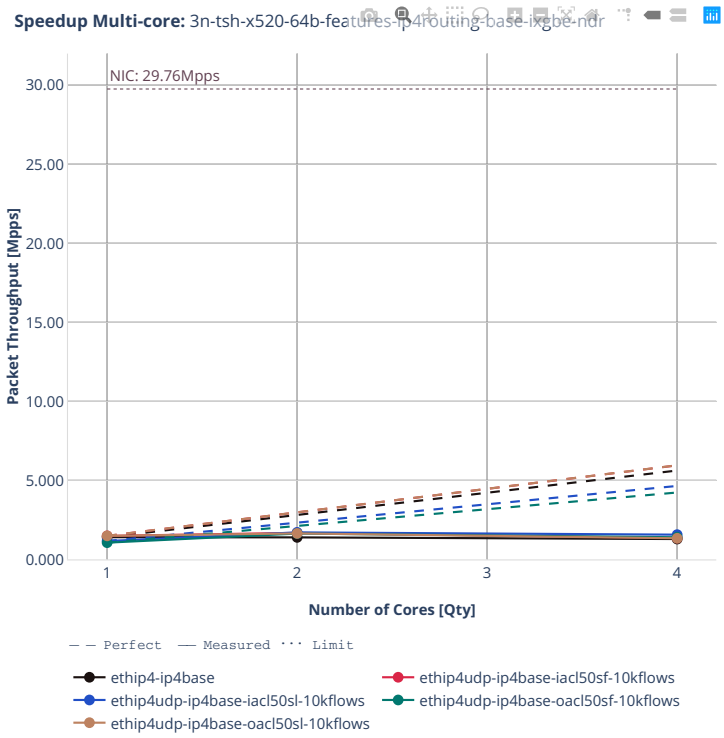
64b-ip4routing-features

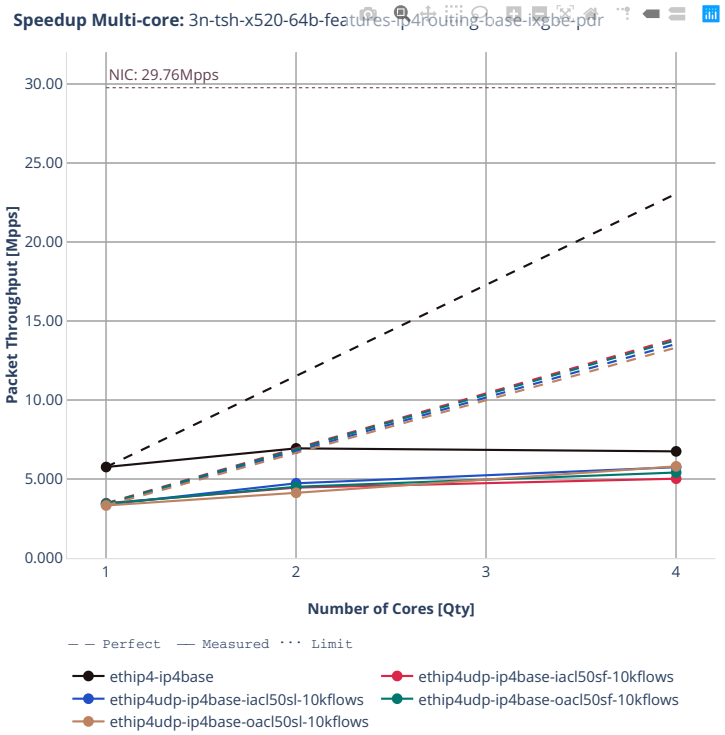



Speedup Multi-core: 3n-alt-xl710-64b-ip4-feature-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base-iacldstbase
ethip4udp-ip4base-iacl50sf-10kflows
ethip4udp-ip4base-iacl50sl-10kflows
ethip4udp-ip4base-oacl50sf-10kflows
ethip4udp-ip4base-oacl50sl-10kflows

3n-tsh-x520

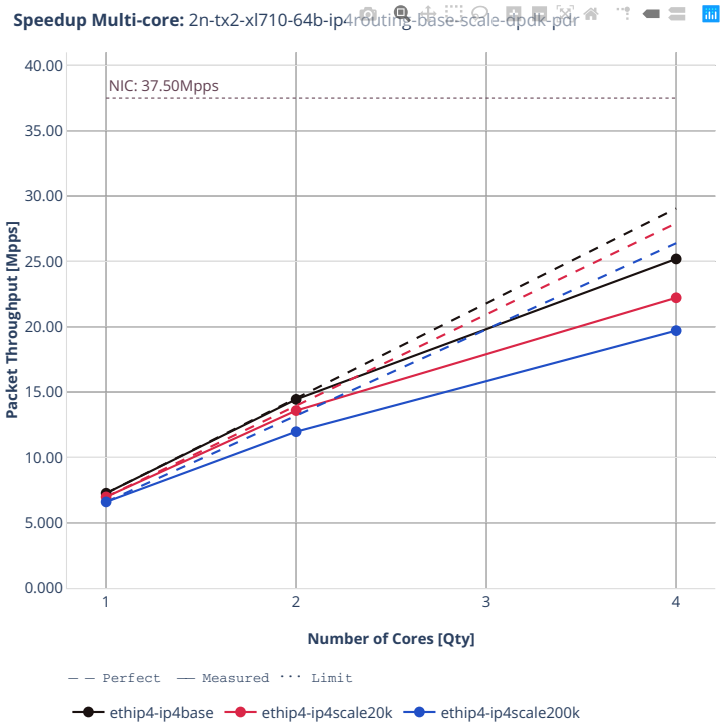
64b-features-ip4routing-base-ixgbe



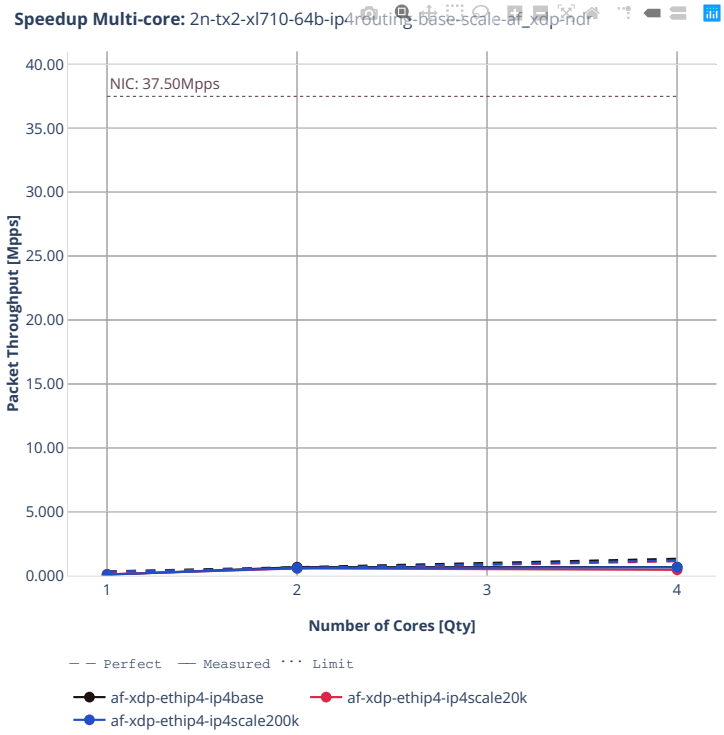

Speedup Multi-core: 3n-tsh-x520-64b-features-ip4routing-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4udp-ip4base-iacl50sf-10kflows
ethip4udp-ip4base-iacl50sl-10kflows
ethip4udp-ip4base-oacl50sf-10kflows
ethip4udp-ip4base-oacl50sl-10kflows

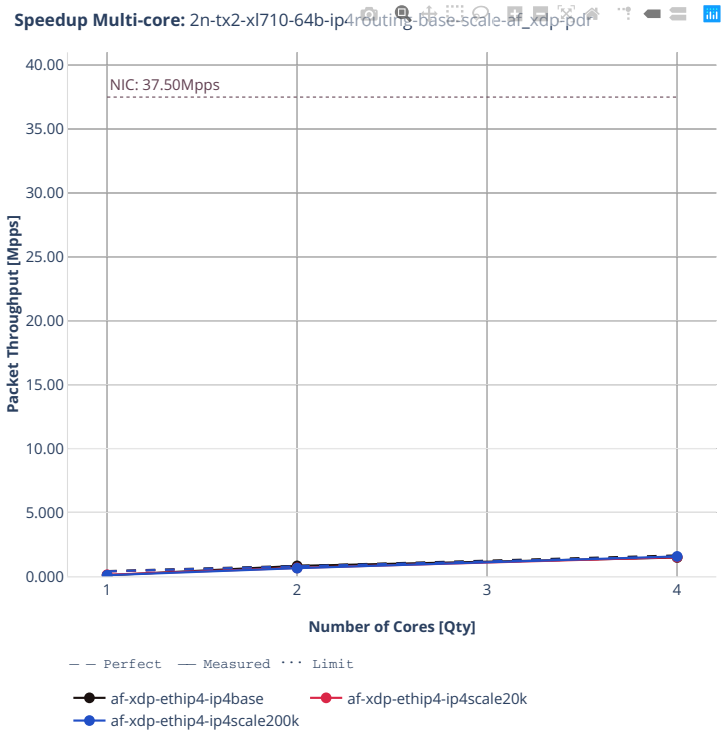
2n-tx2-xl710

64b-ip4routing-base-scale-dpdk

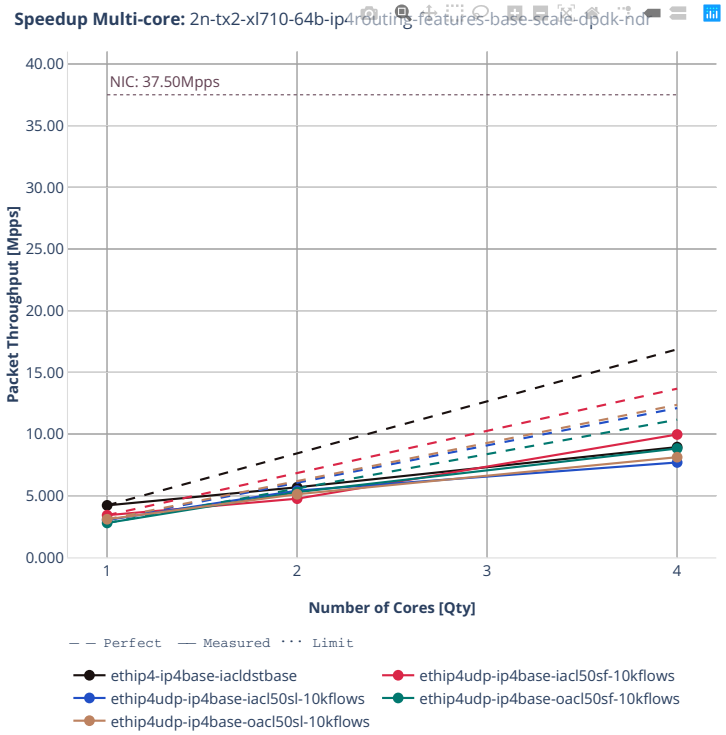

Speedup Multi-core: 2n-tx2-xl710-64b-ip4routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale200k

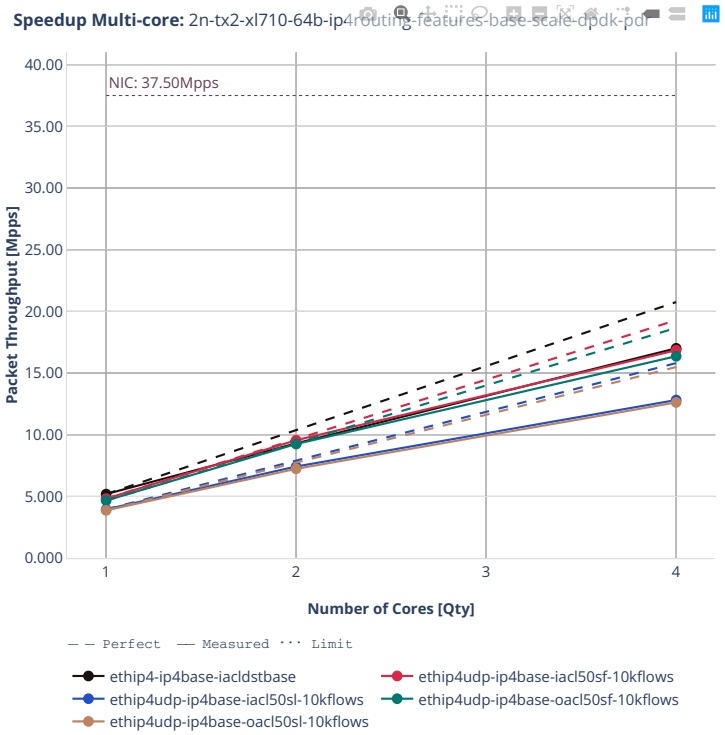
64b-ip4routing-base-scale-af-xdp




Speedup Multi-core: 2n-tx2-xl710-64b-ip4routing-base-scale-af_xdp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
af-xdp-ethip4-ip4base
af-xdp-ethip4-ip4scale20k
af-xdp-ethip4-ip4scale200k

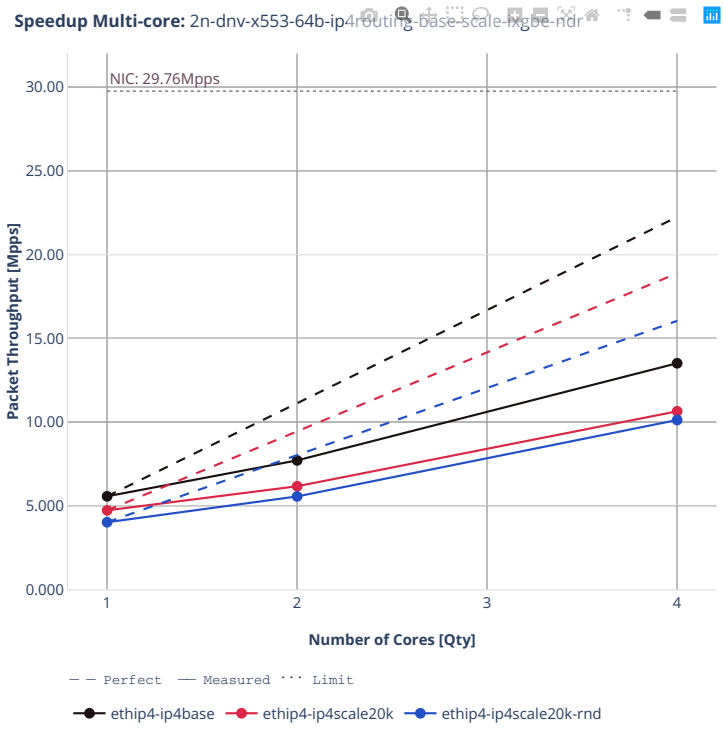
64b-features-ip4routing-base-dpdk

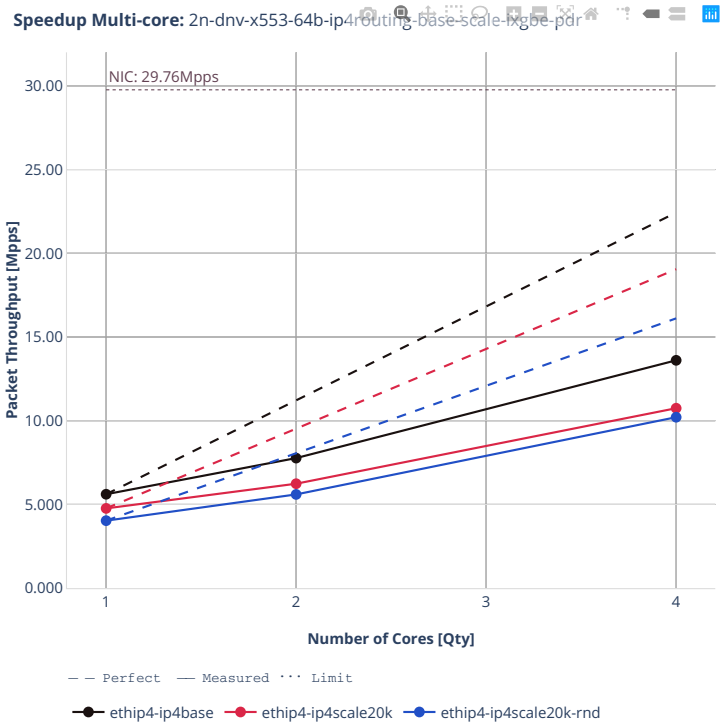



Speedup Multi-core: 2n-tx2-xl710-64b-ip4routing-features-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base-iacldstbase
ethip4udp-ip4base-iacl50sf-10kflows
ethip4udp-ip4base-iacl50sl-10kflows
ethip4udp-ip4base-oacl50sf-10kflows
ethip4udp-ip4base-oacl50sl-10kflows

2n-dnv-x553

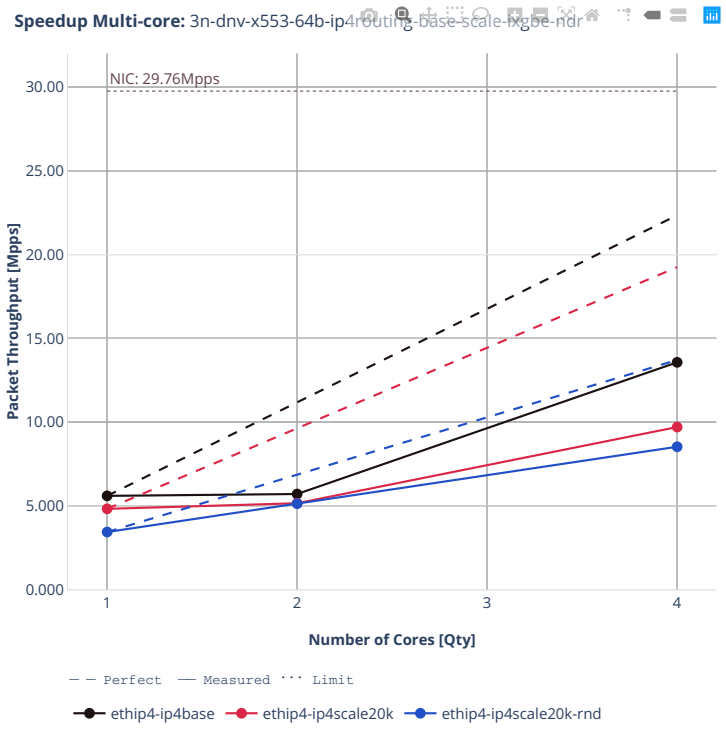
64b-ip4routing-base-scale-ixgbe

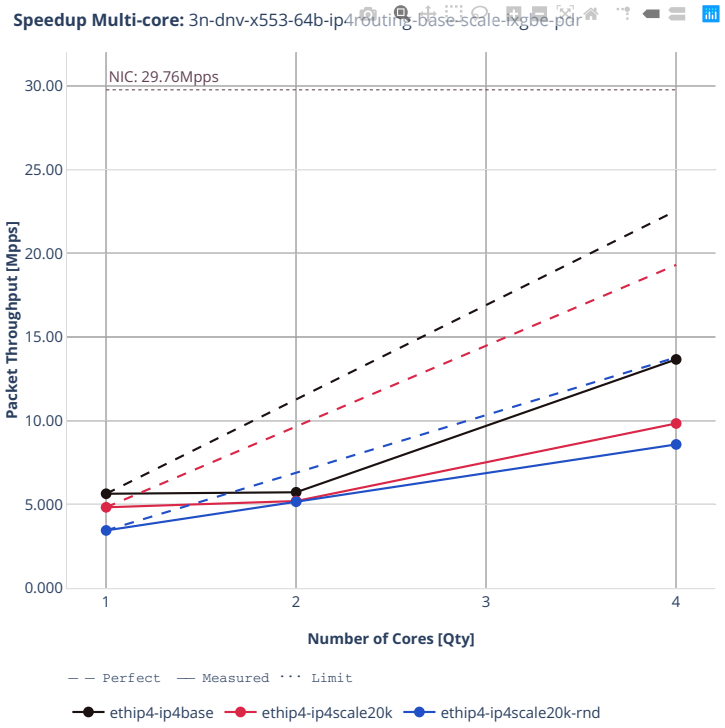



Speedup Multi-core: 2n-dnv-x553-64b-ip4routing-base-scale-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale20k-rnd

3n-dnv-x553

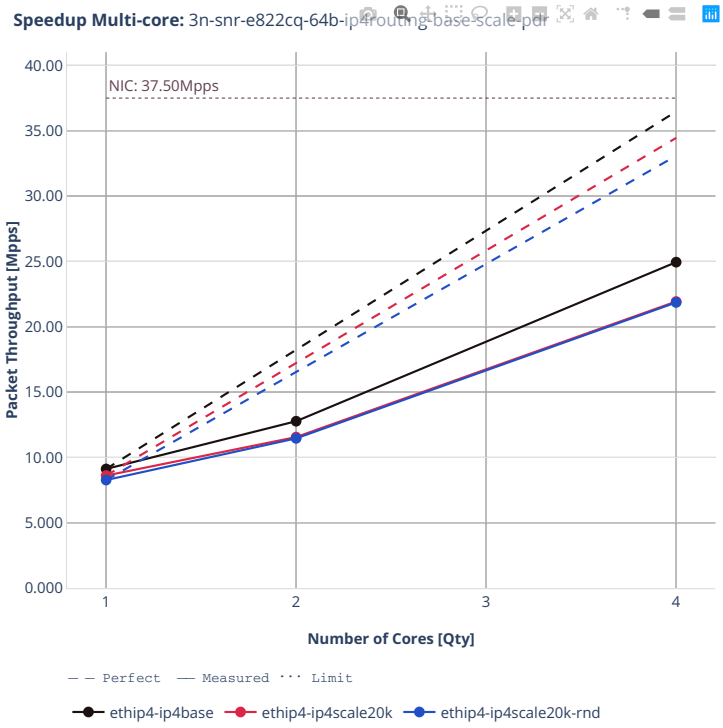
64b-ip4routing-base-scale-ixgbe




Speedup Multi-core: 3n-dnv-x553-64b-ip4routing-base-scale-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale20k-rnd

3n-snr-e822cq

64b-ip4routing-base-scale


Speedup Multi-core: 3n-snr-e822cq-64b-ip4routing-base-scale-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base
ethip4-ip4scale20k
ethip4-ip4scale20k-rnd

2.4.3 IPv6 Routing

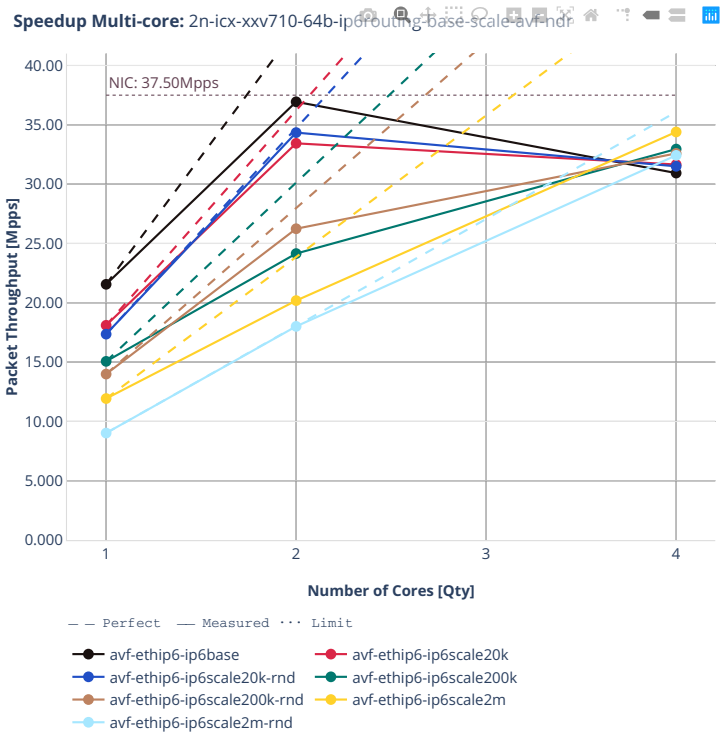
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 78B performance tests with VPP IPv6 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

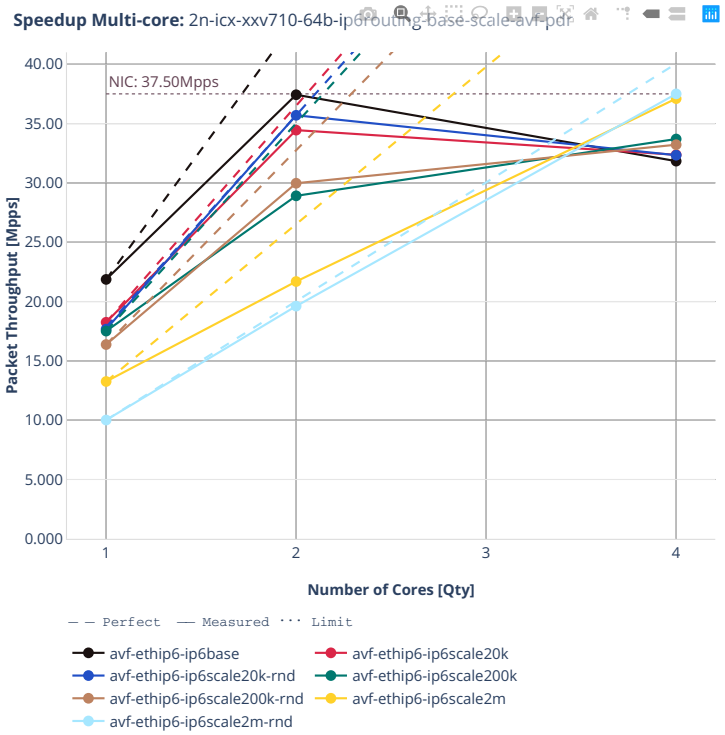
CSIT source code for the test cases used for plots can be found in CSIT git repository[135].

[135] https://git.fd.io/csit/tree/tests/vpp/perf/ip6?h=rls2210

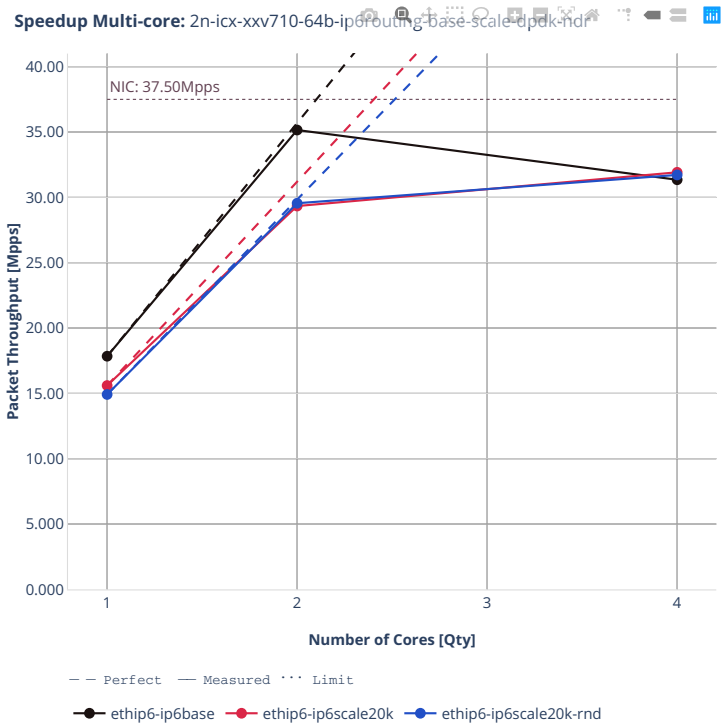
2n-icx-xxv710

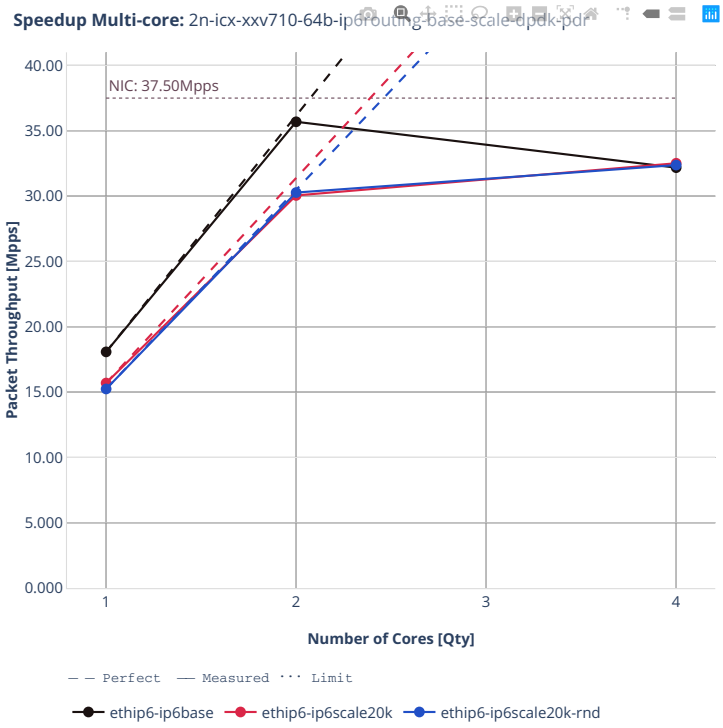
78b-ip6routing-base-scale-avf



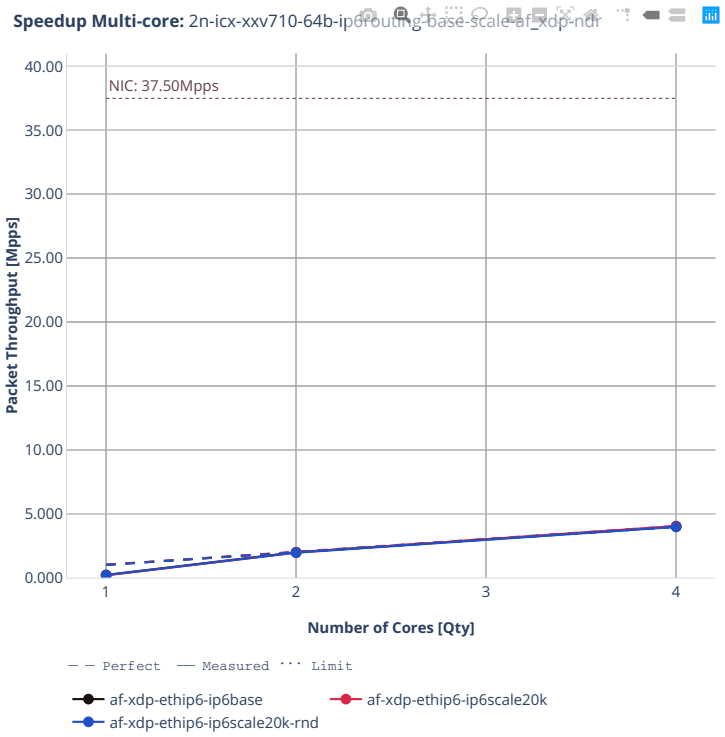

Speedup Multi-core: 2n-icx-xxv710-64b-ip6routing-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6-ip6base
avf-ethip6-ip6scale20k
avf-ethip6-ip6scale20k-rnd
avf-ethip6-ip6scale200k
avf-ethip6-ip6scale200k-rnd
avf-ethip6-ip6scale2m
avf-ethip6-ip6scale2m-rnd

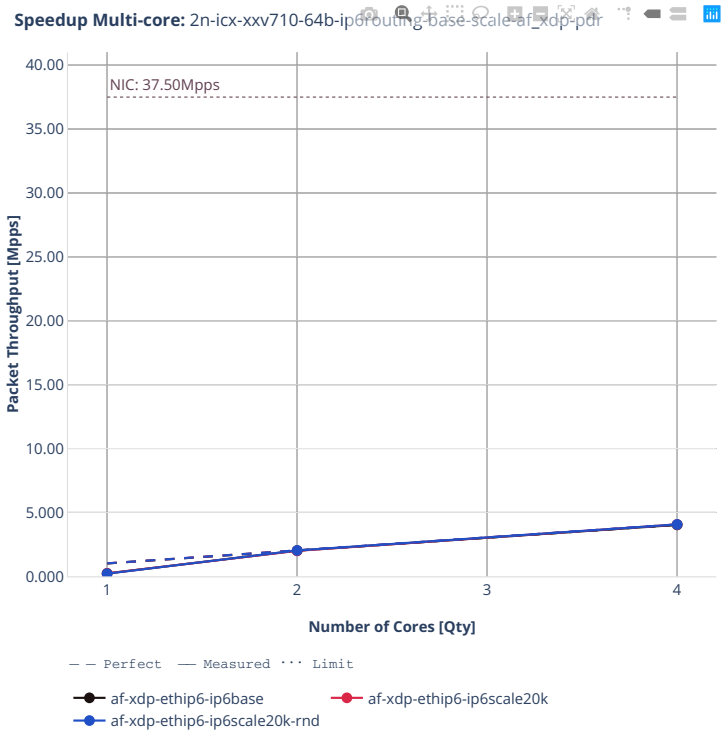
78b-ip6routing-base-scale-dpdk




Speedup Multi-core: 2n-icx-xxv710-64b-ip6routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k
ethip6-ip6scale20k-rnd

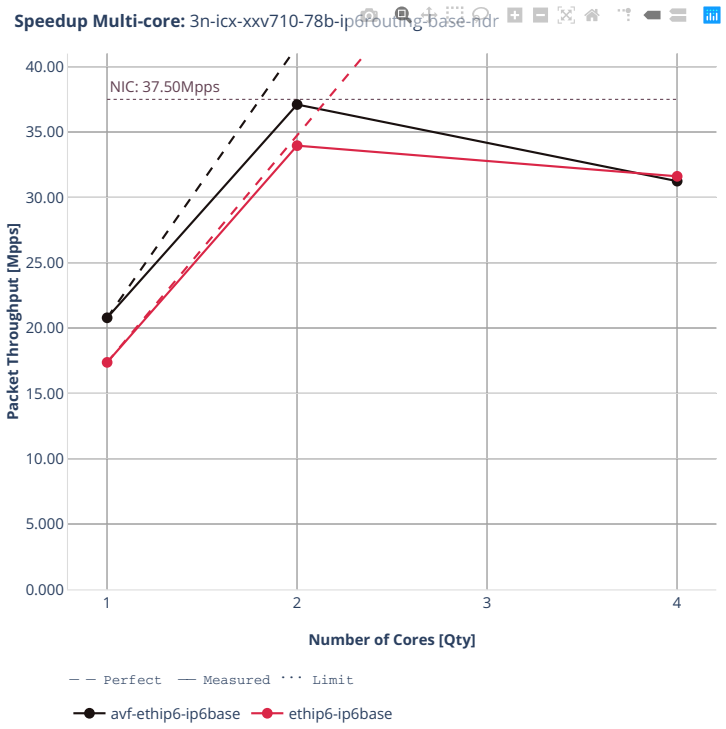
78b-ip6routing-base-scale-af_xdp

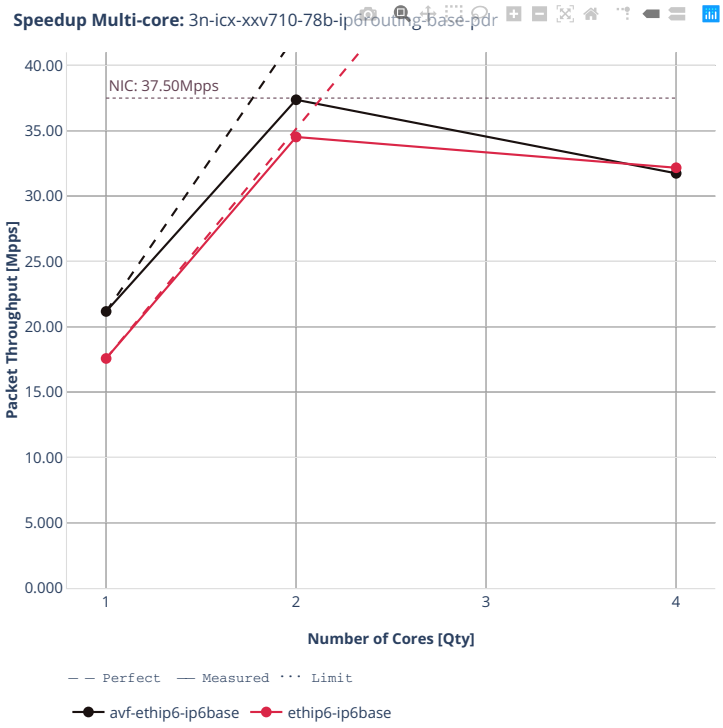



Speedup Multi-core: 2n-icx-xxv710-64b-ip6routing-base-scale-af_xdp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
af-xdp-ethip6-ip6base
af-xdp-ethip6-ip6scale20k
af-xdp-ethip6-ip6scale20k-rnd

3n-icx-xxv710

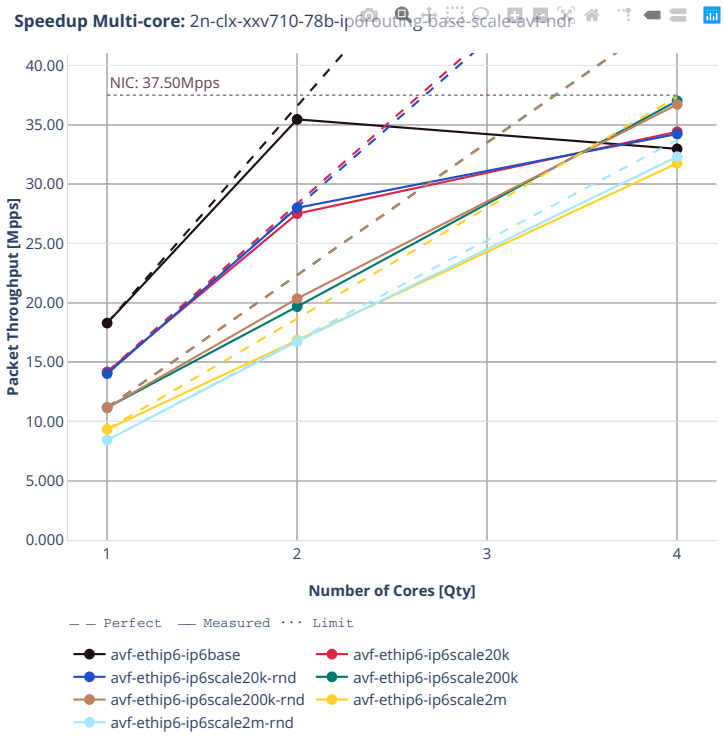
78b-ip6routing-base

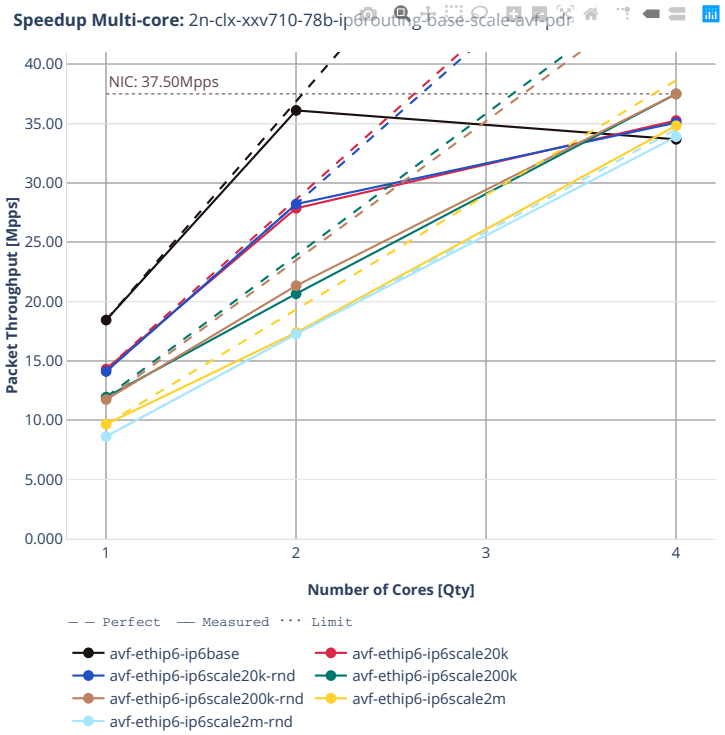



Speedup Multi-core: 3n-icx-xxv710-78b-ip6routing-base-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6-ip6base
ethip6-ip6base

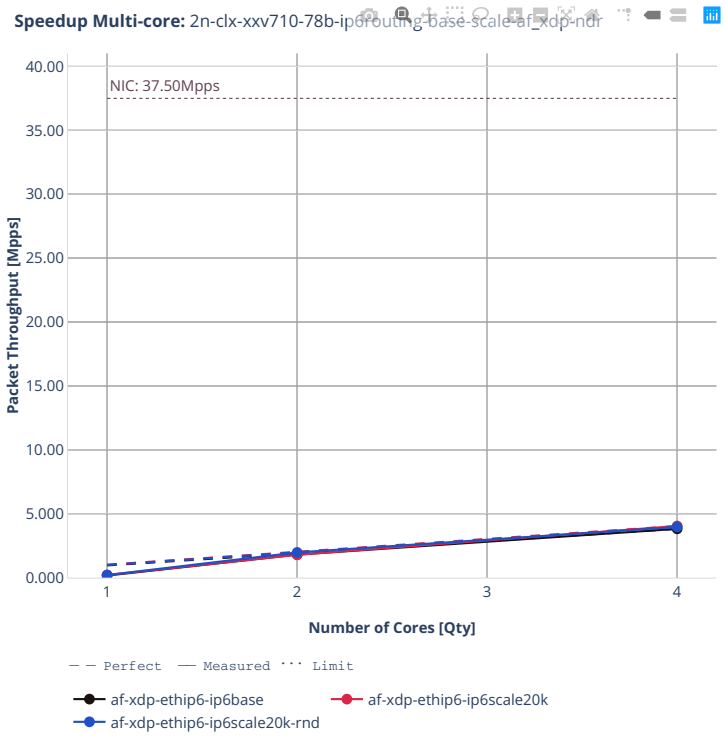
2n-clx-xxv710

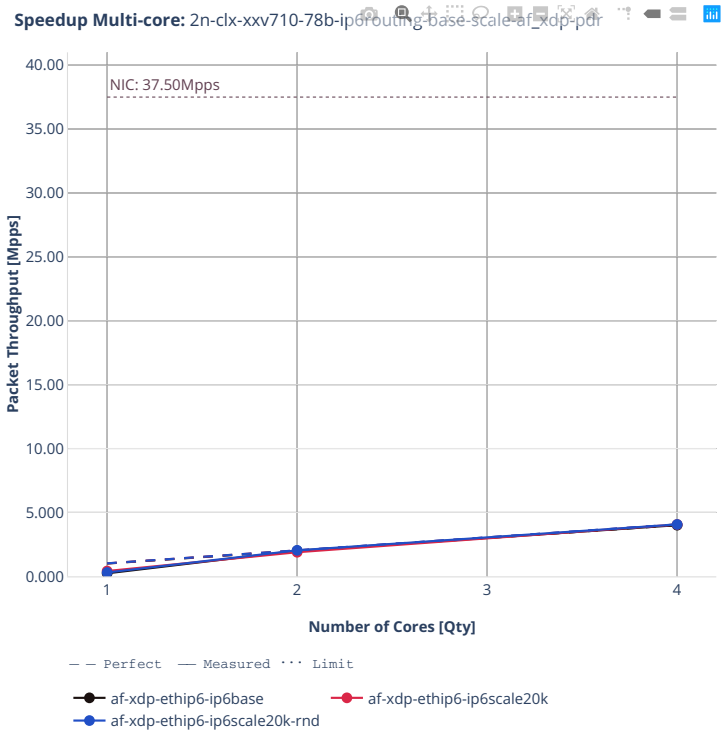
78b-ip6routing-base-scale-avf



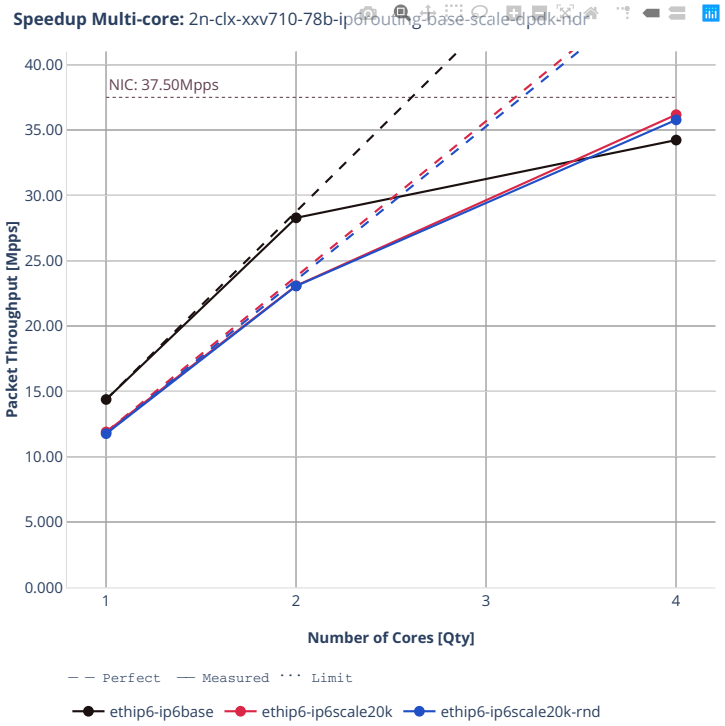

Speedup Multi-core: 2n-clx-xxv710-78b-ip6routing-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6-ip6base
avf-ethip6-ip6scale20k
avf-ethip6-ip6scale20k-rnd
avf-ethip6-ip6scale200k
avf-ethip6-ip6scale200k-rnd
avf-ethip6-ip6scale2m
avf-ethip6-ip6scale2m-rnd

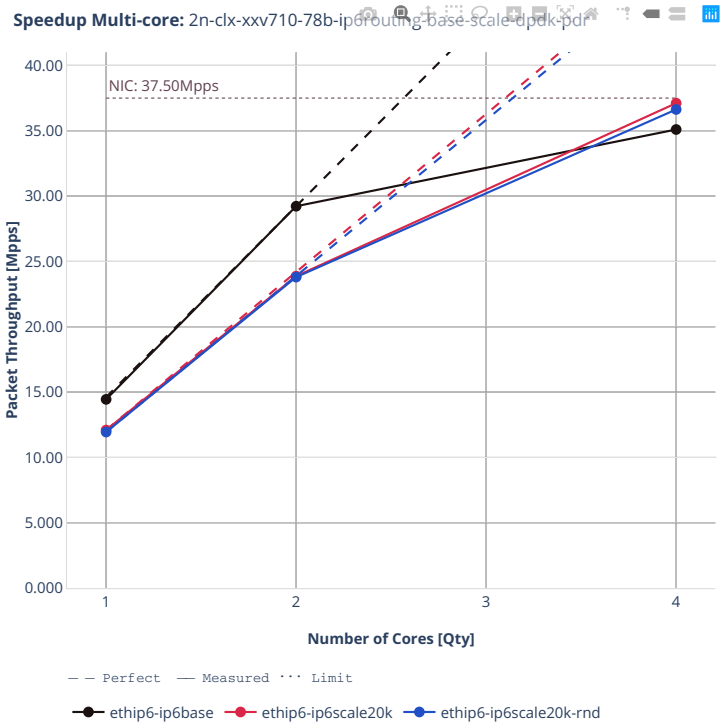
78b-ip6routing-base-scale-af_xdp




Speedup Multi-core: 2n-clx-xxv710-78b-ip6routing-base-scale-af_xdp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
af-xdp-ethip6-ip6base
af-xdp-ethip6-ip6scale20k
af-xdp-ethip6-ip6scale20k-rnd

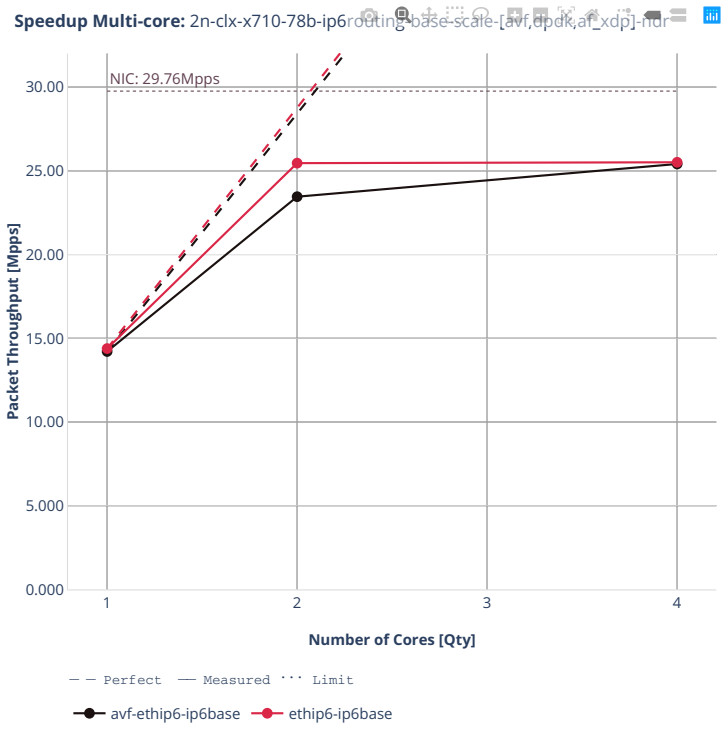
78b-ip6routing-base-scale-dpdk

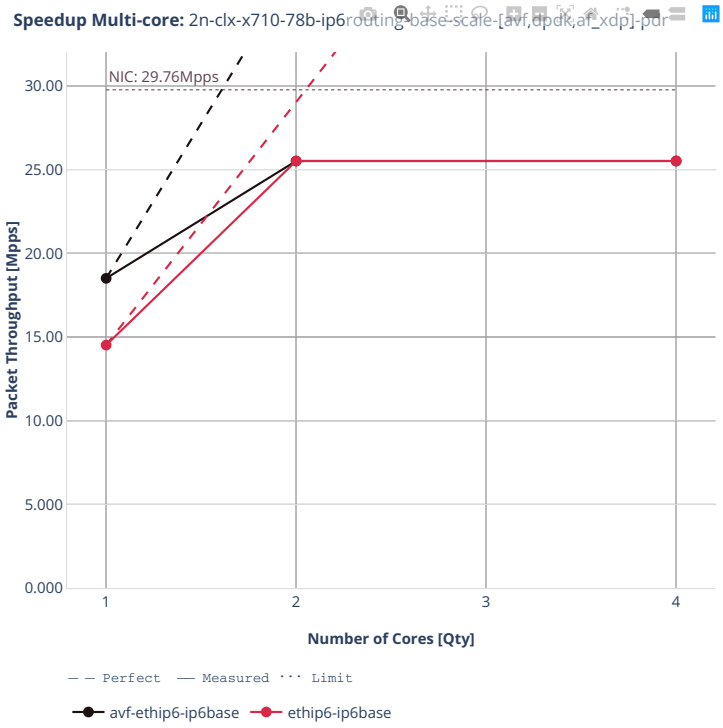



Speedup Multi-core: 2n-clx-xxv710-78b-ip6routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k
ethip6-ip6scale20k-rnd

2n-clx-x710

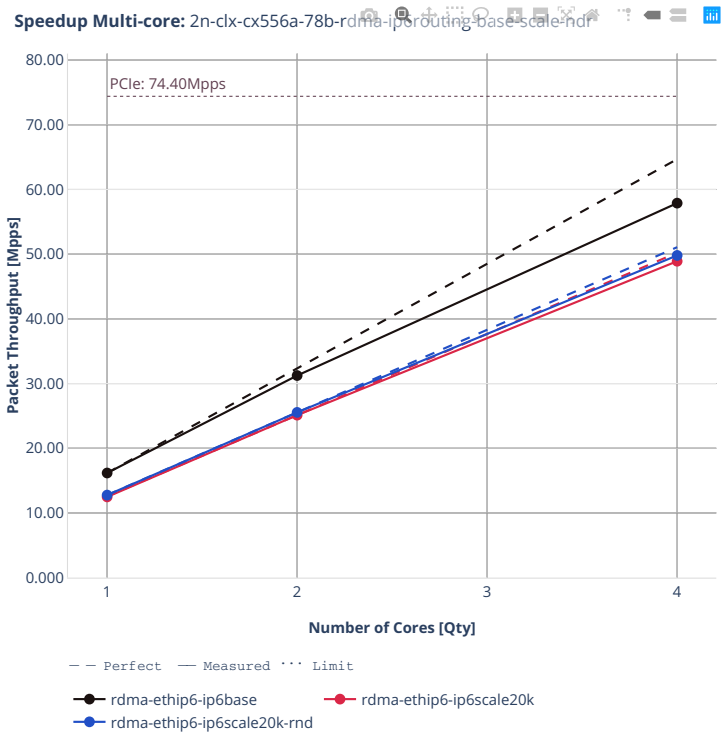
78b-ip6routing-base-scale-[avf,dpdk,af_xdp]

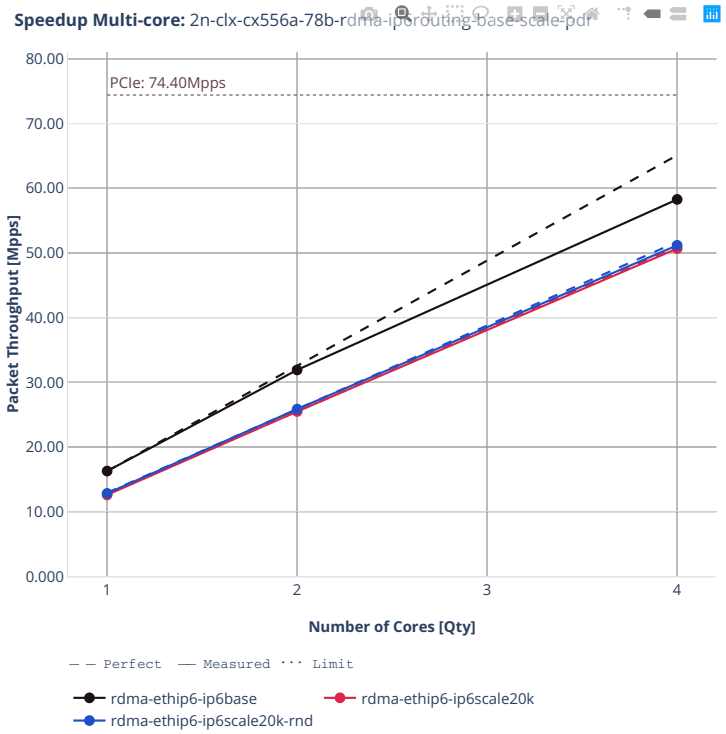



Speedup Multi-core: 2n-clx-x710-78b-ip6routing-base-scale-[avf,dpdk,af_xdp]-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6-ip6base
ethip6-ip6base

2n-clx-cx556a

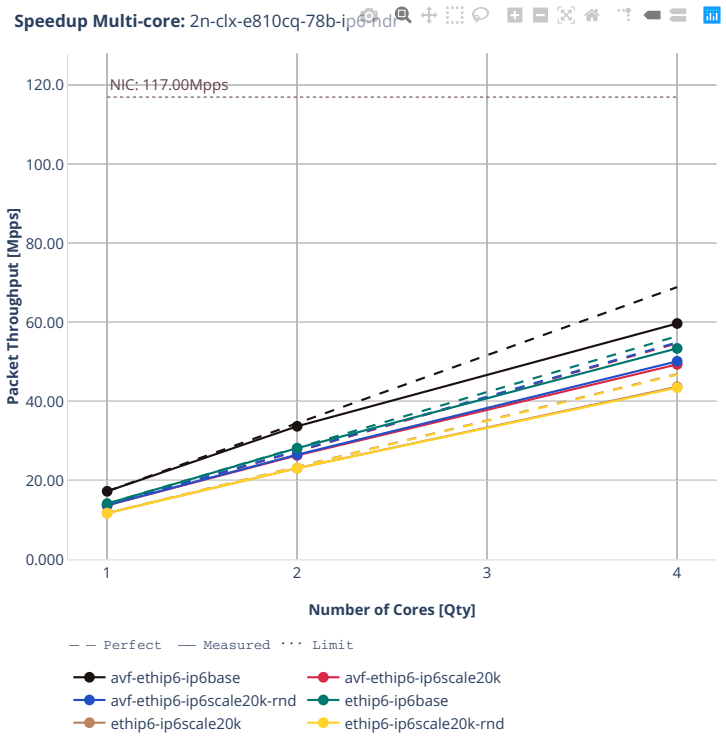
78b-ip6routing-base-scale-rdma-core

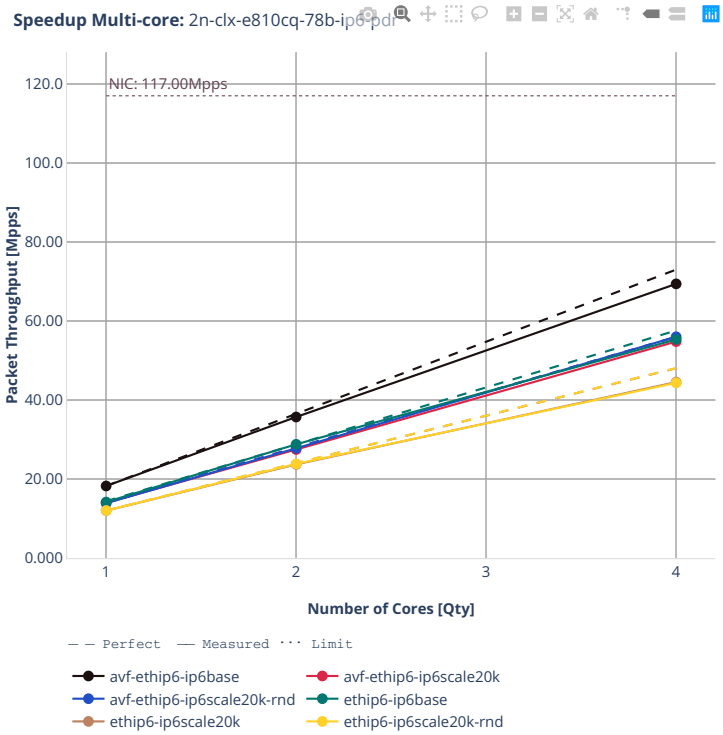



Speedup Multi-core: 2n-clx-cx556a-78b-rdma-ip6routing-base-scale-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-ethip6-ip6base
rdma-ethip6-ip6scale20k
rdma-ethip6-ip6scale20k-rnd

2n-clx-e810cq

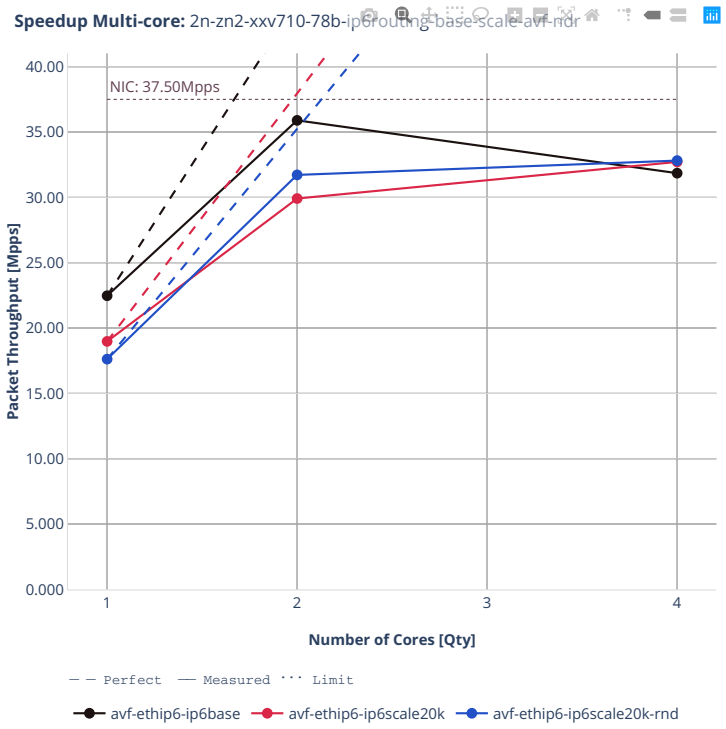
78b-ip6routing-base-scale

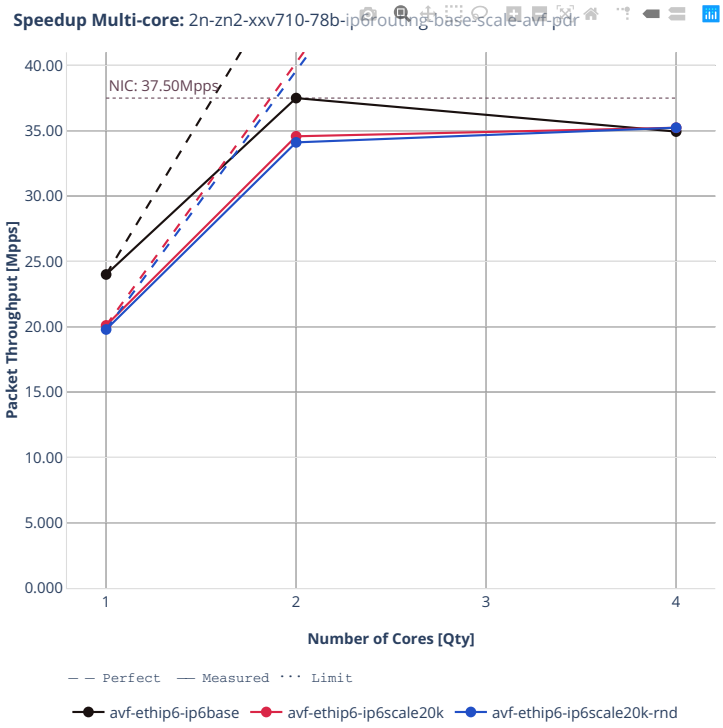



Speedup Multi-core: 2n-clx-e810cq-78b-ip6-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6-ip6base
avf-ethip6-ip6scale20k
avf-ethip6-ip6scale20k-rnd
ethip6-ip6base
ethip6-ip6scale20k
ethip6-ip6scale20k-rnd

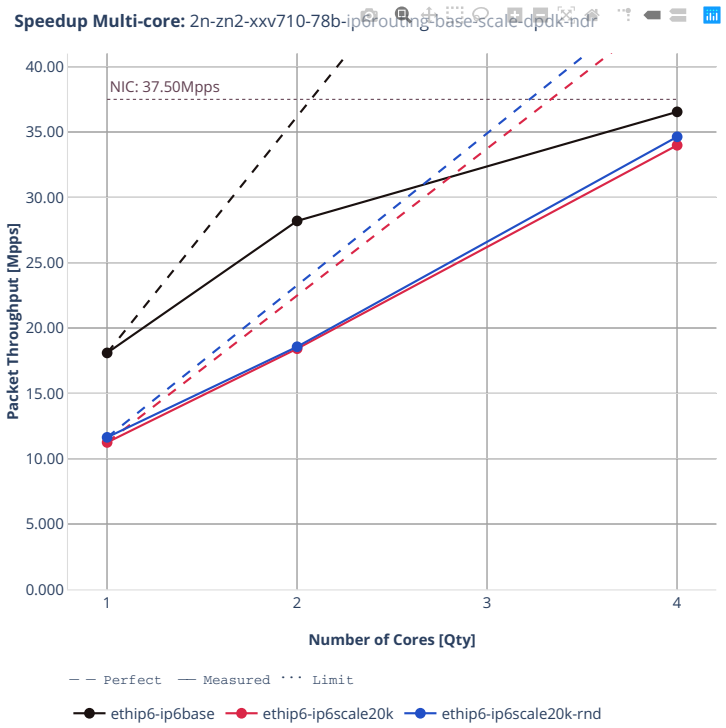
2n-zn2-xxv710

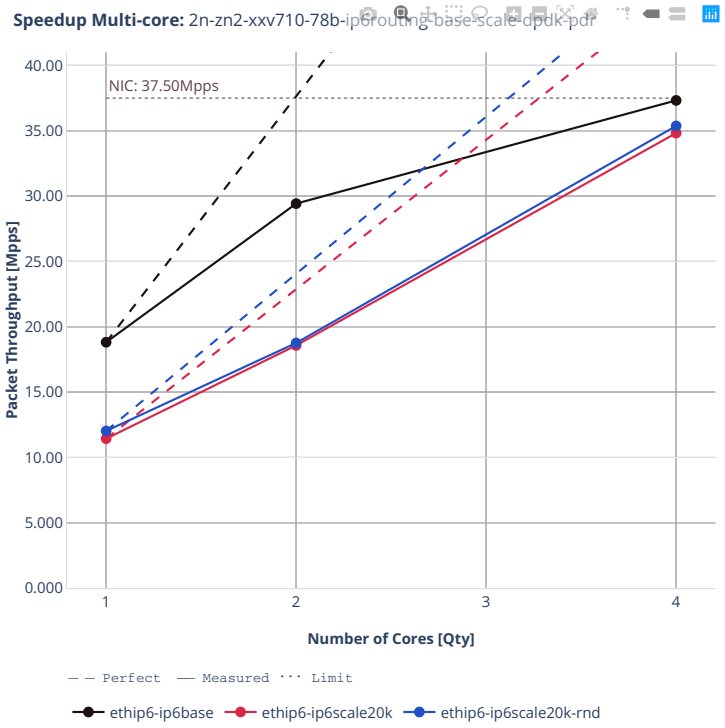
78b-ip6routing-base-scale-avf




Speedup Multi-core: 2n-zn2-xxv710-78b-ip6routing-base-scale-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6-ip6base
avf-ethip6-ip6scale20k
avf-ethip6-ip6scale20k-rnd

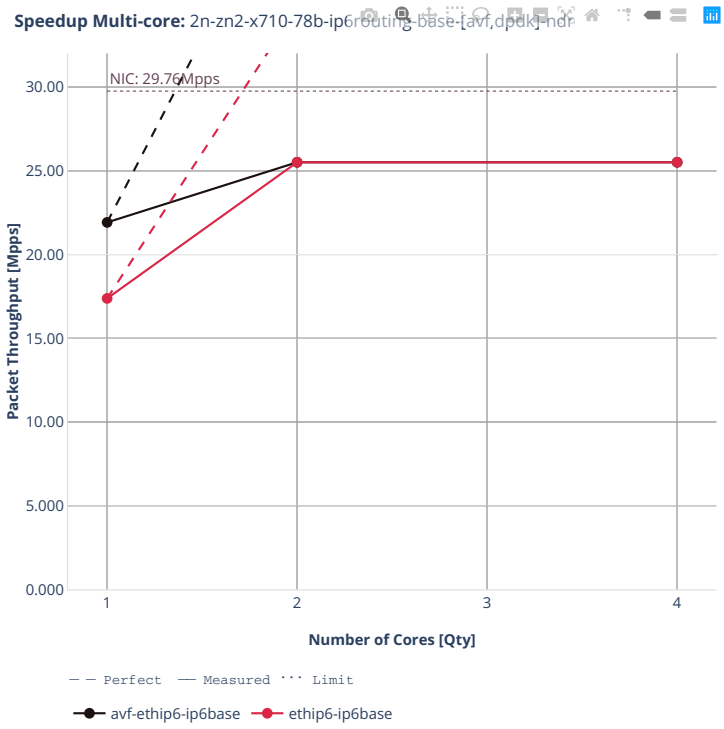
78b-ip6routing-base-scale-dpdk

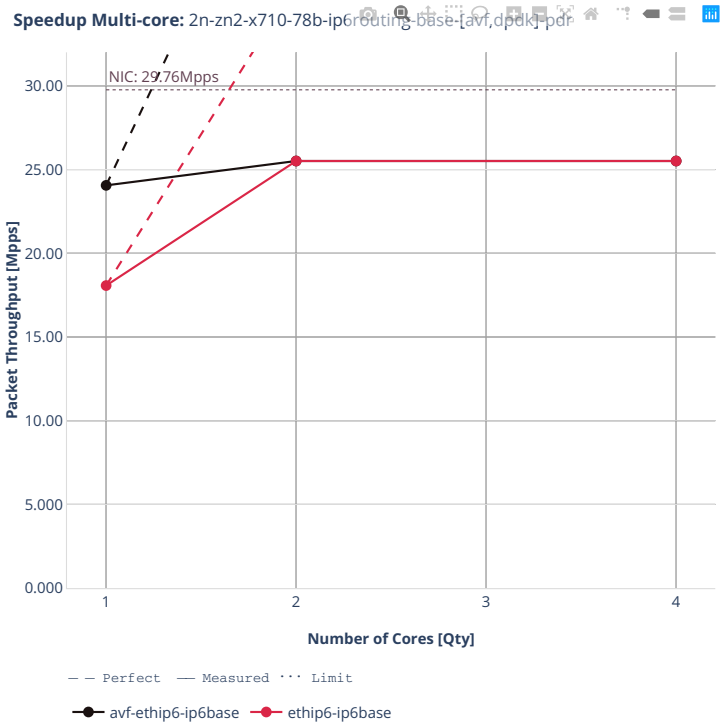



Speedup Multi-core: 2n-zn2-xxv710-78b-ip6routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k
ethip6-ip6scale20k-rnd

2n-zn2-x710

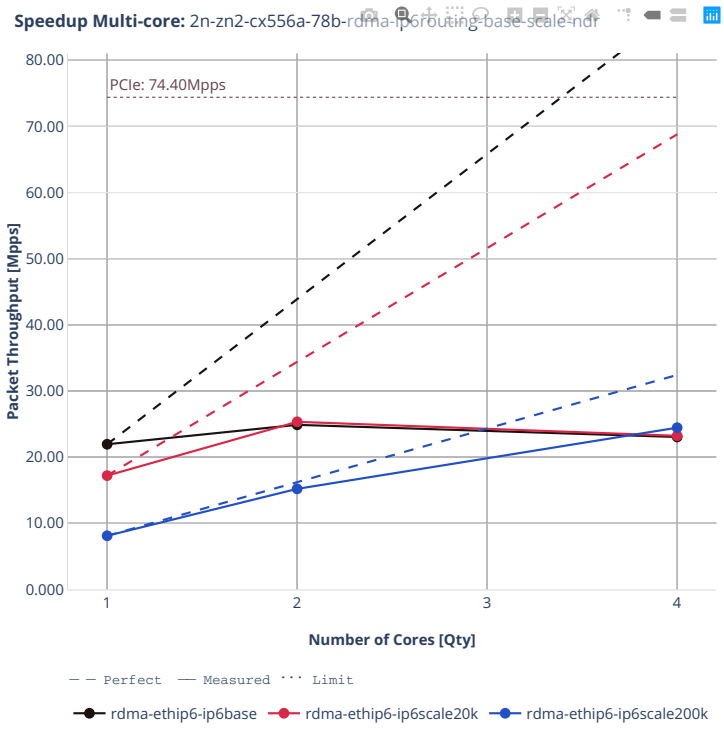
78b-ip6routing-base-[avf,dpdk]

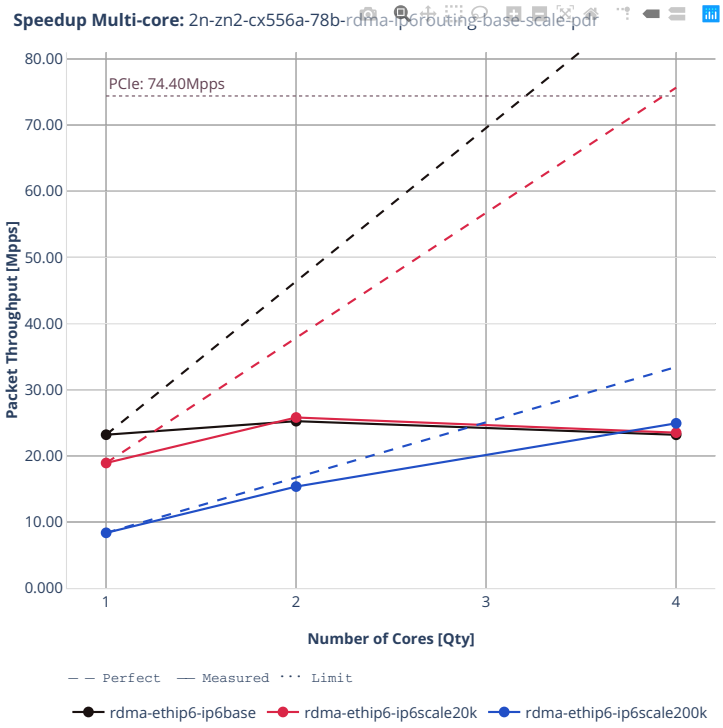



Speedup Multi-core: 2n-zn2-x710-78b-ip6routing-base-[avf,dpdk]-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6-ip6base
ethip6-ip6base

2n-zn2-cx556a

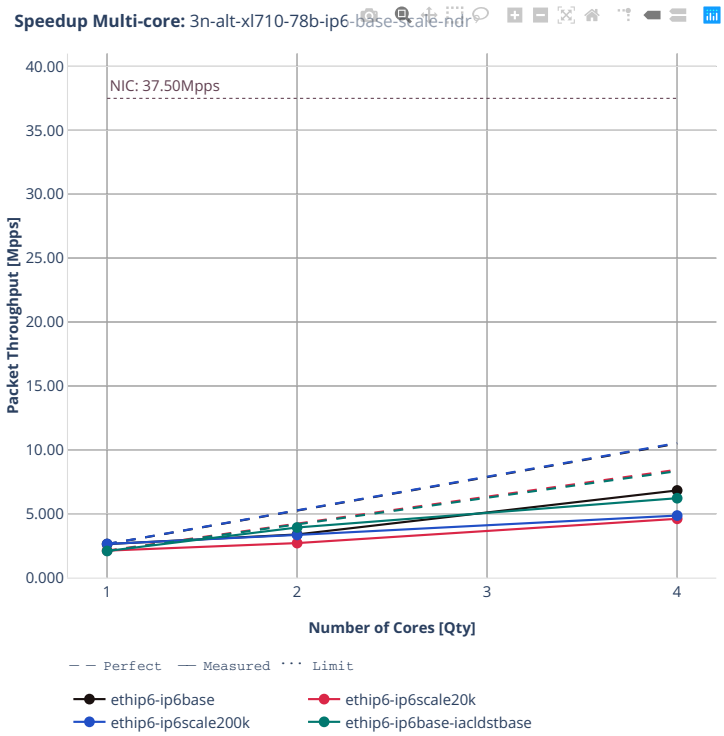
78b-ip6routing-base-scale-rdma-core




Speedup Multi-core: 2n-zn2-cx556a-78b-rdma-ip6routing-base-scale-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-ethip6-ip6base
rdma-ethip6-ip6scale20k
rdma-ethip6-ip6scale200k

3n-alt-xl710

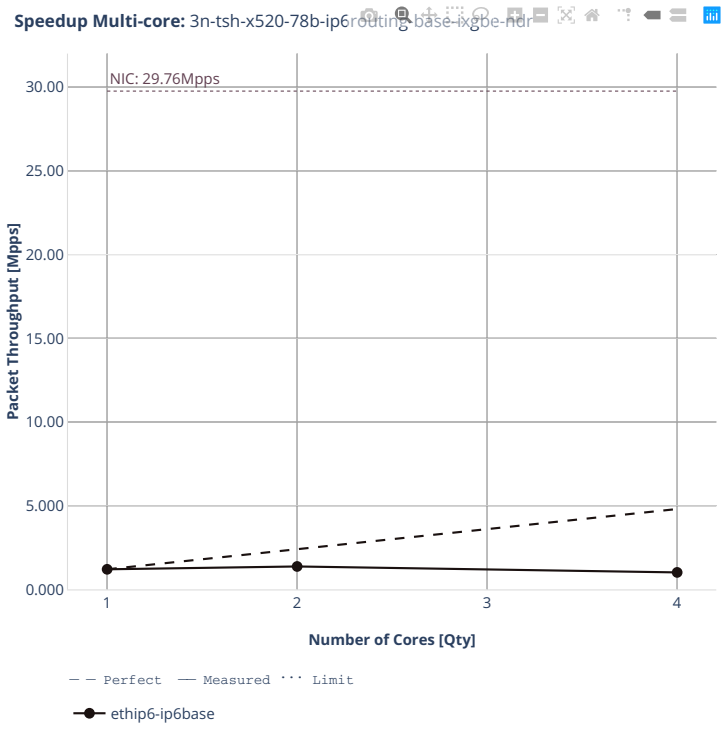
78b-ip6routing-base-scale

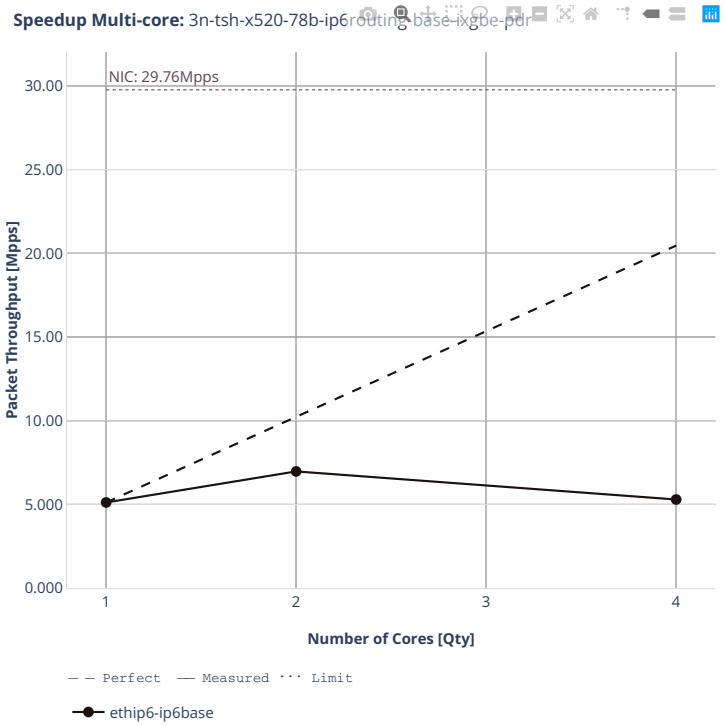



Speedup Multi-core: 3n-alt-xl710-78b-ip6-base-scale-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k
ethip6-ip6scale200k
ethip6-ip6base-iacldstbase

3n-tsh-x520

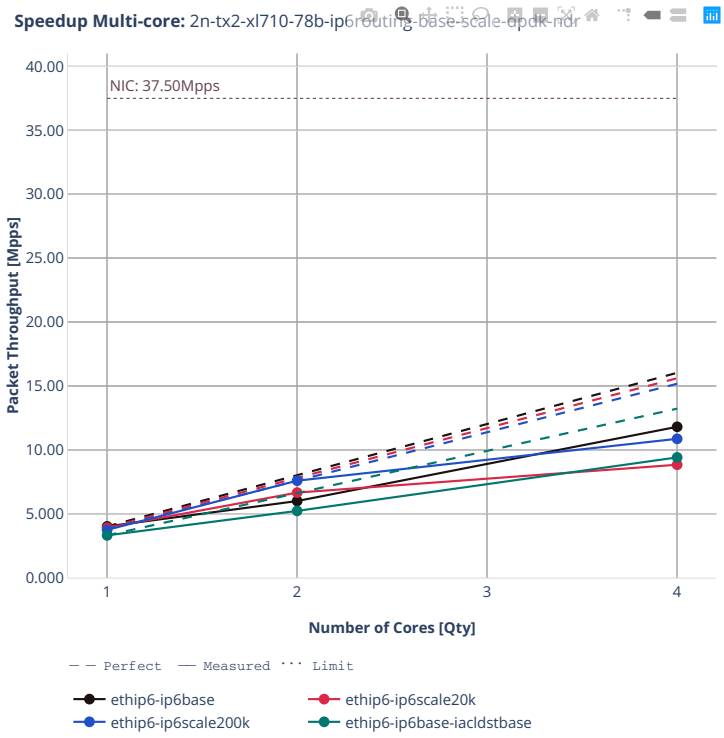
78b-ip6routing-base-ixgbe

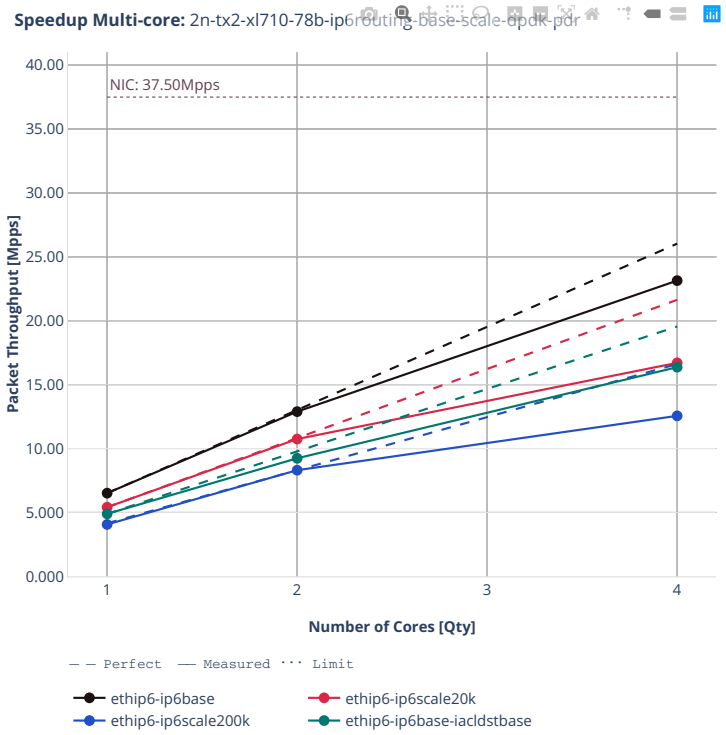



Speedup Multi-core: 3n-tsh-x520-78b-ip6routing-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base

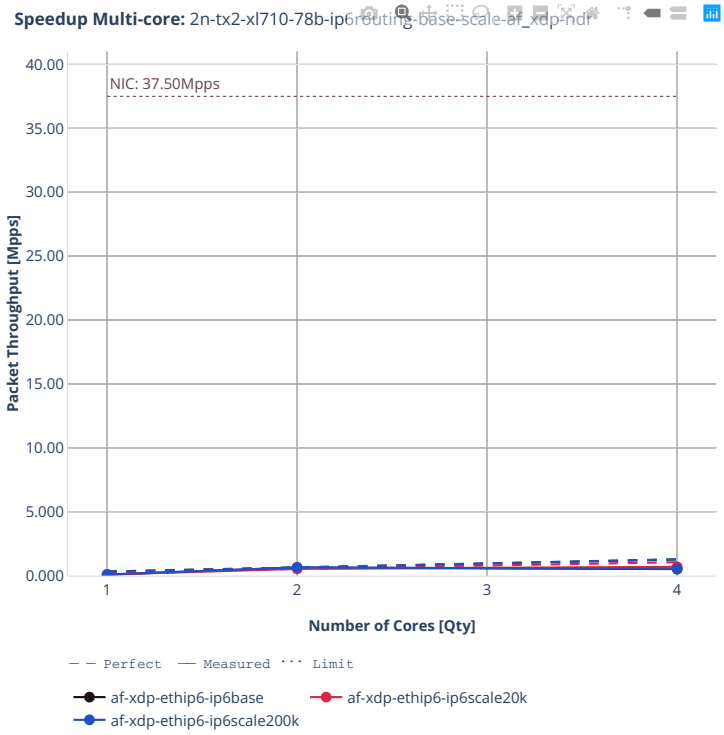
2n-tx2-xl710

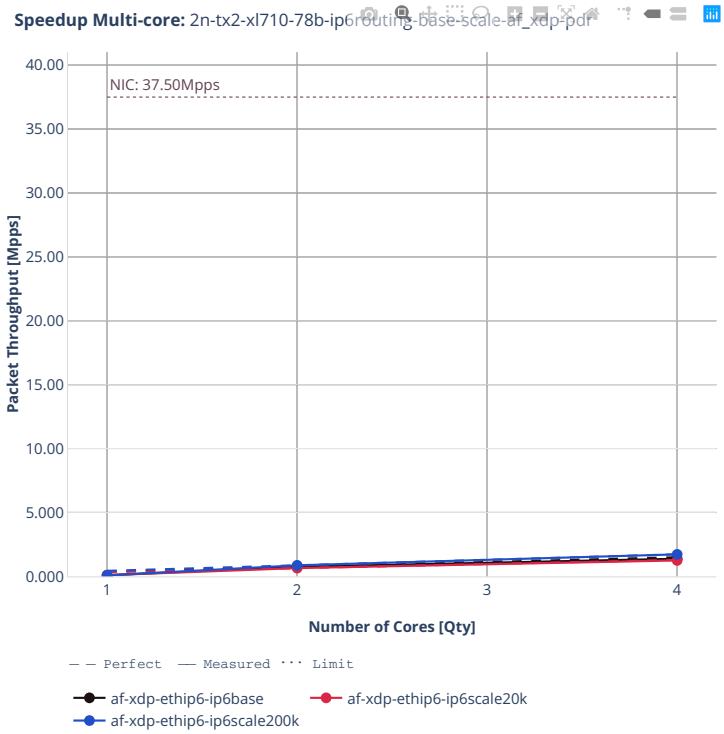
78b-ip6routing-base-scale-dpdk




Speedup Multi-core: 2n-tx2-xl710-78b-ip6routing-base-scale-dpdk-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k
ethip6-ip6scale200k
ethip6-ip6base-iacldstbase

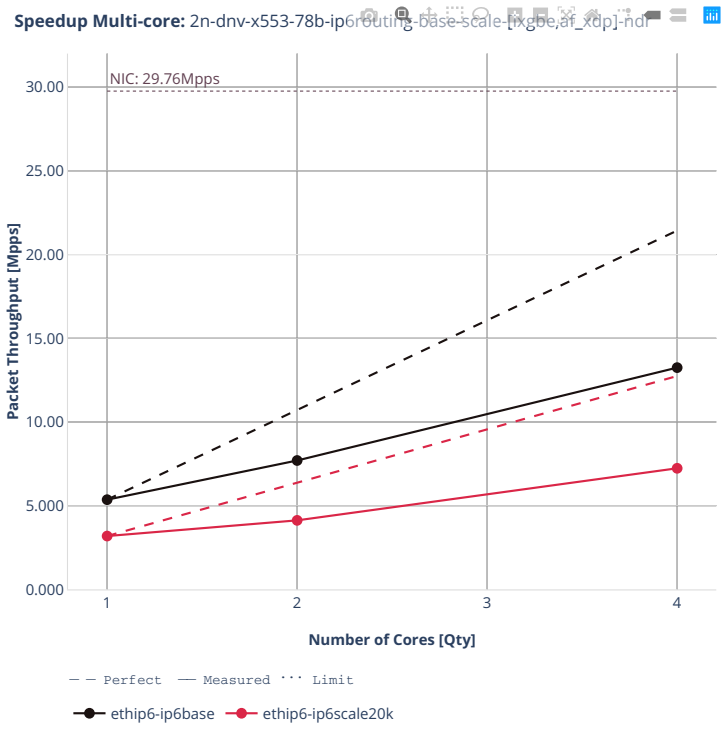
78b-ip6routing-base-scale-af-xdp

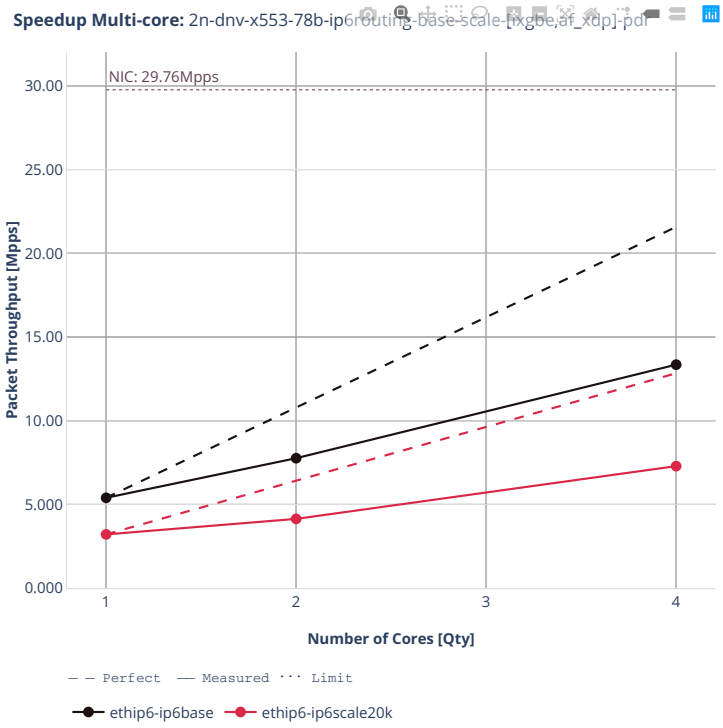



Speedup Multi-core: 2n-tx2-xl710-78b-ip6routing-base-scale-af_xdp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
af-xdp-ethip6-ip6base
af-xdp-ethip6-ip6scale20k
af-xdp-ethip6-ip6scale200k

2n-dnv-x553

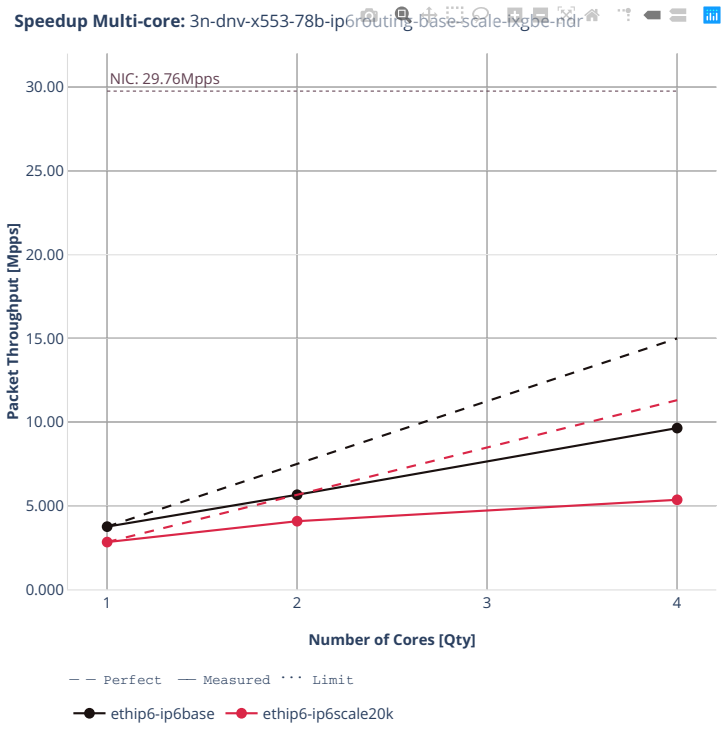
78b-ip6routing-base-scale-ixgbe

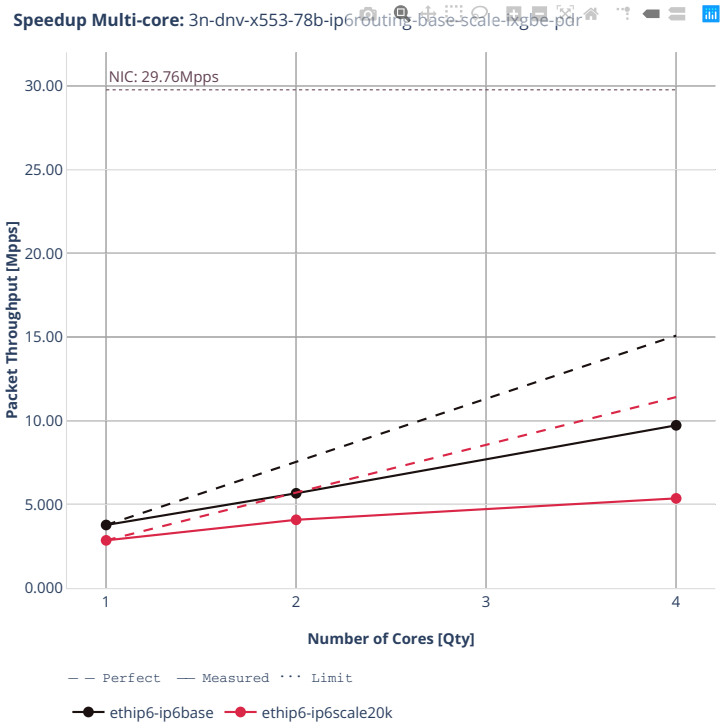



Speedup Multi-core: 2n-dnv-x553-78b-ip6routing-base-scale-[ixgbe,af_xdp]-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k

3n-dnv-x553

78b-ip6routing-base-scale-ixgbe




Speedup Multi-core: 3n-dnv-x553-78b-ip6routing-base-scale-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k

3n-snr-e822cq

78b-ip6routing-base-scale


Speedup Multi-core: 3n-snr-e822cq-78b-ip6routing-base-scale-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6-ip6base
ethip6-ip6scale20k
ethip6-ip6scale20k-rnd

2.4.4 SRv6 Routing

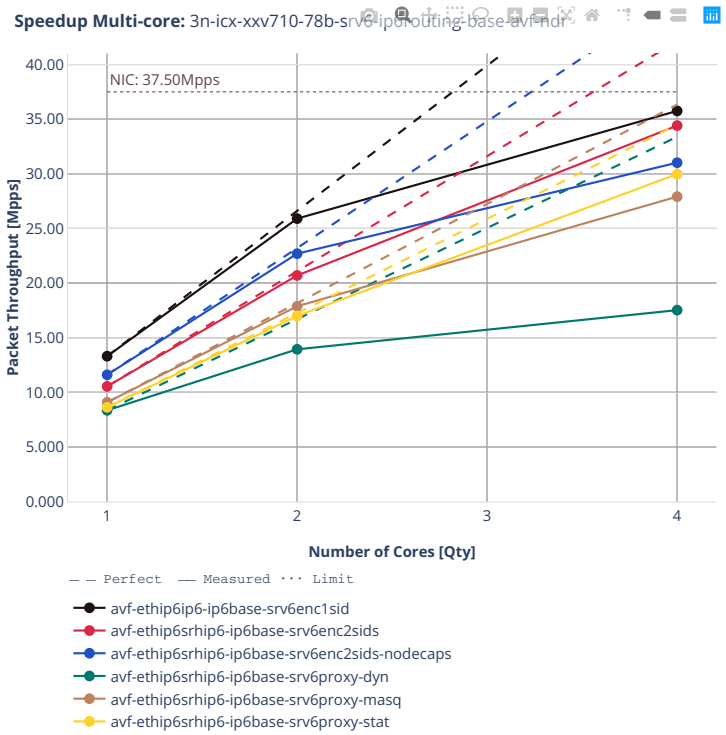
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 78B performance tests with VPP SRv6, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

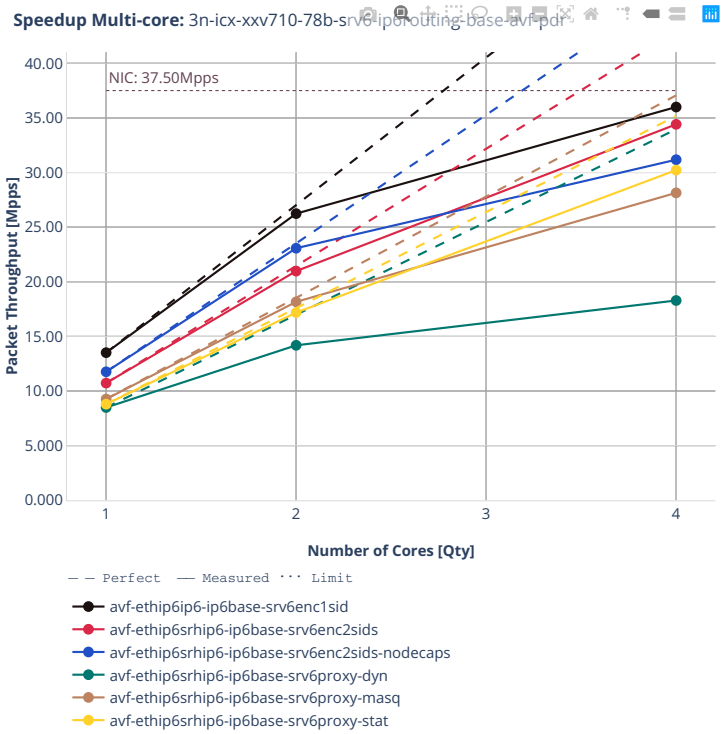
CSIT source code for the test cases used for plots can be found in CSIT git repository[136].

[136] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls2210

3n-icx-xxv710

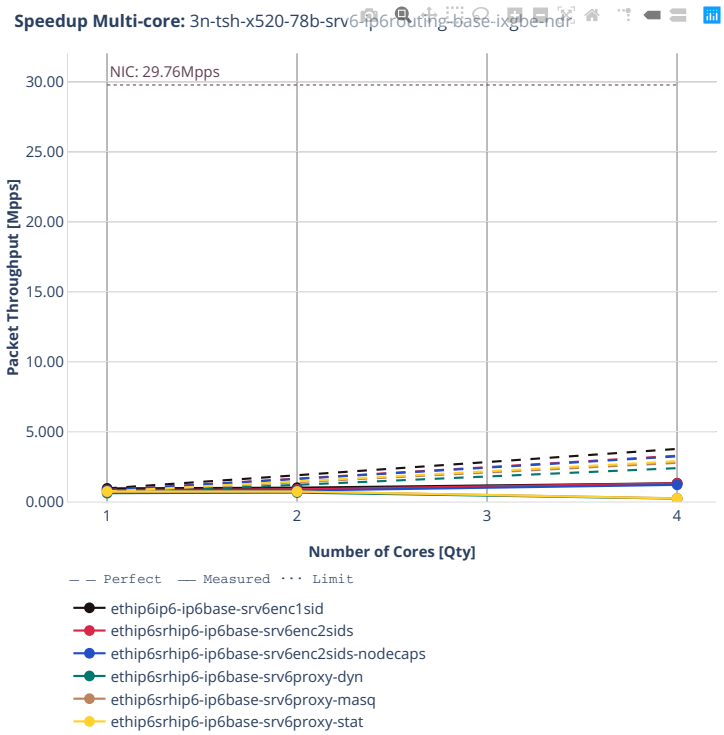
78b-srv6-ip6routing-base-avf

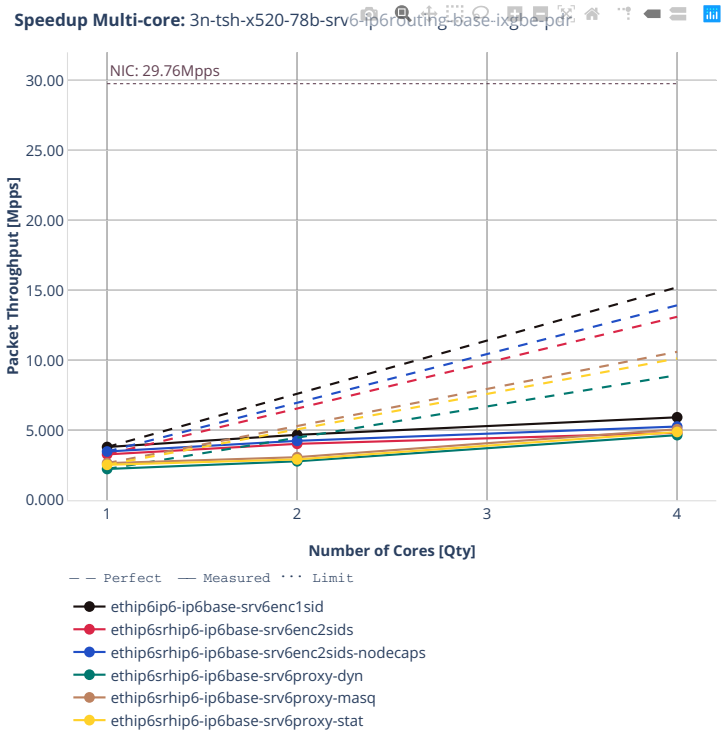



Speedup Multi-core: 3n-icx-xxv710-78b-srv6-ipv6routing-base-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip6ip6-ip6base-srv6enc1sid
avf-ethip6srhip6-ip6base-srv6enc2sids
avf-ethip6srhip6-ip6base-srv6enc2sids-nodecaps
avf-ethip6srhip6-ip6base-srv6proxy-dyn
avf-ethip6srhip6-ip6base-srv6proxy-masq
avf-ethip6srhip6-ip6base-srv6proxy-stat

3n-tsh-x520

78b-srv6-ip6routing-base-ixgbe




Speedup Multi-core: 3n-tsh-x520-78b-srv6-ip6routing-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip6ip6-ip6base-srv6enc1sid
ethip6srhip6-ip6base-srv6enc2sids
ethip6srhip6-ip6base-srv6enc2sids-nodecaps
ethip6srhip6-ip6base-srv6proxy-dyn
ethip6srhip6-ip6base-srv6proxy-masq
ethip6srhip6-ip6base-srv6proxy-stat

2.4.5 IPv4 Tunnels

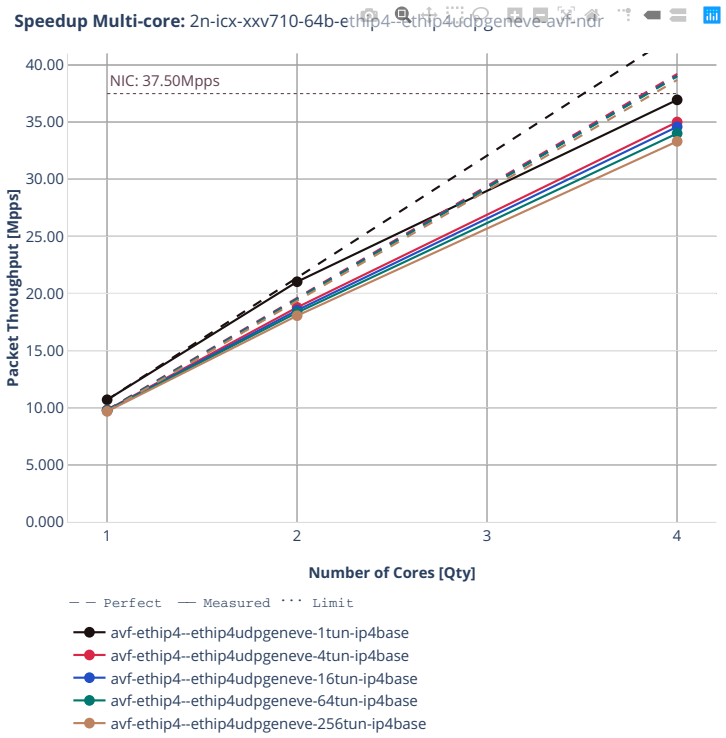
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

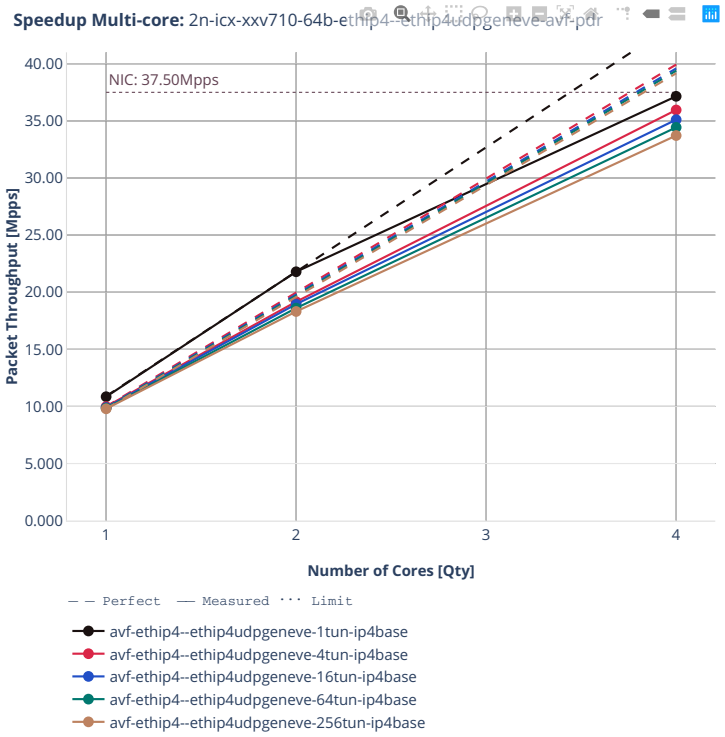
CSIT source code for the test cases used for plots can be found in CSIT git repository[137].

[137] https://git.fd.io/csit/tree/tests/vpp/perf/ip4_tunnels?h=rls2210

2n-icx-xxv710

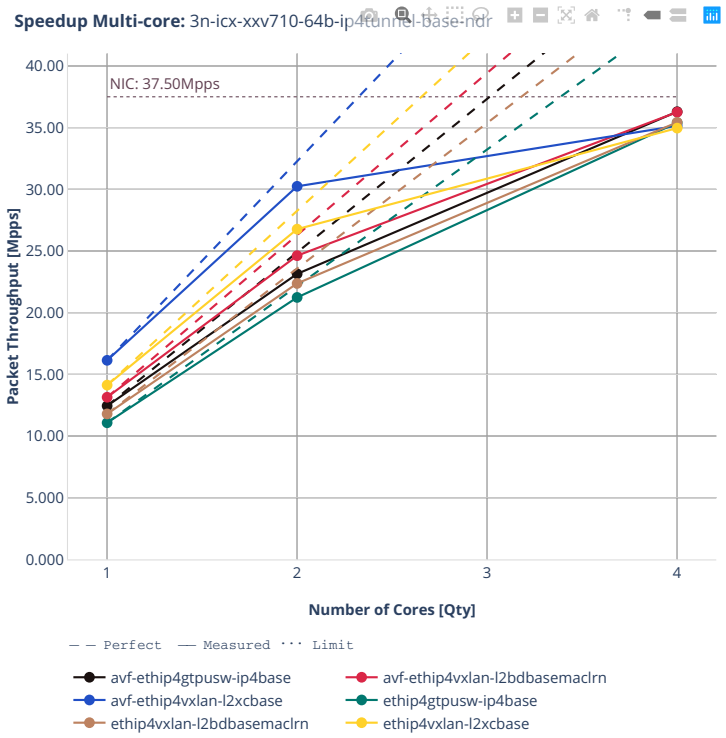
64b-2t1c-ethip4–ethip4udpgeneve-avf

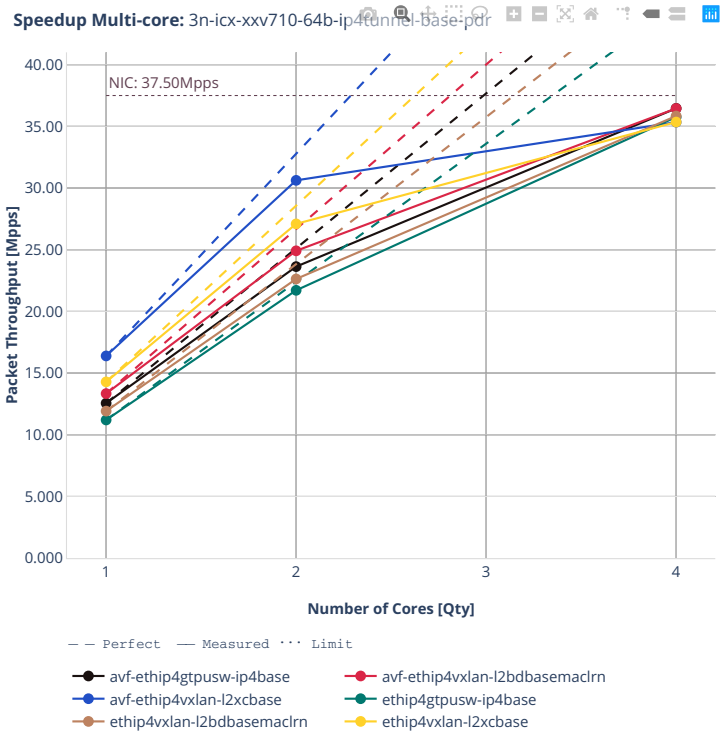



Speedup Multi-core: 2n-icx-xxv710-64b-ethip4-ethip4udpgeneve-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4--ethip4udpgeneve-1tun-ip4base
avf-ethip4--ethip4udpgeneve-4tun-ip4base
avf-ethip4--ethip4udpgeneve-16tun-ip4base
avf-ethip4--ethip4udpgeneve-64tun-ip4base
avf-ethip4--ethip4udpgeneve-256tun-ip4base

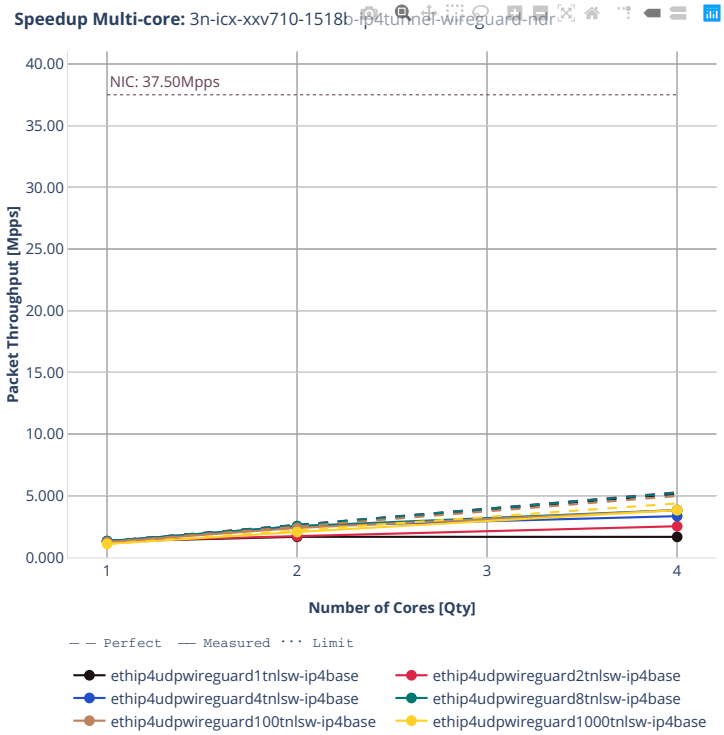
3n-icx-xxv710

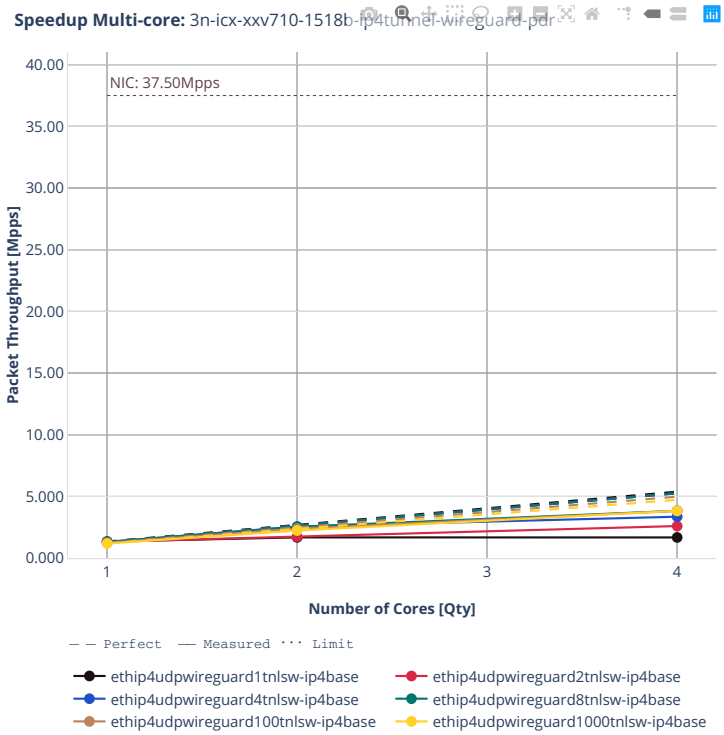
64b-ip4tunnel-base



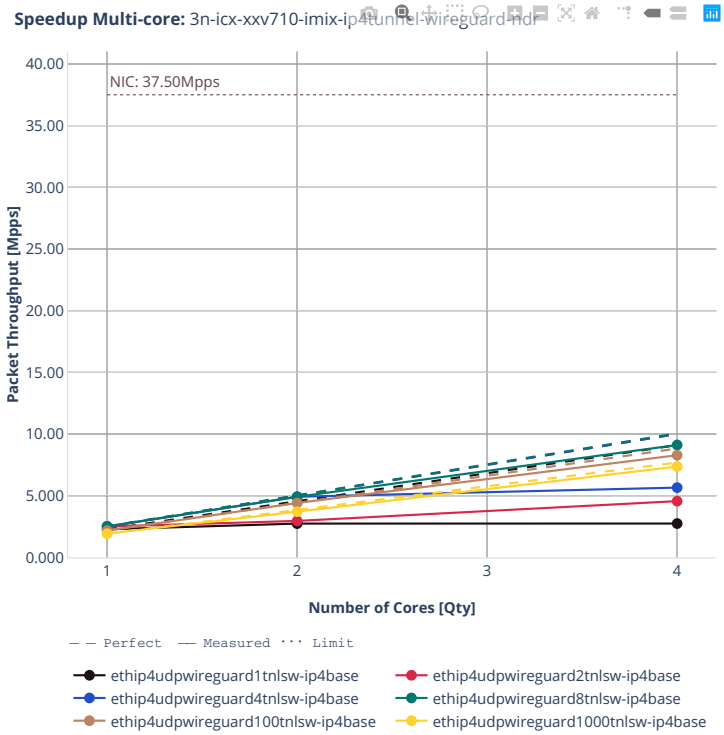

Speedup Multi-core: 3n-icx-xxv710-64b-ip4tunnel-base-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4gtpusw-ip4base
avf-ethip4vxlan-l2bdbasemaclrn
avf-ethip4vxlan-l2xcbase
ethip4gtpusw-ip4base
ethip4vxlan-l2bdbasemaclrn
ethip4vxlan-l2xcbase

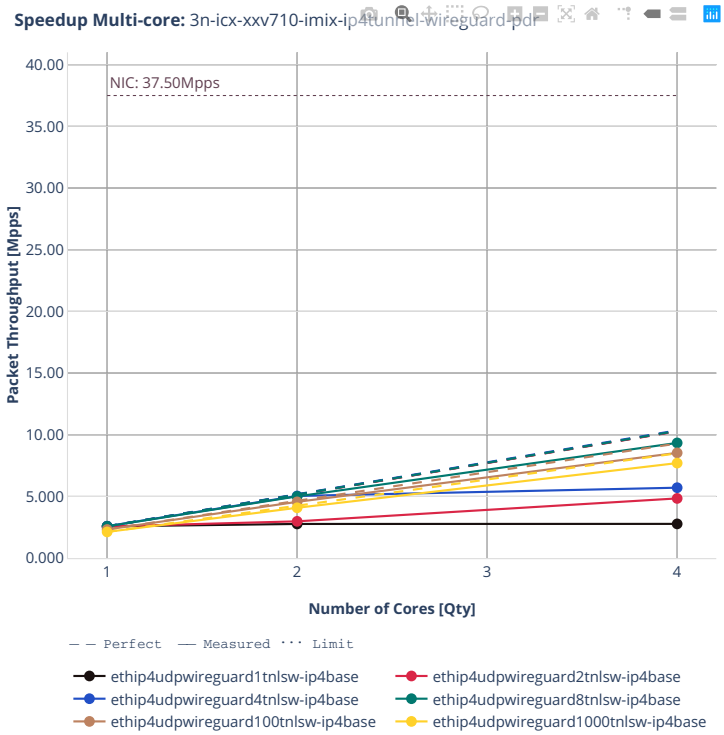
1518b-ip4tunnel-wireguard




Speedup Multi-core: 3n-icx-xxv710-1518b-ip4tunnel-wireguard-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4udpwireguard1tnlsw-ip4base
ethip4udpwireguard2tnlsw-ip4base
ethip4udpwireguard4tnlsw-ip4base
ethip4udpwireguard8tnlsw-ip4base
ethip4udpwireguard100tnlsw-ip4base
ethip4udpwireguard1000tnlsw-ip4base

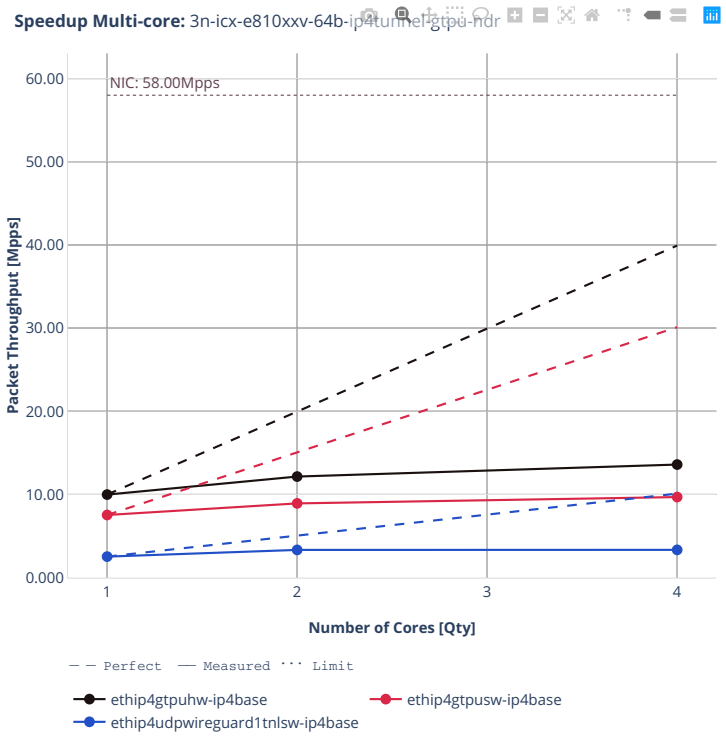
imix-ip4tunnel-wireguard

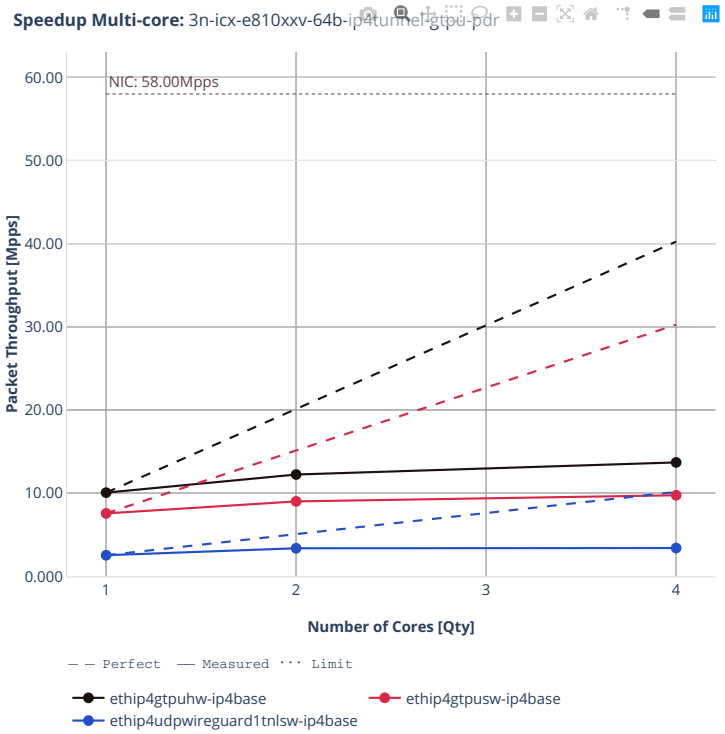



Speedup Multi-core: 3n-icx-xxv710-imix-ip4tunnel-wireguard-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4udpwireguard1tnlsw-ip4base
ethip4udpwireguard2tnlsw-ip4base
ethip4udpwireguard4tnlsw-ip4base
ethip4udpwireguard8tnlsw-ip4base
ethip4udpwireguard100tnlsw-ip4base
ethip4udpwireguard1000tnlsw-ip4base

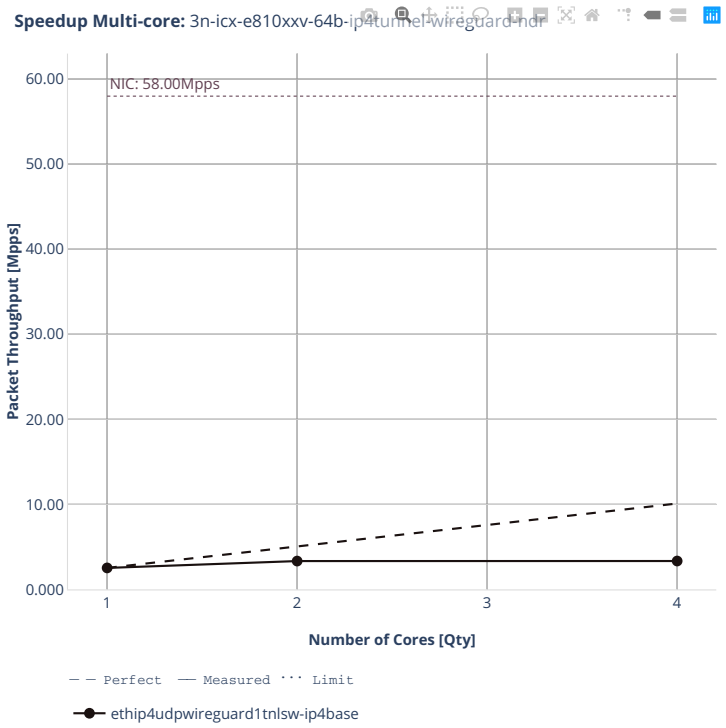
3n-icx-e810xxv

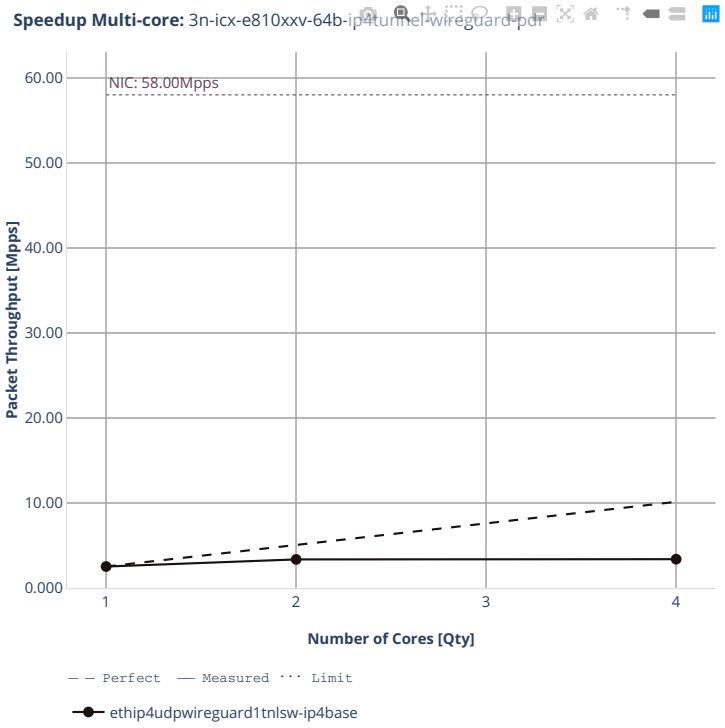
64b-ip4tunnel-gtpu




Speedup Multi-core: 3n-icx-e810xxv-64b-ip4tunnel-gtpu-pdr
NIC: 58.00Mpps
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4gtpuhw-ip4base
ethip4gtpusw-ip4base
ethip4udpwireguard1tnlsw-ip4base

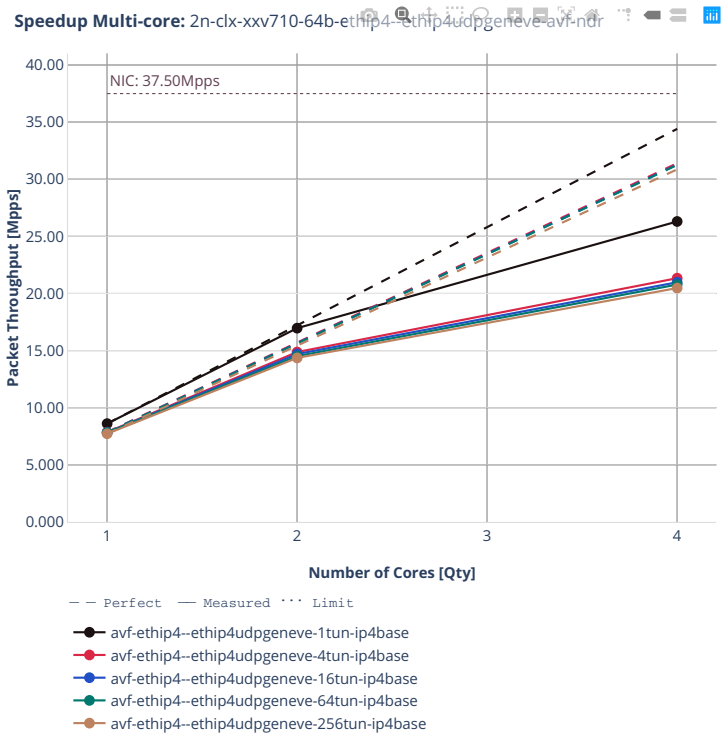
64b-ip4tunnel-wireguard

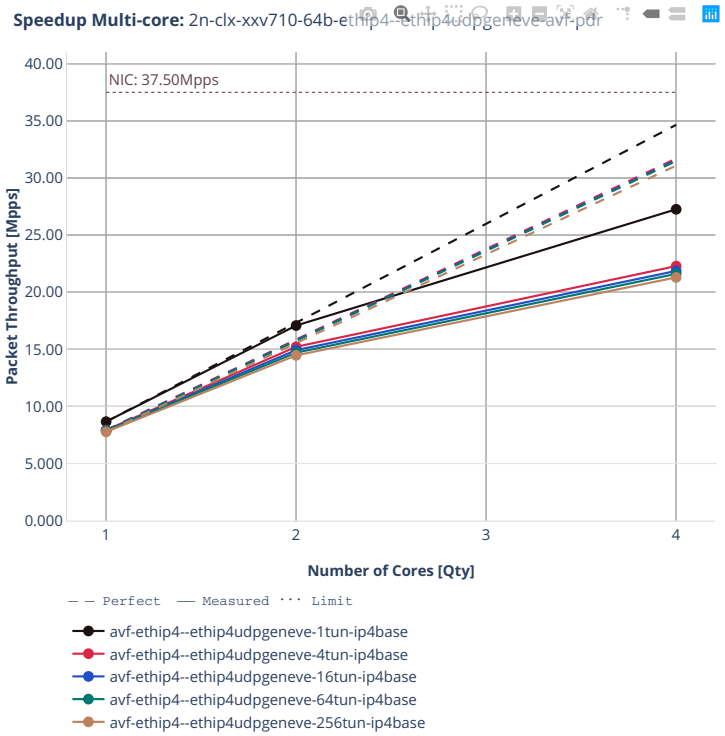



Speedup Multi-core: 3n-icx-e810xxv-64b-ip4tunnel-wireguard-pdr
NIC: 58.00Mpps
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4udpwireguard1tnlsw-ip4base

2n-clx-xxv710

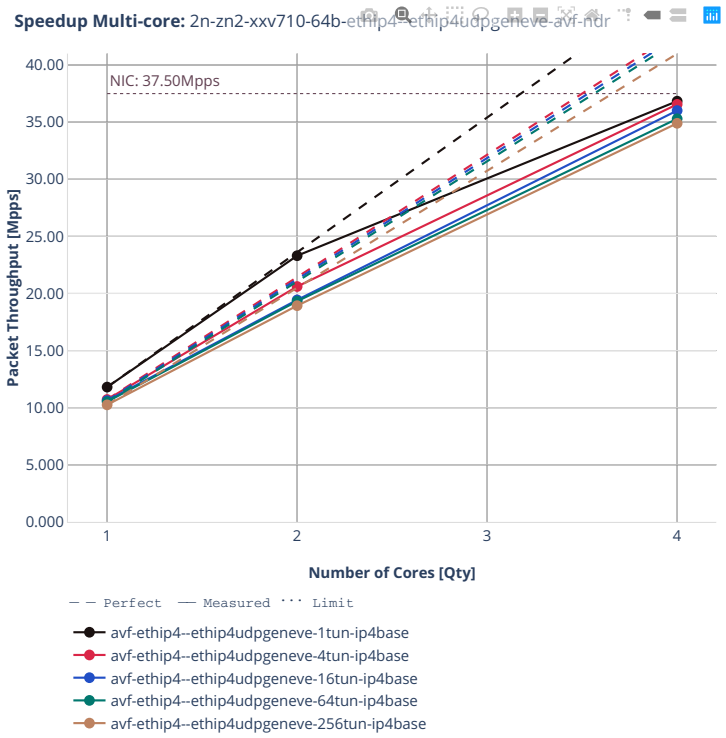
64b-2t1c-ethip4–ethip4udpgeneve-avf

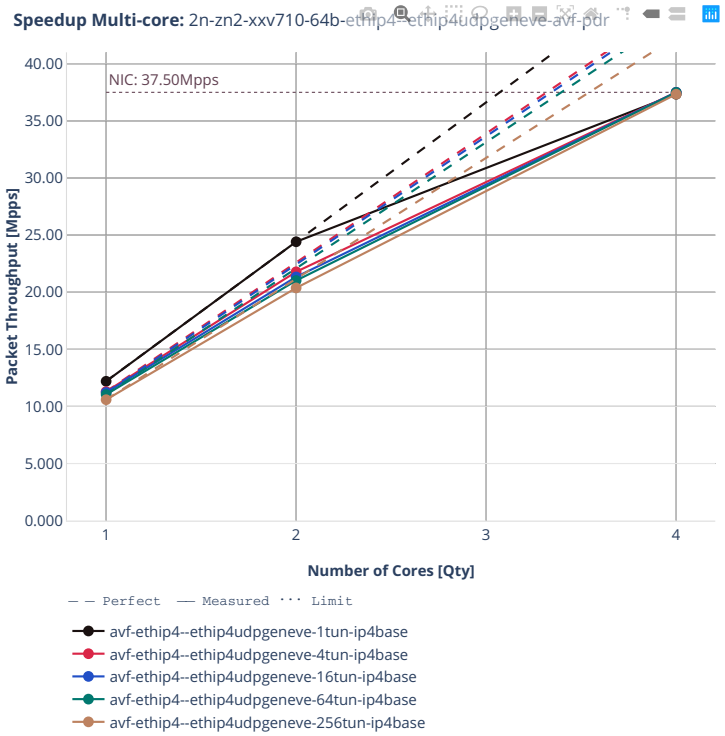



Speedup Multi-core: 2n-clx-xxv710-64b-ethip4-ethip4udpgeneve-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4--ethip4udpgeneve-1tun-ip4base
avf-ethip4--ethip4udpgeneve-4tun-ip4base
avf-ethip4--ethip4udpgeneve-16tun-ip4base
avf-ethip4--ethip4udpgeneve-64tun-ip4base
avf-ethip4--ethip4udpgeneve-256tun-ip4base

2n-zn2-xxv710

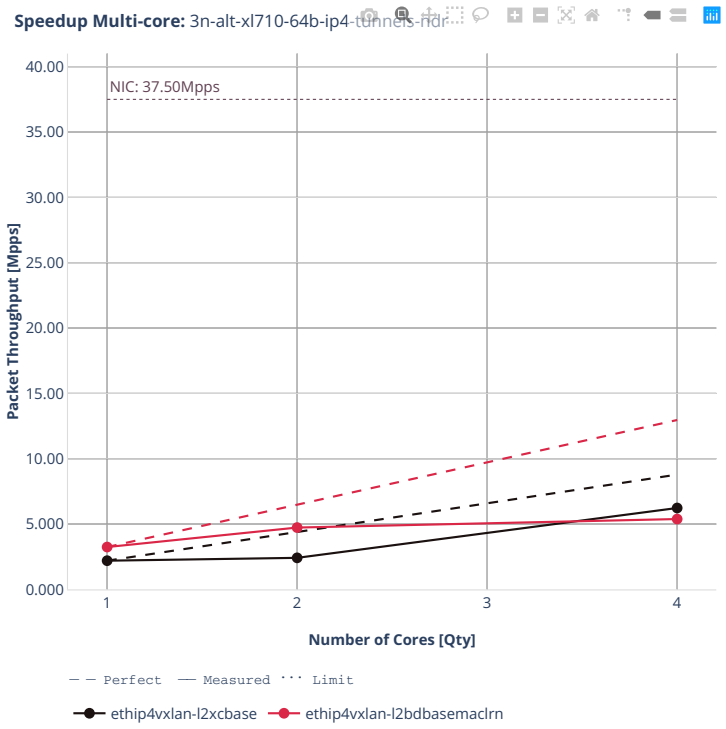
64b-2t1c-ethip4–ethip4udpgeneve-avf

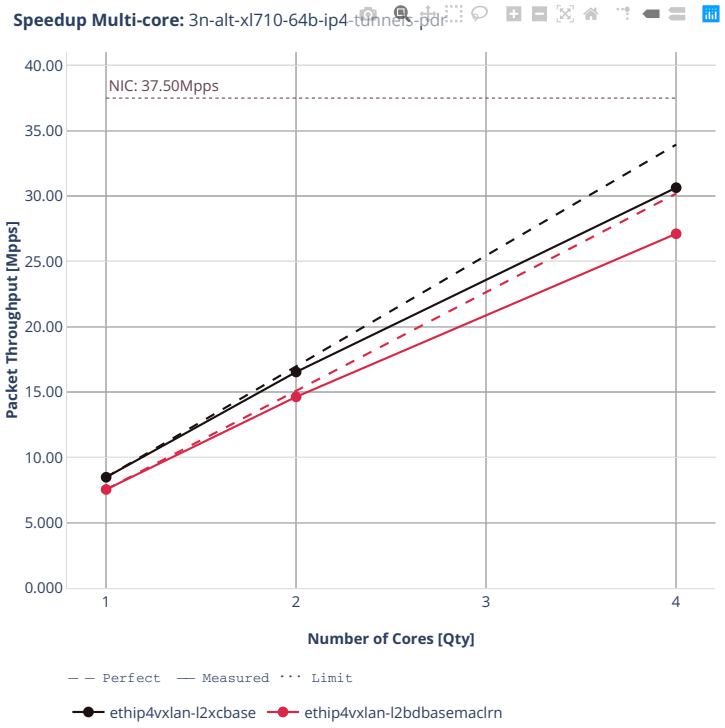



Speedup Multi-core: 2n-zn2-xxv710-64b-ethip4-ethip4udpgeneve-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4--ethip4udpgeneve-1tun-ip4base
avf-ethip4--ethip4udpgeneve-4tun-ip4base
avf-ethip4--ethip4udpgeneve-16tun-ip4base
avf-ethip4--ethip4udpgeneve-64tun-ip4base
avf-ethip4--ethip4udpgeneve-256tun-ip4base

3n-alt-xl710

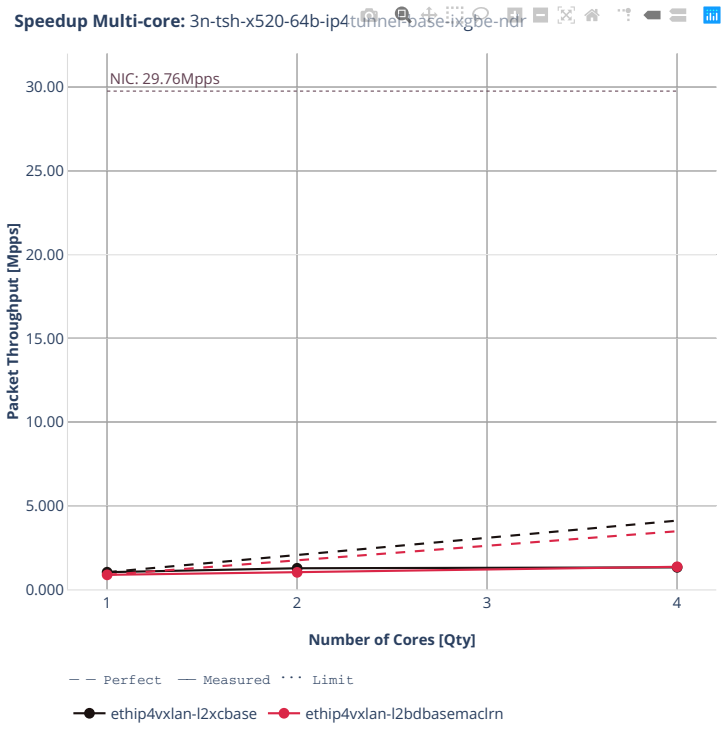
64b-ip4tunnel-base

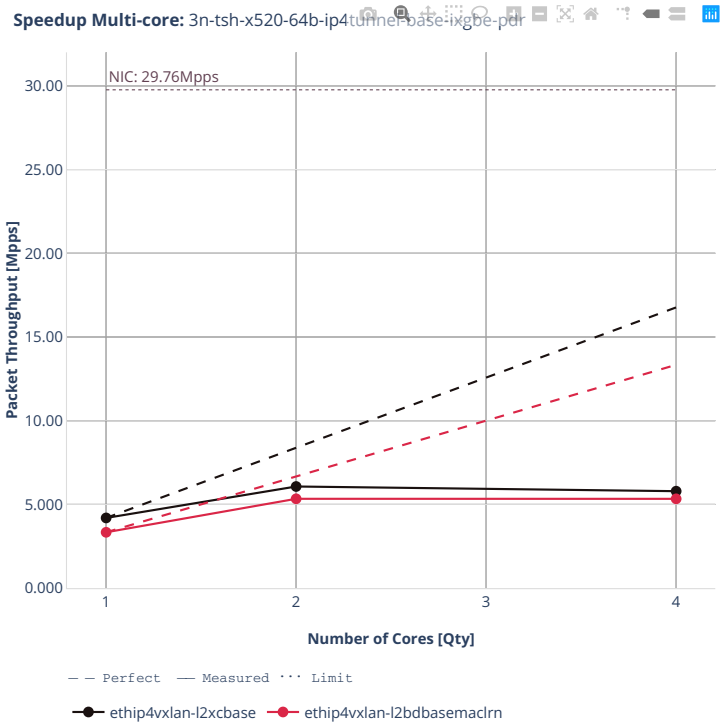



Speedup Multi-core: 3n-alt-xl710-64b-ip4-tunnels-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4vxlan-l2xcbase
ethip4vxlan-l2bdbasemaclrn

3n-tsh-x520

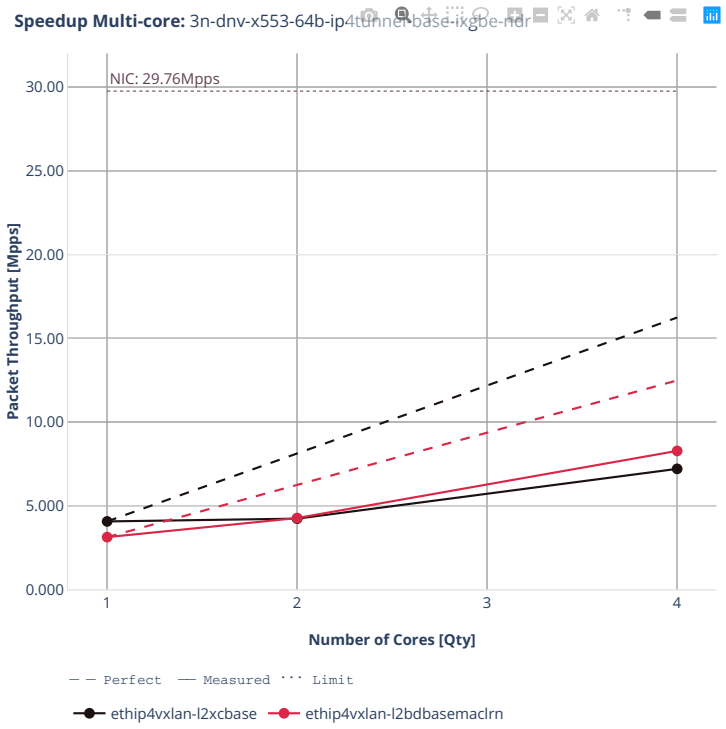
64b-ip4tunnel-base-ixgbe

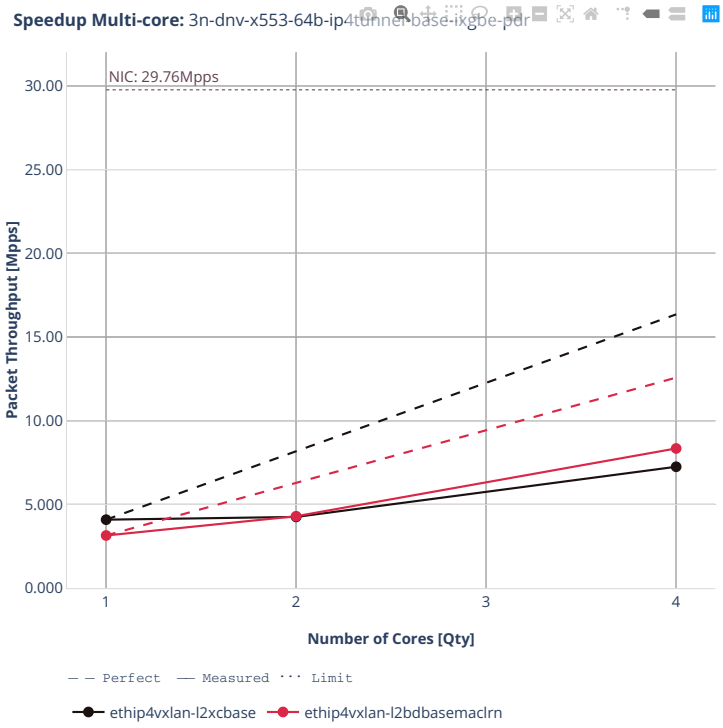



Speedup Multi-core: 3n-tsh-x520-64b-ip4tunnel-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4vxlan-l2xcbase
ethip4vxlan-l2bdbasemaclrn

3n-dnv-x553

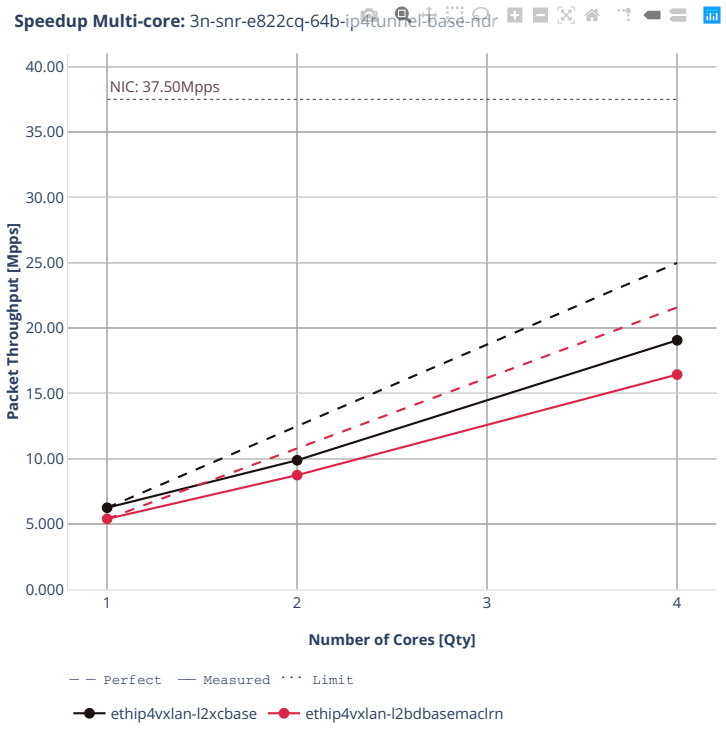
64b-ip4tunnel-base-ixgbe

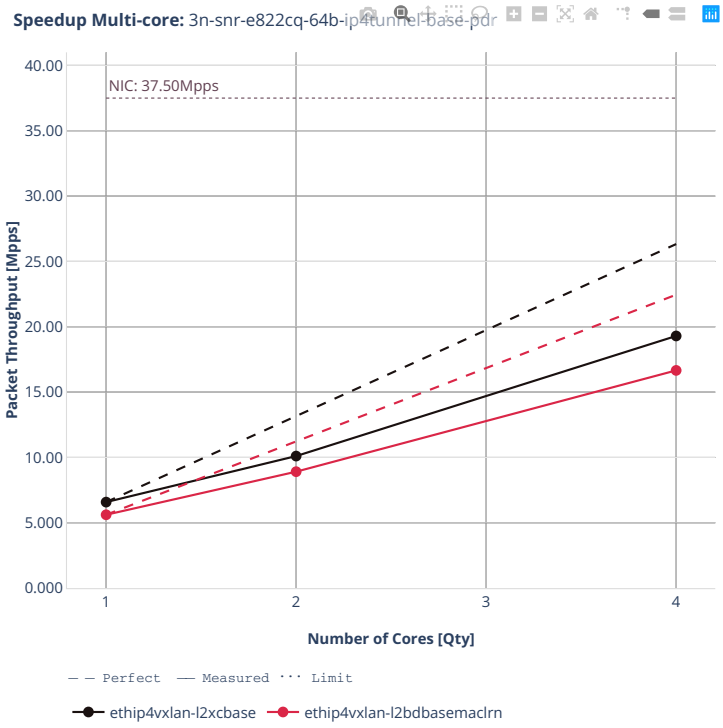



Speedup Multi-core: 3n-dnv-x553-64b-ip4tunnel-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4vxlan-l2xcbase
ethip4vxlan-l2bdbasemaclrn

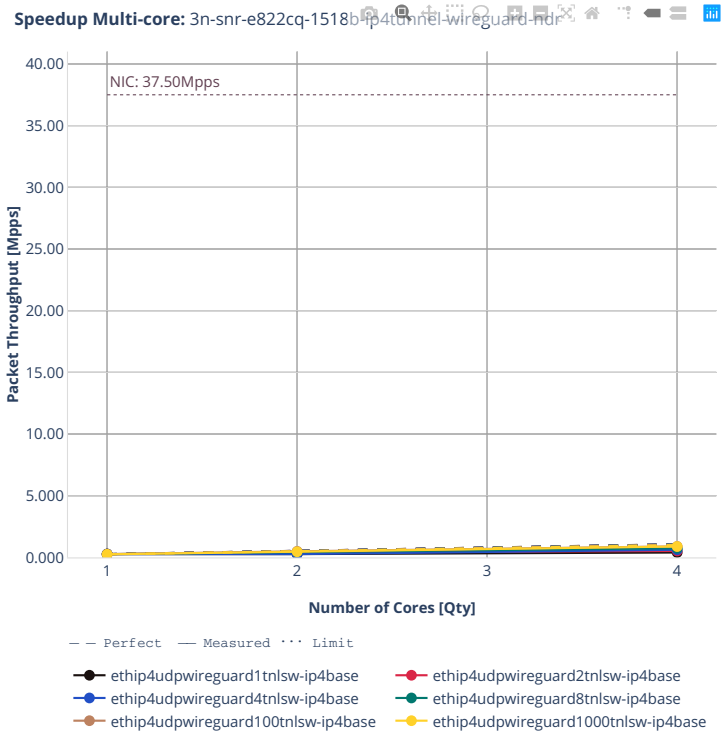
3n-snr-e822cq

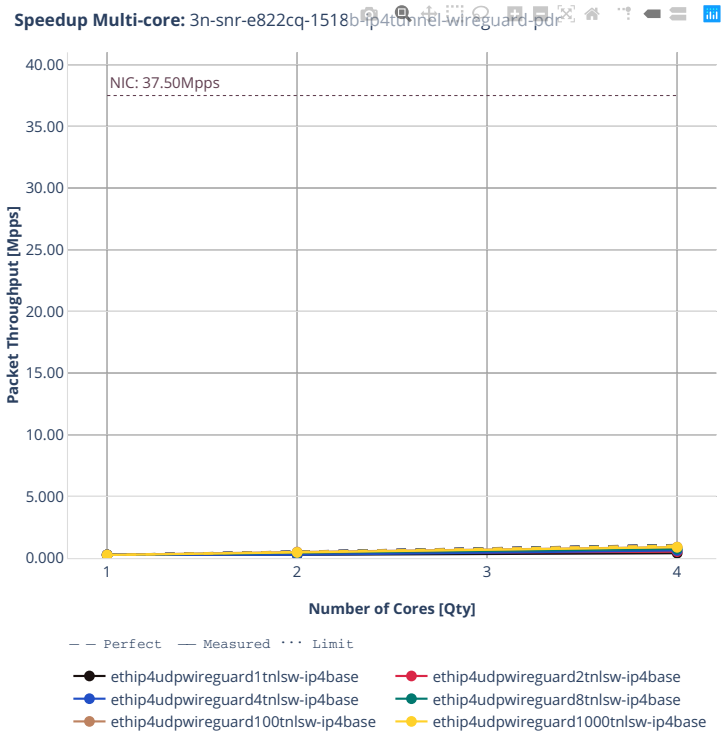
64b-ip4tunnel-base



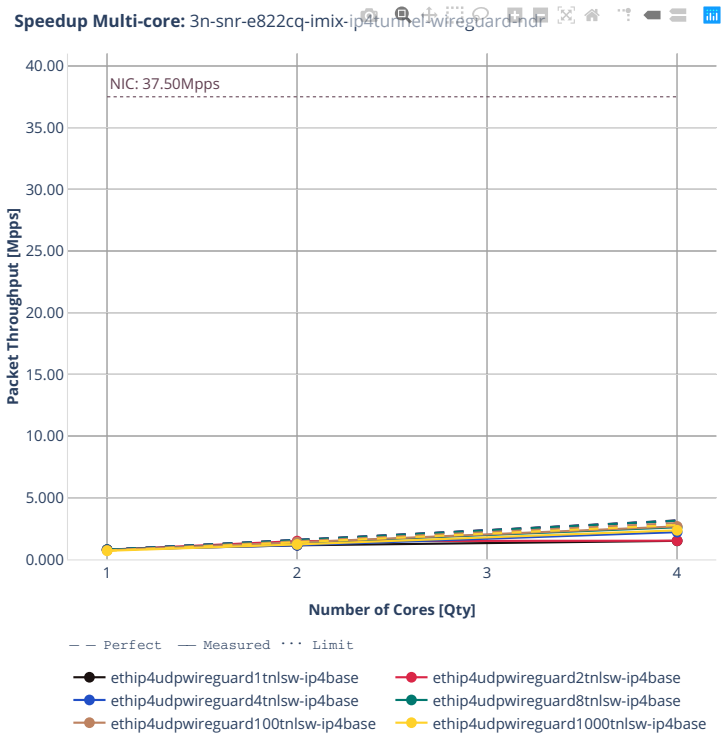

Speedup Multi-core: 3n-snr-e822cq-64b-ip4tunnel-base-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4vxlan-l2xcbase
ethip4vxlan-l2bdbasemaclrn

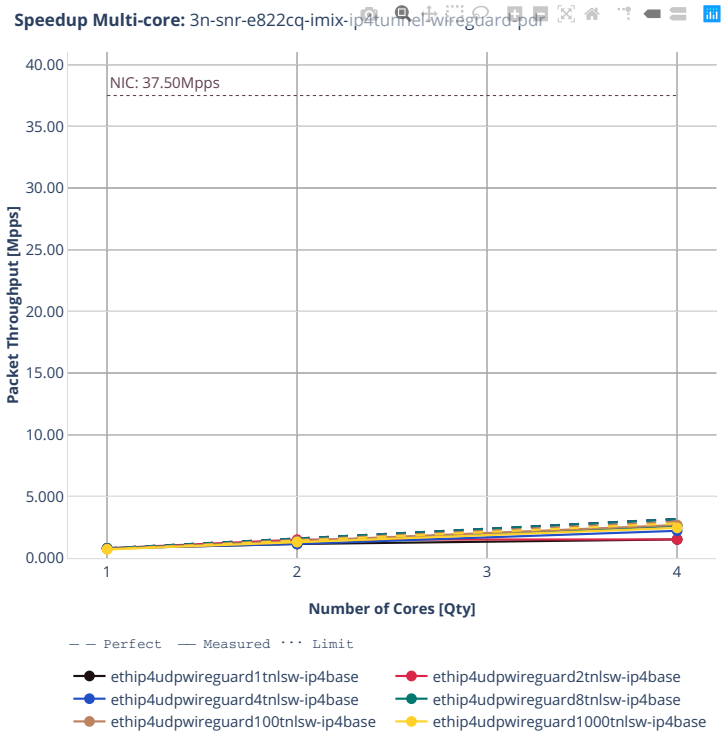
1518b-ip4tunnel-wireguard




Speedup Multi-core: 3n-snr-e822cq-1518b-ip4tunnel-wireguard-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4udpwireguard1tnlsw-ip4base
ethip4udpwireguard2tnlsw-ip4base
ethip4udpwireguard4tnlsw-ip4base
ethip4udpwireguard8tnlsw-ip4base
ethip4udpwireguard100tnlsw-ip4base
ethip4udpwireguard1000tnlsw-ip4base

imix-ip4tunnel-wireguard




Speedup Multi-core: 3n-snr-e822cq-imix-ip4tunnel-wireguard-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4udpwireguard1tnlsw-ip4base
ethip4udpwireguard2tnlsw-ip4base
ethip4udpwireguard4tnlsw-ip4base
ethip4udpwireguard8tnlsw-ip4base
ethip4udpwireguard100tnlsw-ip4base
ethip4udpwireguard1000tnlsw-ip4base

2.4.6 NAT44 IPv4 Routing

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 64B performance tests with VPP IPv4 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

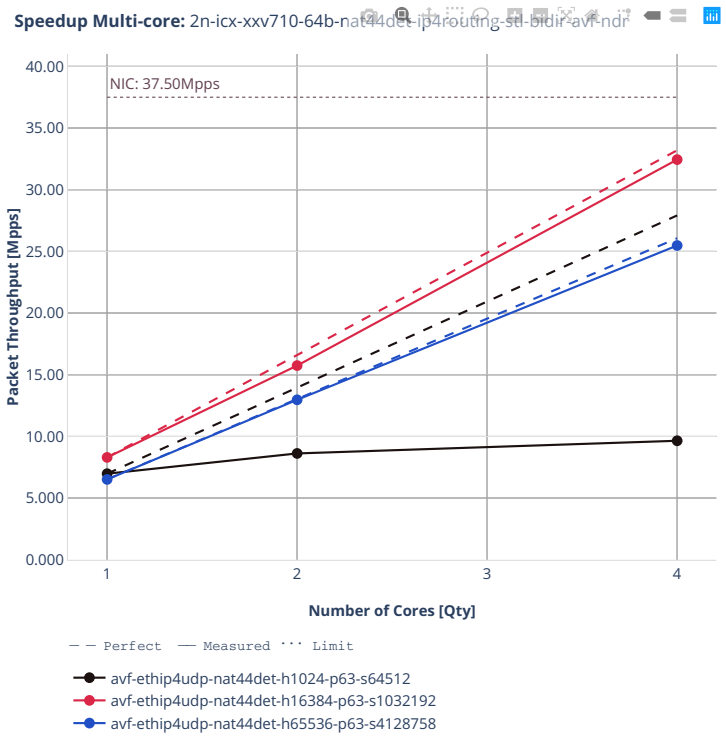
CSIT source code for the test cases used for plots can be found in CSIT git repository[138].

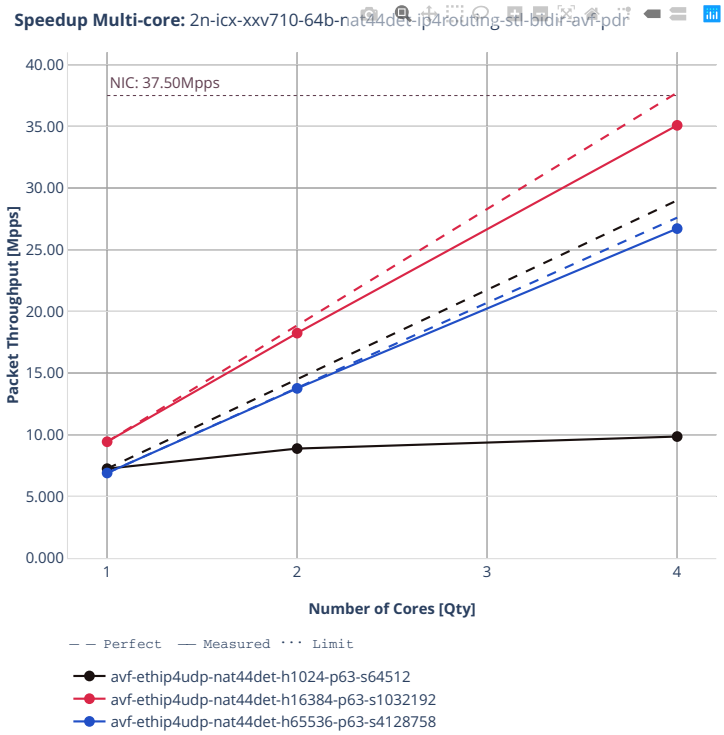
[138] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls2210

Det BiDir

2n-icx-xxv710

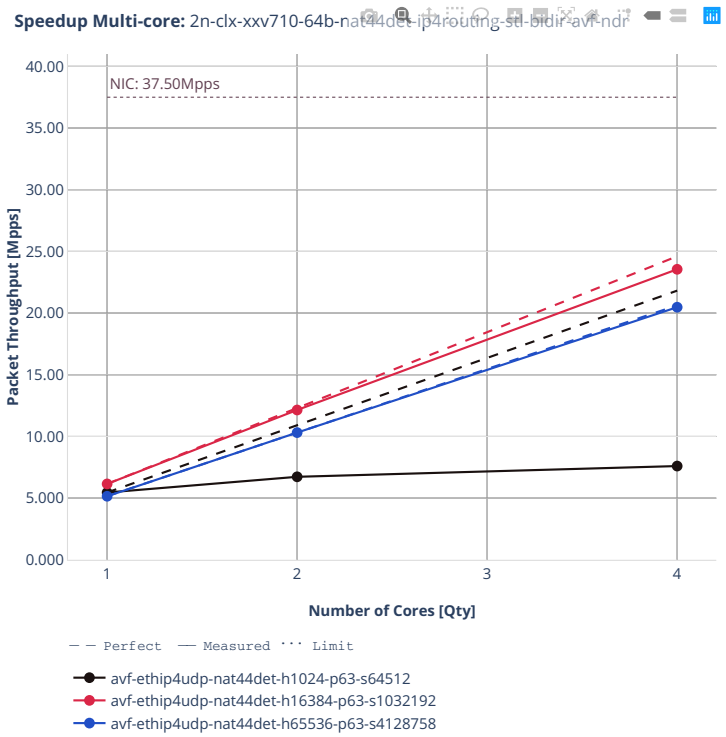
64b-nat44det-ip4routing-stl-bidir-avf

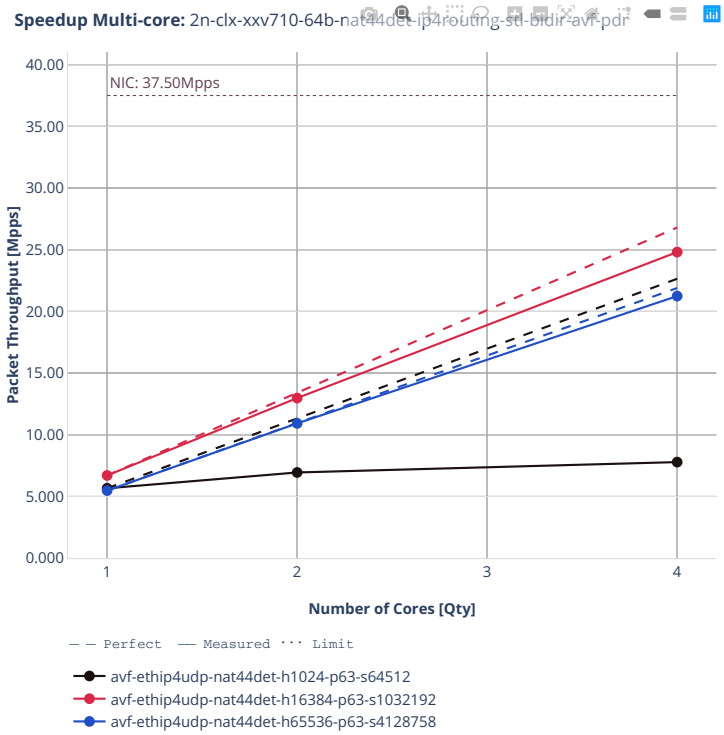



Speedup Multi-core: 2n-icx-xxv710-64b-nat44det-ip4routing-std-bidir-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-nat44det-h1024-p63-s64512
avf-ethip4udp-nat44det-h16384-p63-s1032192
avf-ethip4udp-nat44det-h65536-p63-s4128758

2n-clx-xxv710

64b-nat44det-ip4routing-stl-bidir-avf

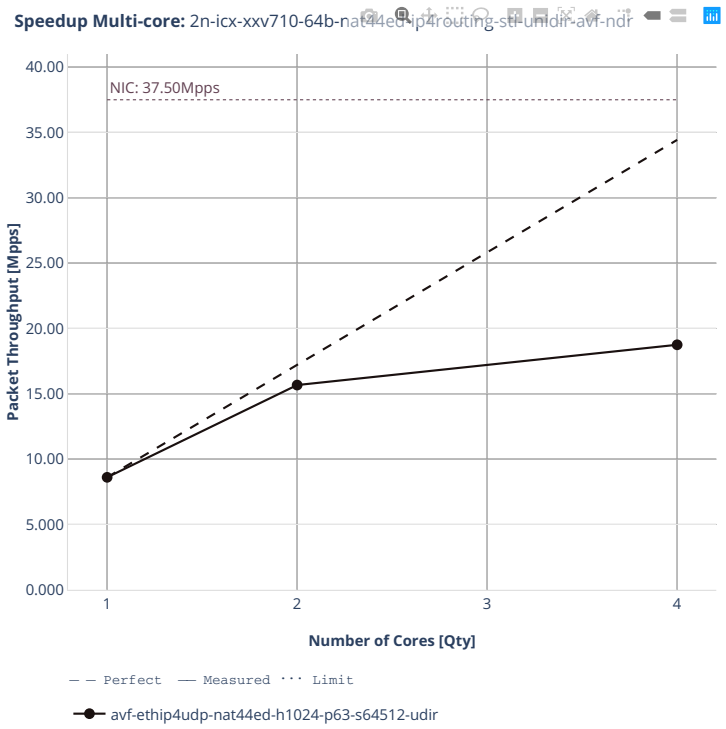


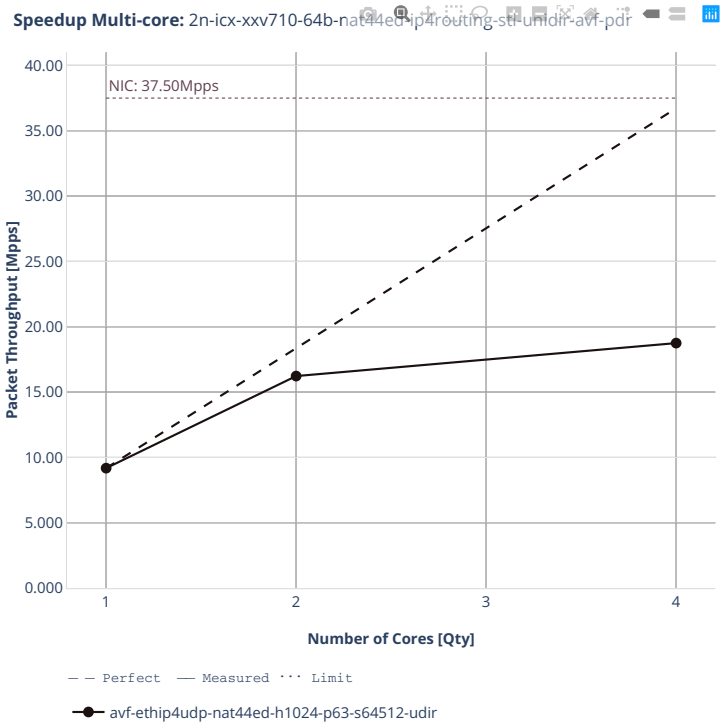

Speedup Multi-core: 2n-clx-xxv710-64b-nat44det-ip4routing-std-bidir-avf-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-nat44det-h1024-p63-s64512
avf-ethip4udp-nat44det-h16384-p63-s1032192
avf-ethip4udp-nat44det-h65536-p63-s4128758

ED UniDir

2n-icx-xxv710

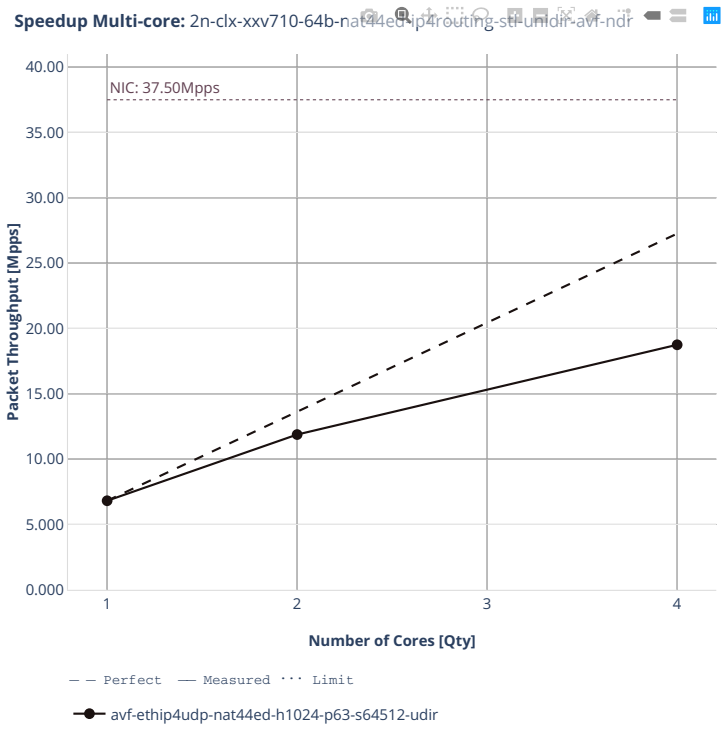
64b-nat44ed-ip4routing-stl-unidir-avf

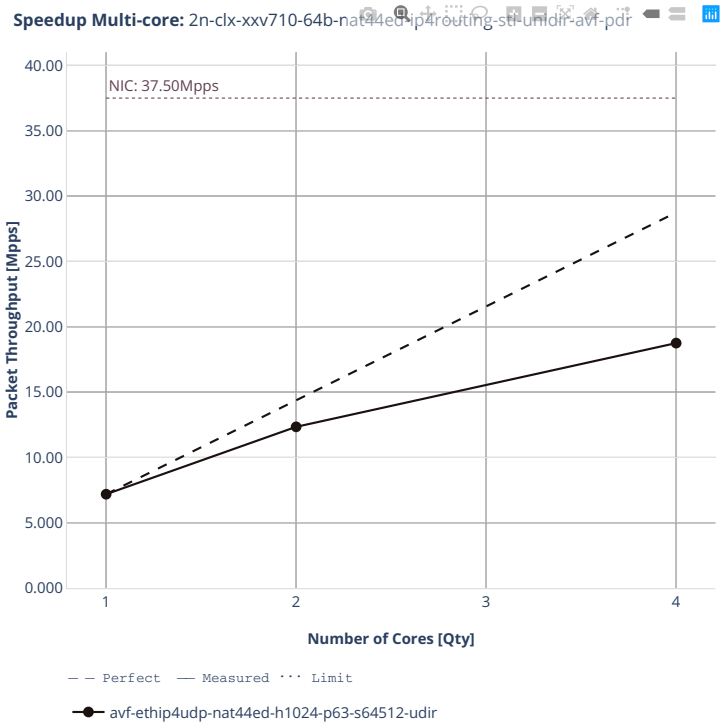



Speedup Multi-core: 2n-icx-xxv710-64b-nat44ed-ip4routing-std-udir-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-nat44ed-h1024-p63-s64512-udir

2n-clx-xxv710

64b-nat44ed-ip4routing-stl-unidir-avf

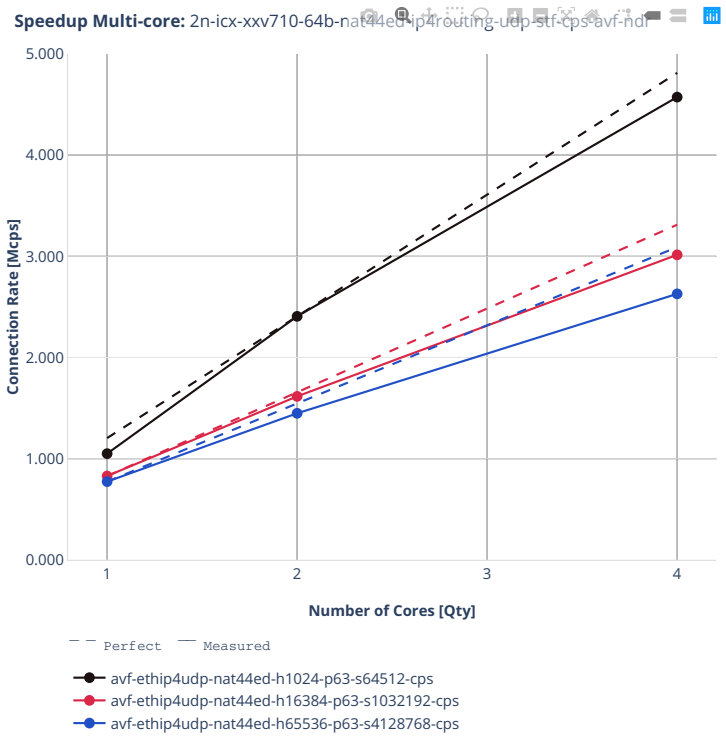


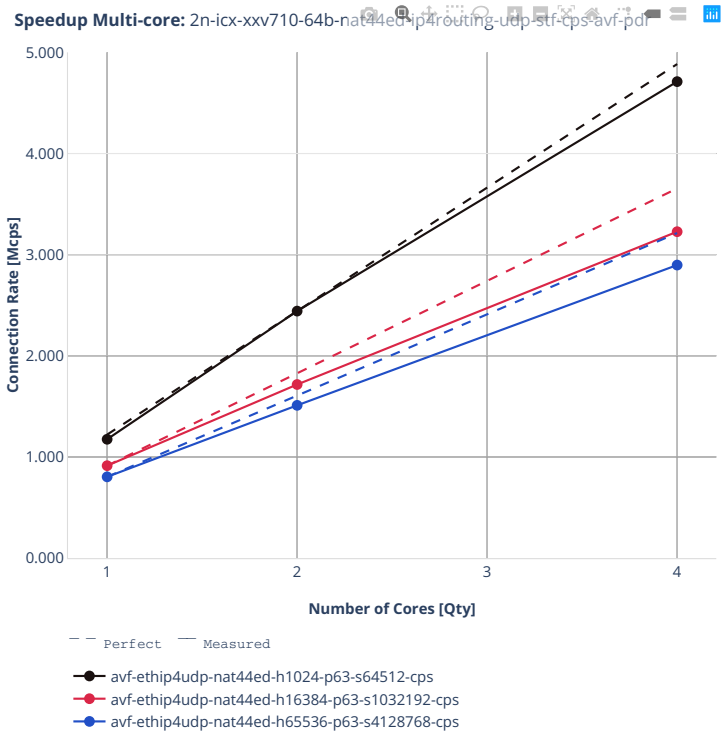

Speedup Multi-core: 2n-clx-xxv710-64b-nat44ed-ip4routing-std-undir-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-nat44ed-h1024-p63-s64512-udir

ED UDP CPS

2n-icx-xxv710

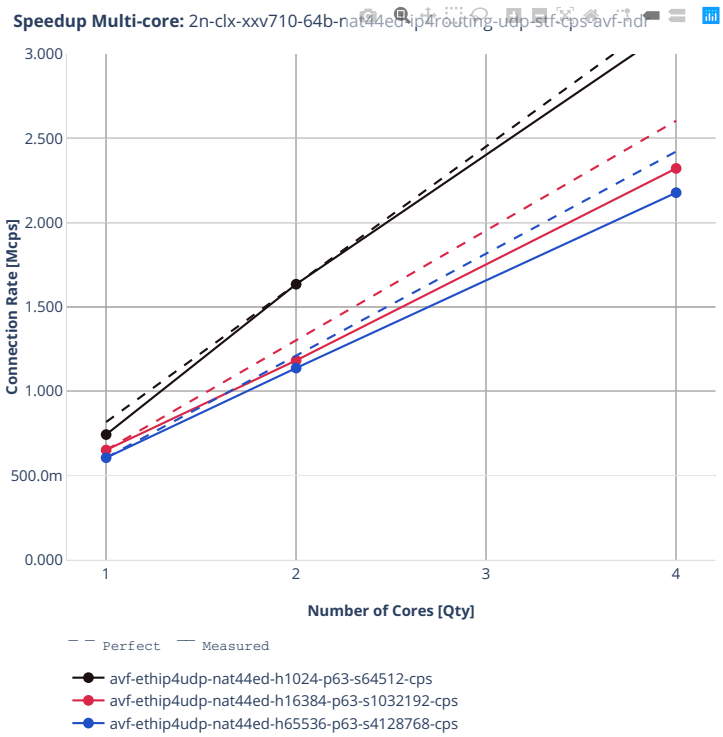
64b-nat44ed-ip4routing-udp-stf-cps-avf

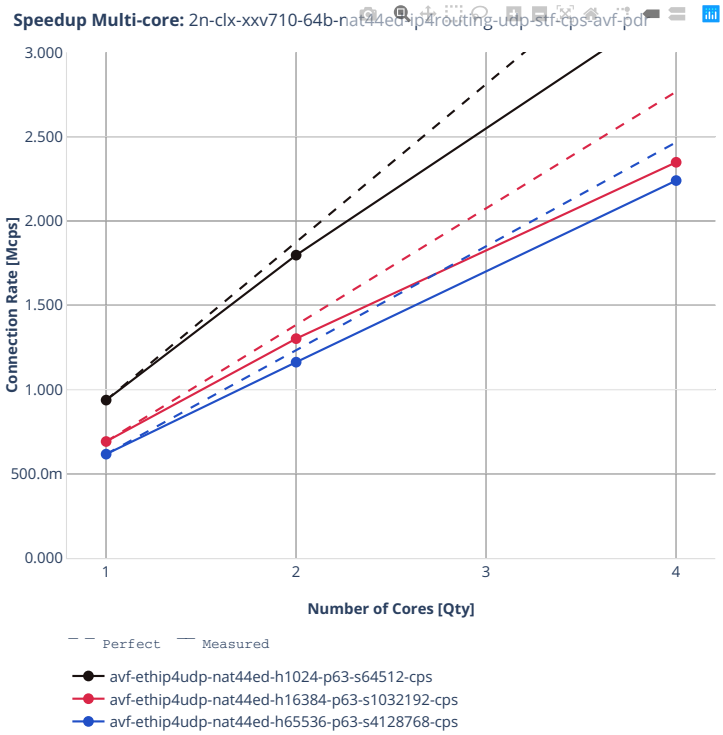



Speedup Multi-core: 2n-icx-xxv710-64b-nat44ed-ip4routing-udp-stf-cps-avf-pdr
Connection Rate [Mcps]
5.000
4.000
3.000
2.000
1.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
avf-ethip4udp-nat44ed-h1024-p63-s64512-cps
avf-ethip4udp-nat44ed-h16384-p63-s1032192-cps
avf-ethip4udp-nat44ed-h65536-p63-s4128768-cps

2n-clx-xxv710

64b-nat44ed-ip4routing-udp-stf-cps-avf

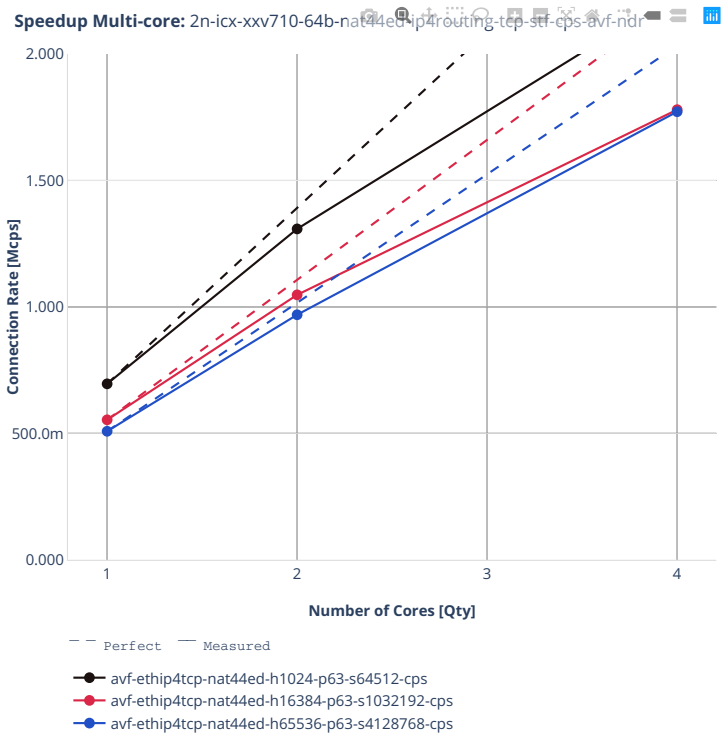


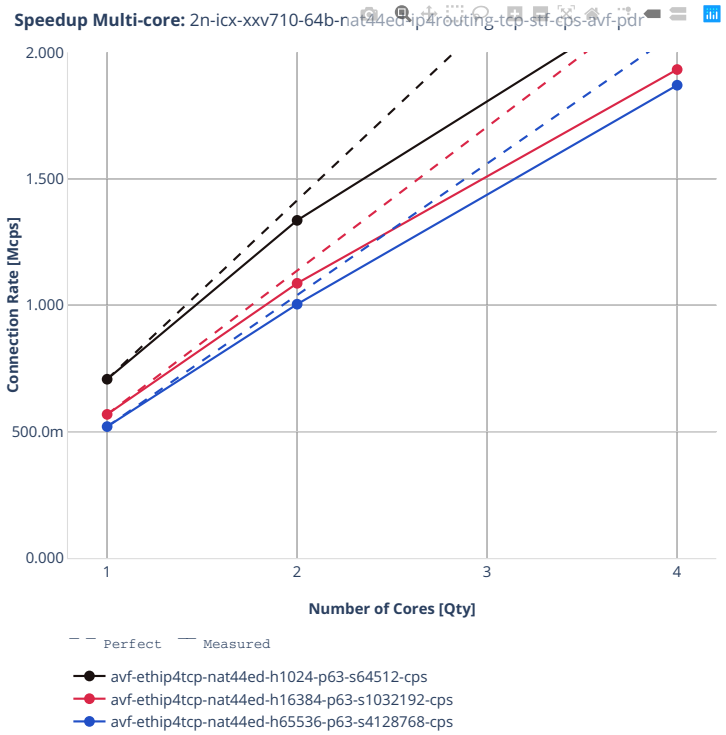

Speedup Multi-core: 2n-clx-xxv710-64b-nat44ed-ip4routing-udp-stf-cps-avf-pdr
3.000
2.500
2.000
1.500
1.000
500.0m
0.000
Connection Rate [Mcps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
avf-ethip4udp-nat44ed-h1024-p63-s64512-cps
avf-ethip4udp-nat44ed-h16384-p63-s1032192-cps
avf-ethip4udp-nat44ed-h65536-p63-s4128768-cps

ED TCP CPS

2n-icx-xxv710

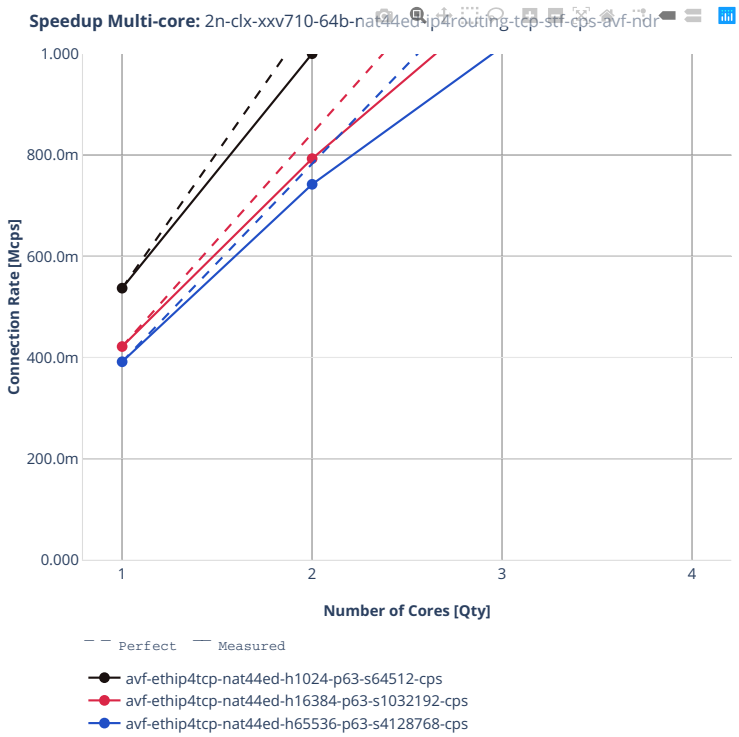
64b-nat44ed-ip4routing-tcp-stf-cps-avf

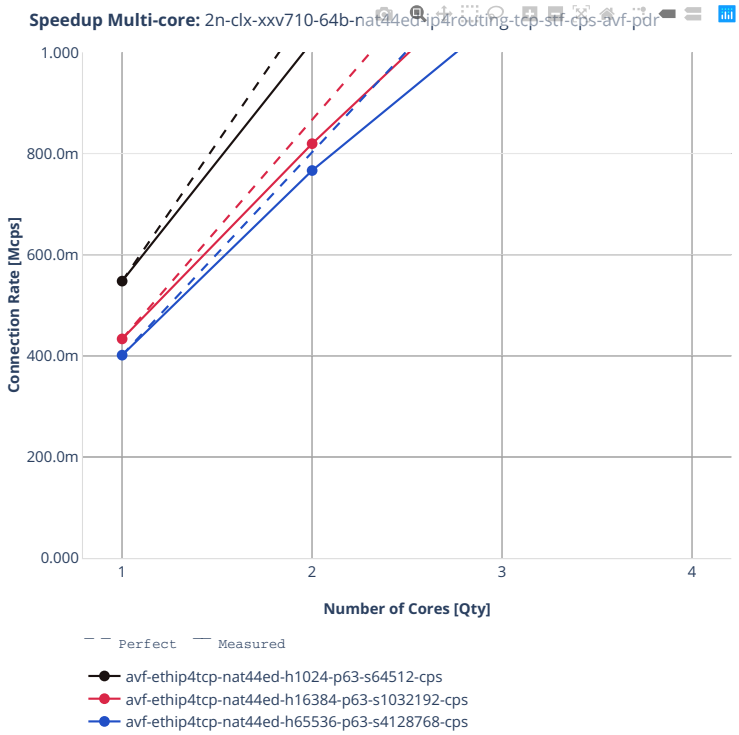



Speedup Multi-core: 2n-icx-xxv710-64b-nat44ed-ip4routing-tcp-stf-cps-avf-pdr
2.000
1.500
1.000
500.0m
0.000
Connection Rate [Mcps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
avf-ethip4tcp-nat44ed-h1024-p63-s64512-cps
avf-ethip4tcp-nat44ed-h16384-p63-s1032192-cps
avf-ethip4tcp-nat44ed-h65536-p63-s4128768-cps

2n-clx-xxv710

64b-nat44ed-ip4routing-tcp-stf-cps-avf

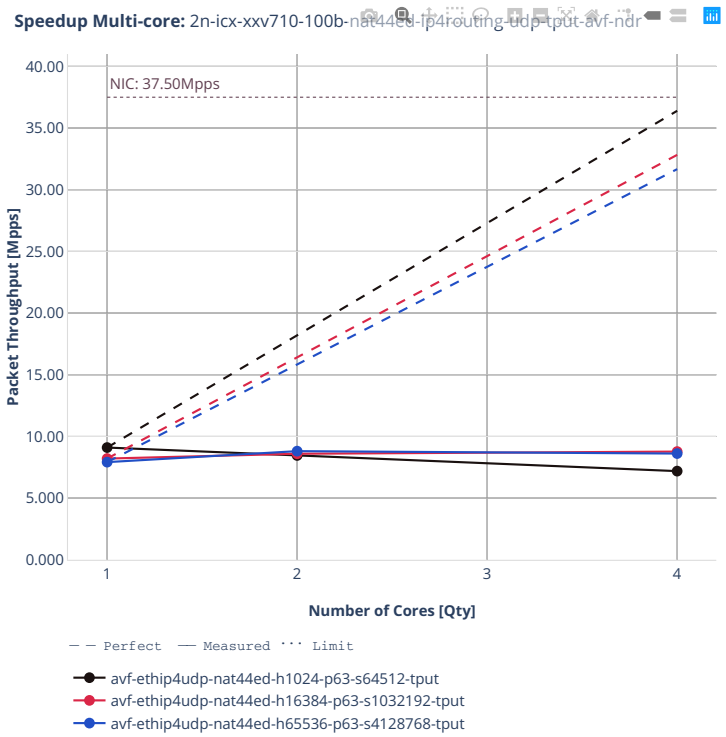


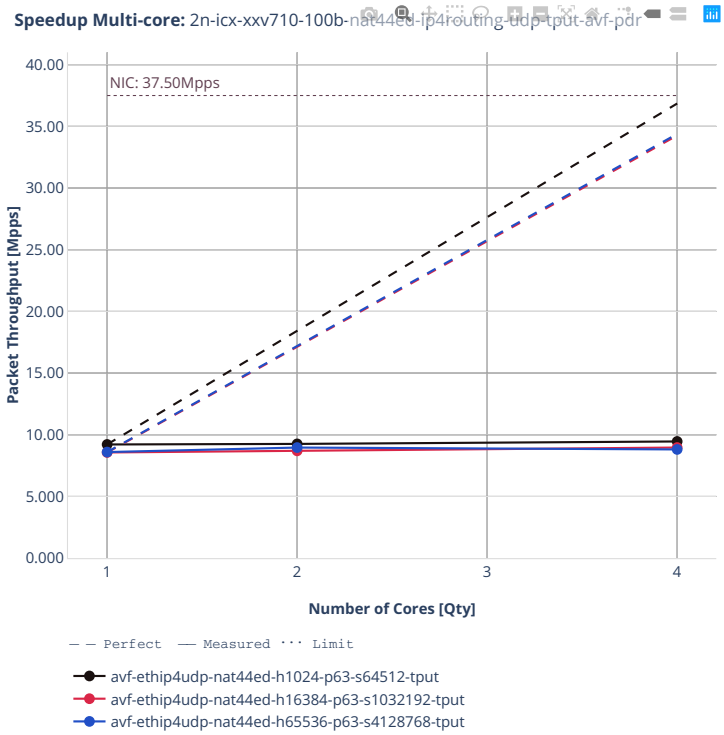

Speedup Multi-core: 2n-clx-xxv710-64b-nat44ed-ip4routing-tcp-stf-cps-avf-pdr
1.000
800.0m
600.0m
400.0m
200.0m
0.000
Connection Rate [Mcps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
avf-ethip4tcp-nat44ed-h1024-p63-s64512-cps
avf-ethip4tcp-nat44ed-h16384-p63-s1032192-cps
avf-ethip4tcp-nat44ed-h65536-p63-s4128768-cps

ED UDP TPUT

2n-icx-xxv710

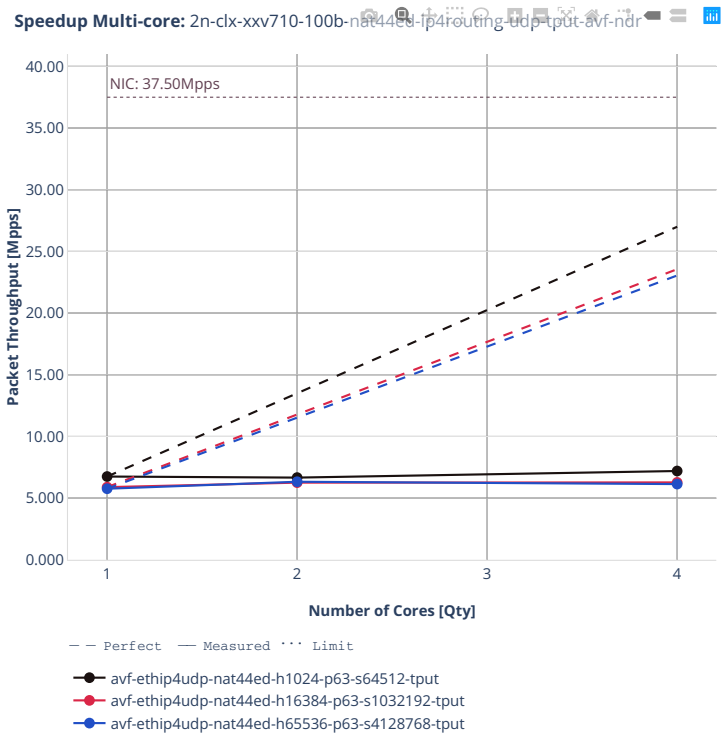
100b-nat44ed-ip4routing-udp-tput-avf

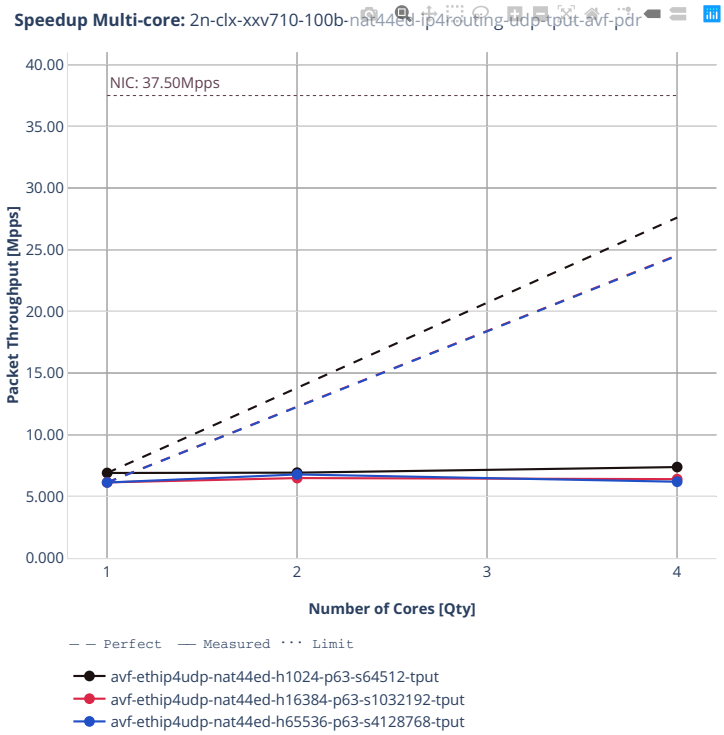



Speedup Multi-core: 2n-icx-xxv710-100b-nat44ed-ip4routing-udp-tput-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-nat44ed-h1024-p63-s64512-tput
avf-ethip4udp-nat44ed-h16384-p63-s1032192-tput
avf-ethip4udp-nat44ed-h65536-p63-s4128768-tput

2n-clx-xxv710

100b-nat44ed-ip4routing-udp-tput-avf

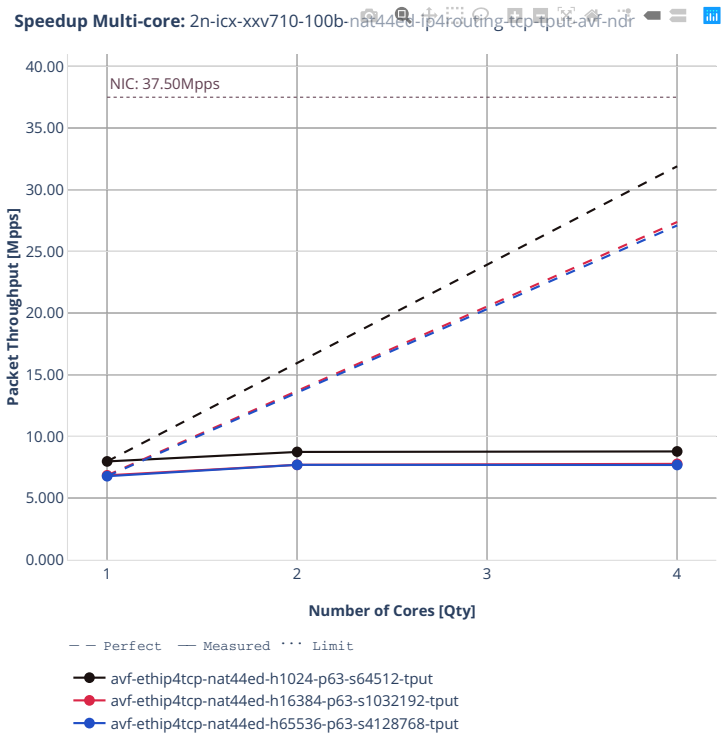


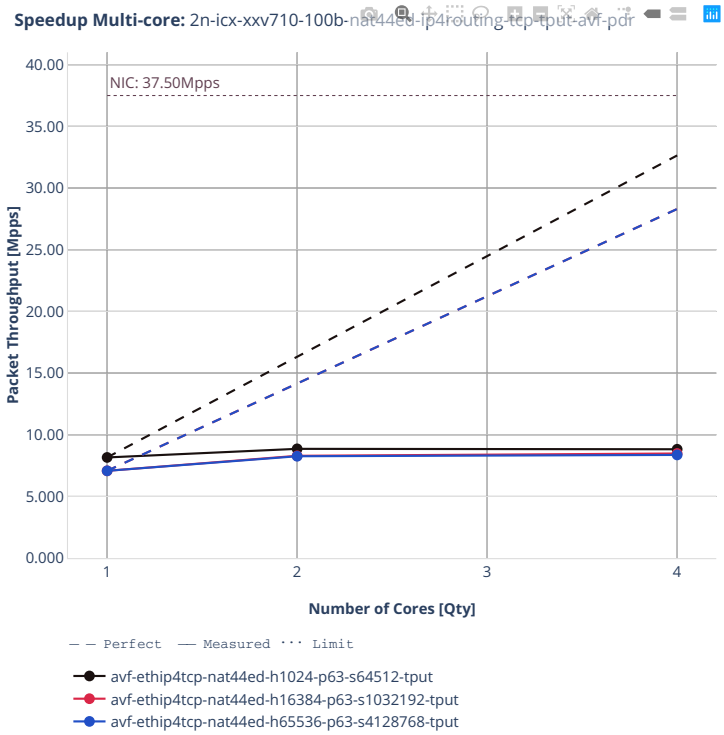

Speedup Multi-core: 2n-clx-xxv710-100b-nat44ed-ip4routing-udp-tput-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4udp-nat44ed-h1024-p63-s64512-tput
avf-ethip4udp-nat44ed-h16384-p63-s1032192-tput
avf-ethip4udp-nat44ed-h65536-p63-s4128768-tput

ED TCP TPUT

2n-icx-xxv710

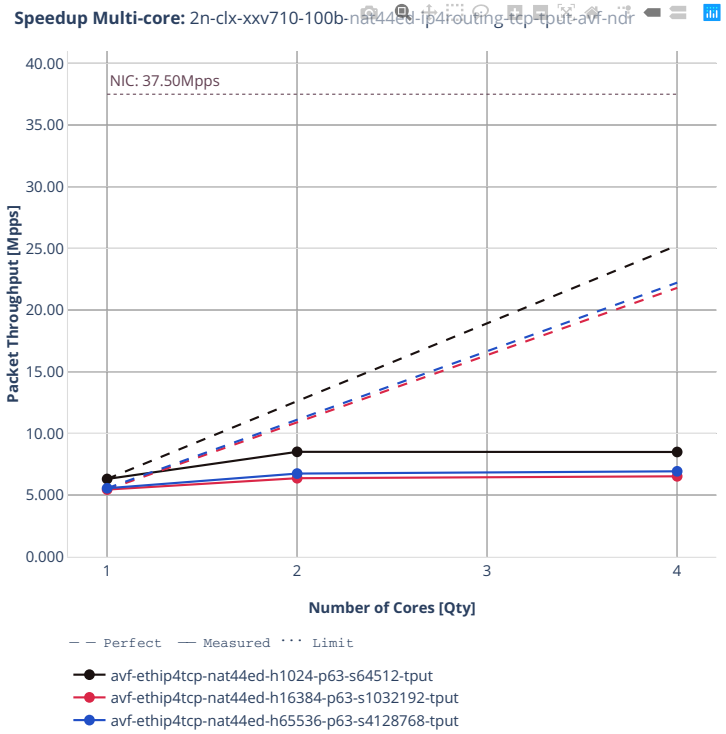
100b-nat44ed-ip4routing-tcp-tput-avf

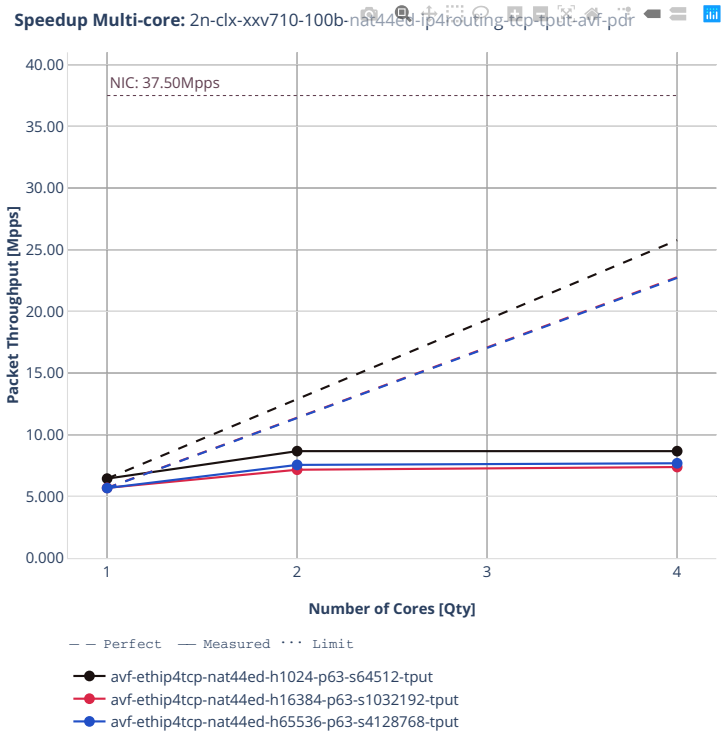



Speedup Multi-core: 2n-icx-xxv710-100b-nat44ed-ip4routing-tcp-tput-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4tcp-nat44ed-h1024-p63-s64512-tput
avf-ethip4tcp-nat44ed-h16384-p63-s1032192-tput
avf-ethip4tcp-nat44ed-h65536-p63-s4128768-tput

2n-clx-xxv710

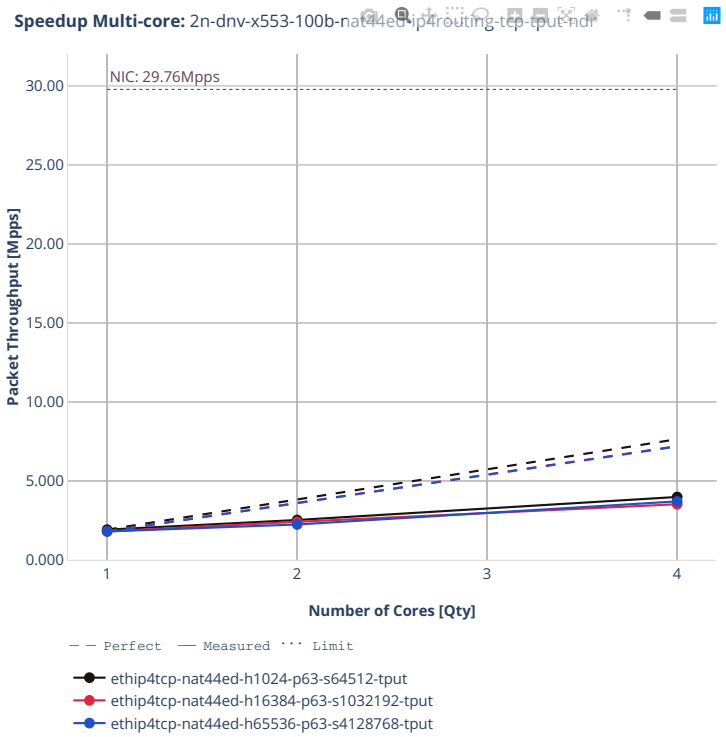
100b-nat44ed-ip4routing-tcp-tput-avf

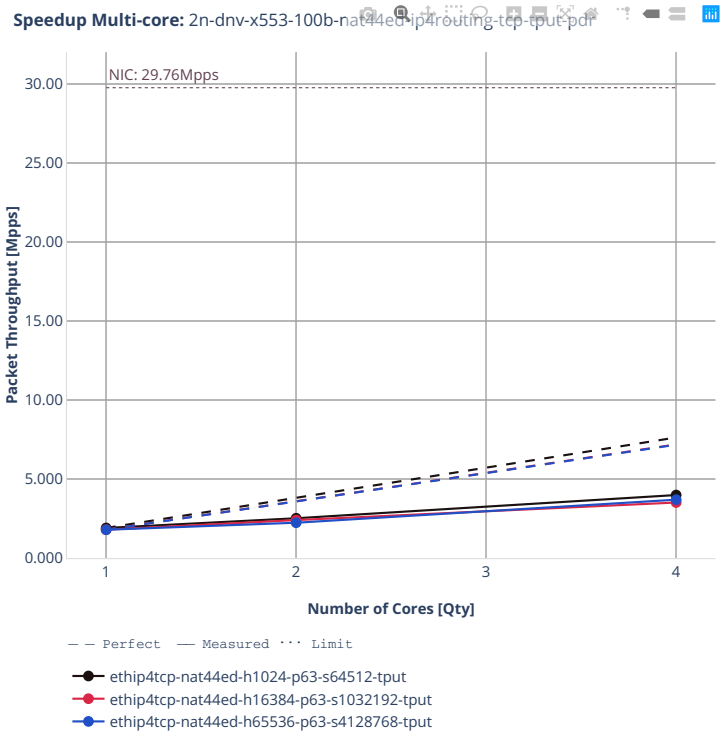



Speedup Multi-core: 2n-clx-xxv710-100b-nat44ed-ip4routing-tcp-tput-avf-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4tcp-nat44ed-h1024-p63-s64512-tput
avf-ethip4tcp-nat44ed-h16384-p63-s1032192-tput
avf-ethip4tcp-nat44ed-h65536-p63-s4128768-tput

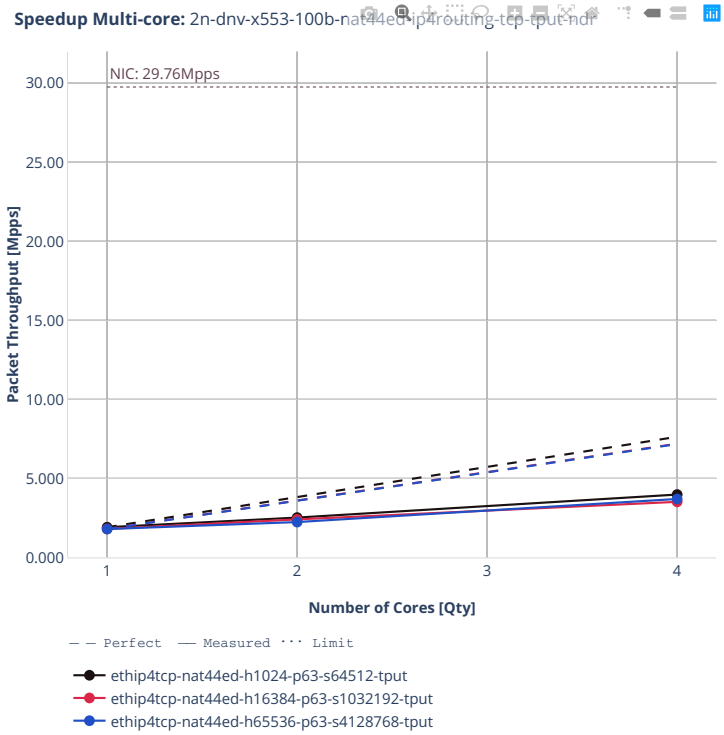
2n-dnv-x553

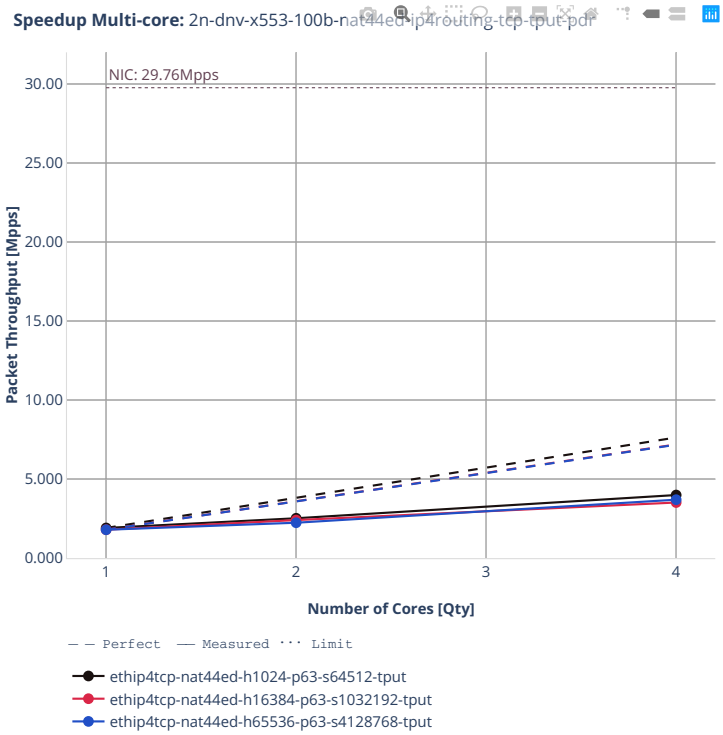
100b-nat44ed-ip4routing-tcp-tput-avf




Speedup Multi-core: 2n-dnv-x553-100b-nat44ed-ip4routing-tcp-tput-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4tcp-nat44ed-h1024-p63-s64512-tput
ethip4tcp-nat44ed-h16384-p63-s1032192-tput
ethip4tcp-nat44ed-h65536-p63-s4128768-tput

100b-nat44ed-ip4routing-tcp-tput-avf




Speedup Multi-core: 2n-dnv-x553-100b-nat44ed-ip4routing-tcp-tput-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4tcp-nat44ed-h1024-p63-s64512-tput
ethip4tcp-nat44ed-h16384-p63-s1032192-tput
ethip4tcp-nat44ed-h65536-p63-s4128768-tput

2.4.7 KVM VMs vhost-user

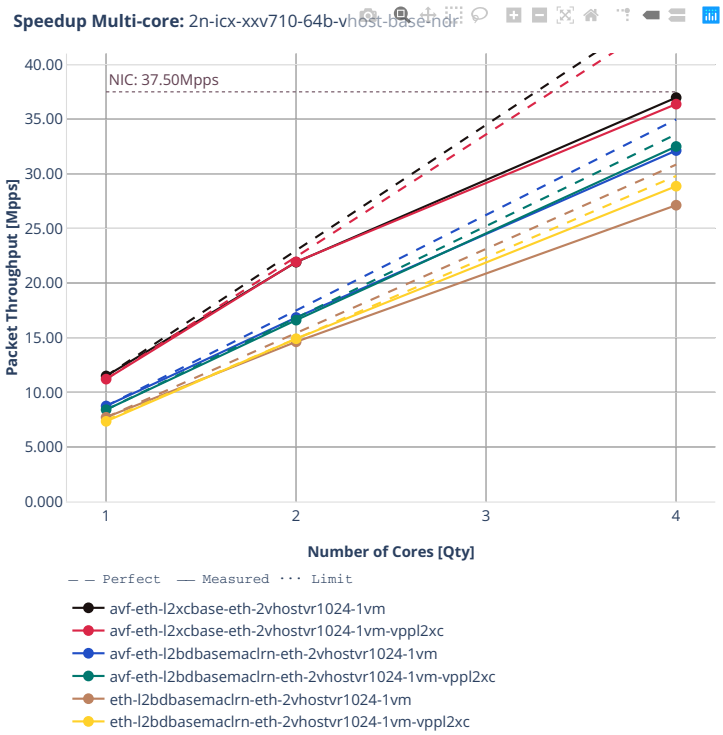
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 64B performance tests with VM vhost-user, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

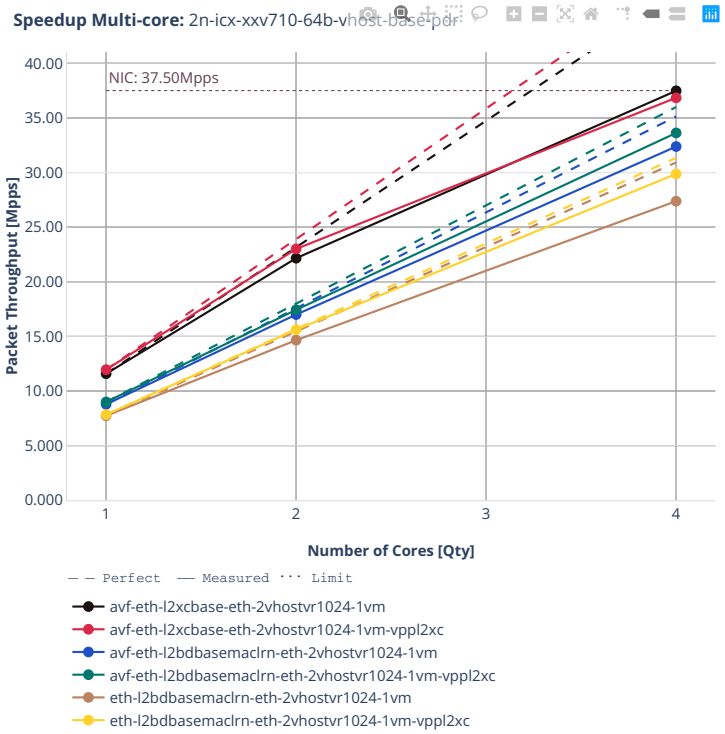
CSIT source code for the test cases used for plots can be found in CSIT git repository[139].

[139] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls2210

2n-icx-xxv710

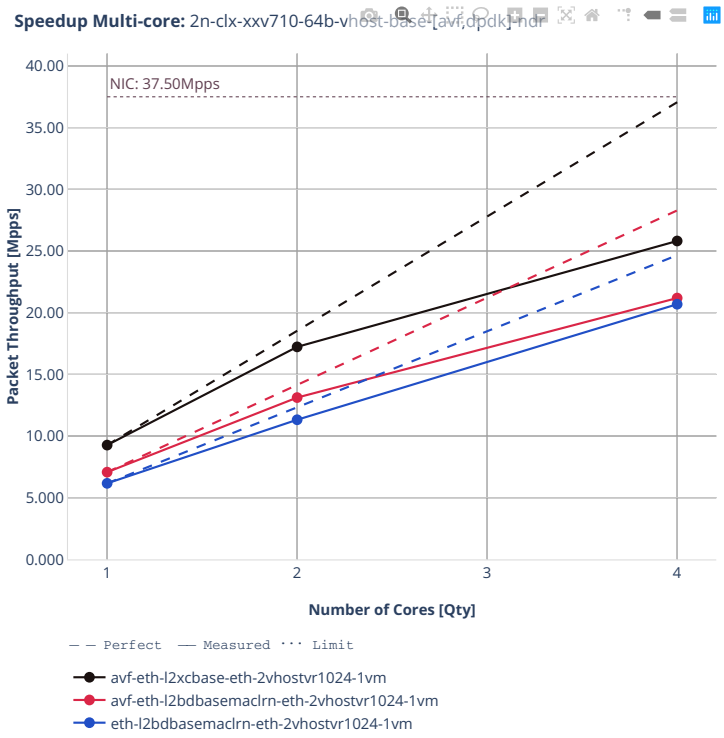
64b-vhost-base

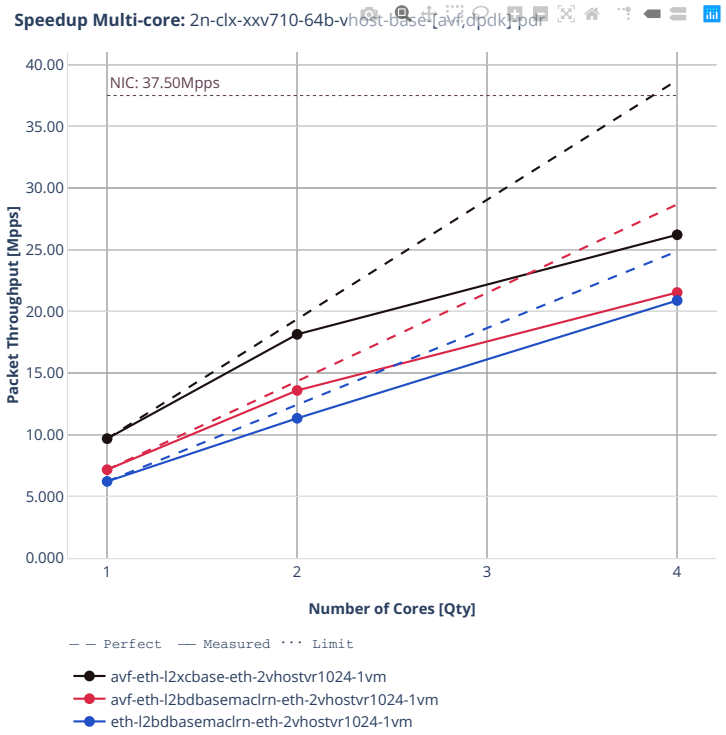



Speedup Multi-core: 2n-icx-xxv710-64b-vhost-base-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2vhostvr1024-1vm
avf-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

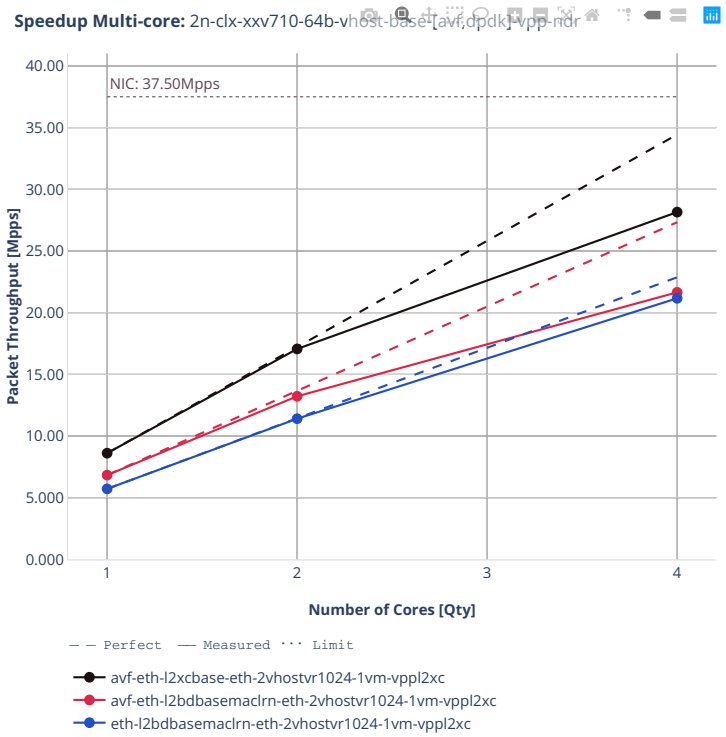
2n-clx-xxv710

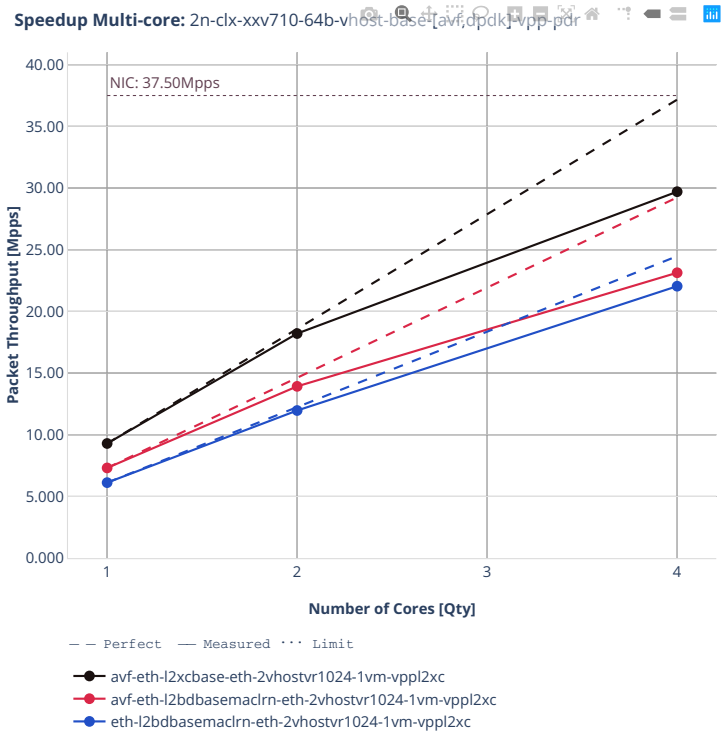
64b-vhost-base-testpmd




Speedup Multi-core: 2n-clx-xxv710-64b-vhost-base-[avf,dpdk]-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2vhostvr1024-1vm
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm

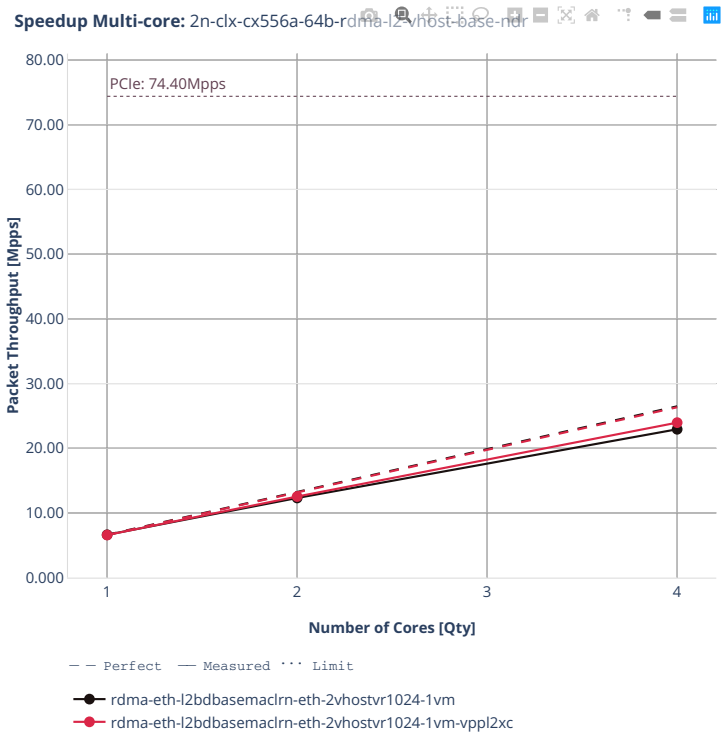
64b-vhost-base-vpp

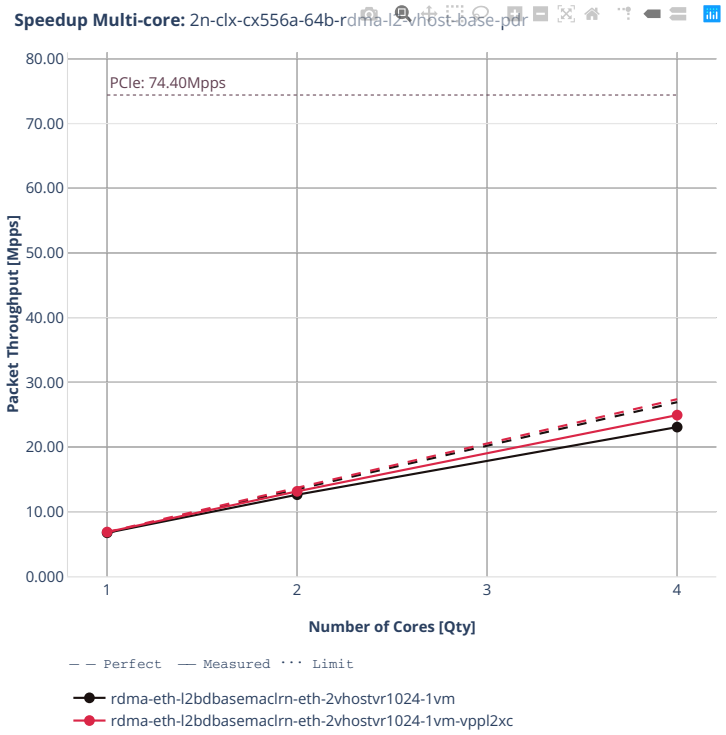



Speedup Multi-core: 2n-clx-xxv710-64b-vhost-base-[avf,dpdk]-vpp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2n-clx-cx556a

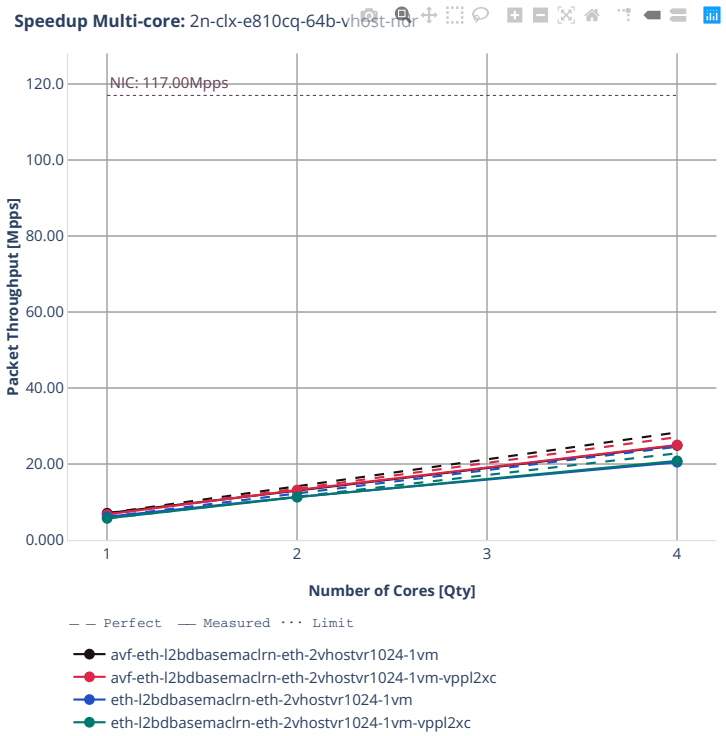
64b-vhost-base-rdma-core

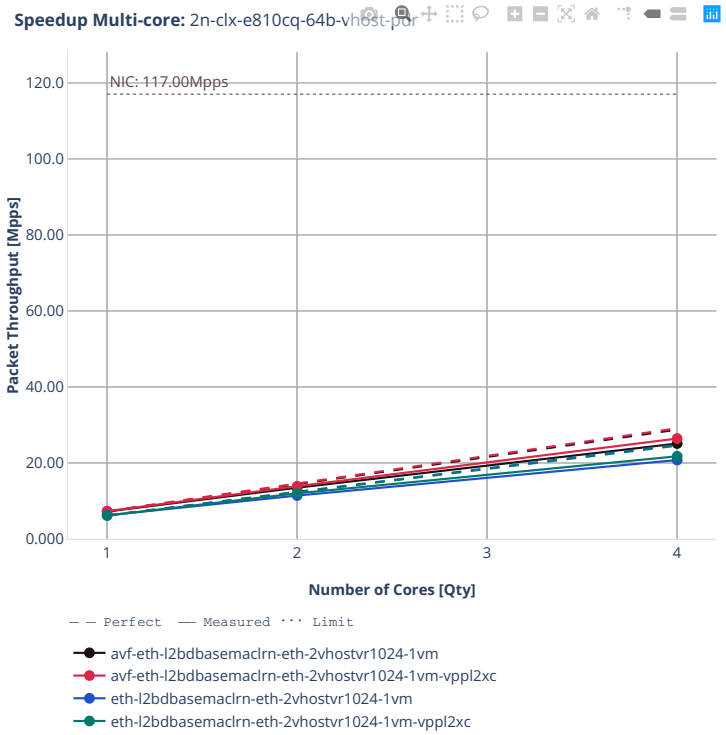



Speedup Multi-core: 2n-clx-cx556a-64b-rdma-l2-vhost-base-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2n-clx-e810cq

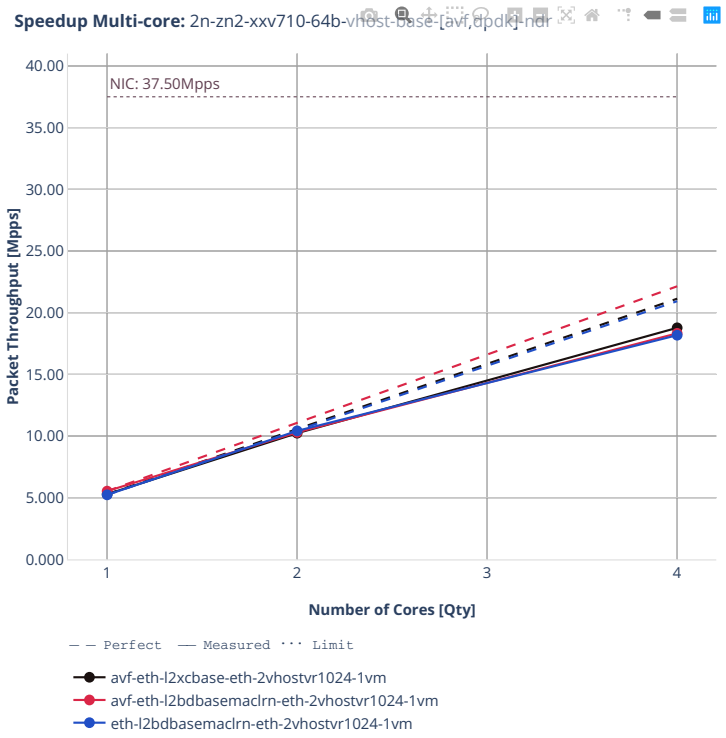
64b-vhost

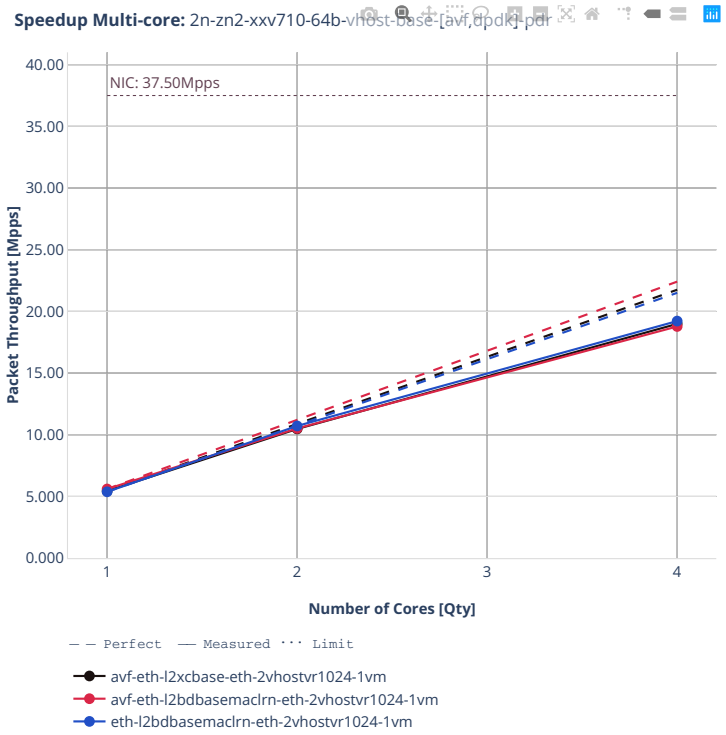



Speedup Multi-core: 2n-clx-e810cq-64b-vhost-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

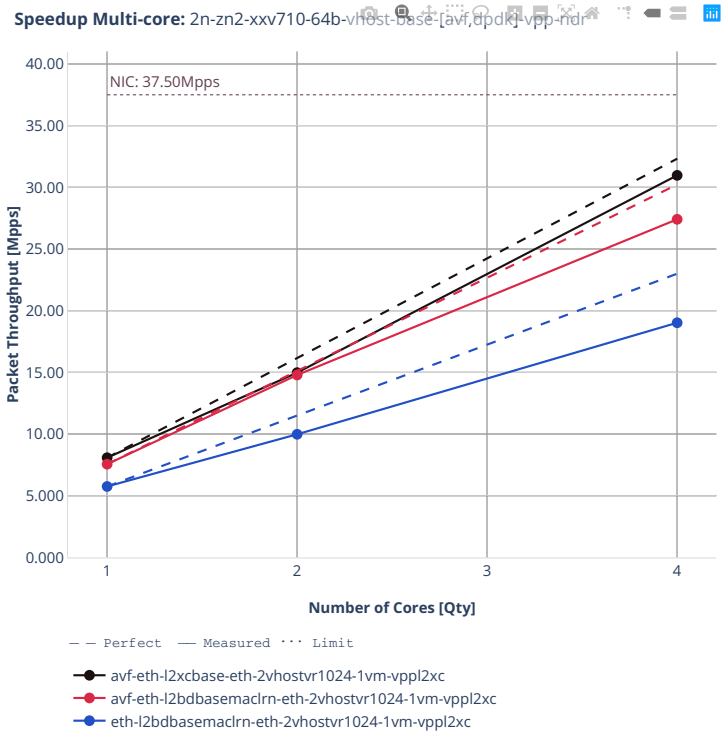
2n-zn2-xxv710

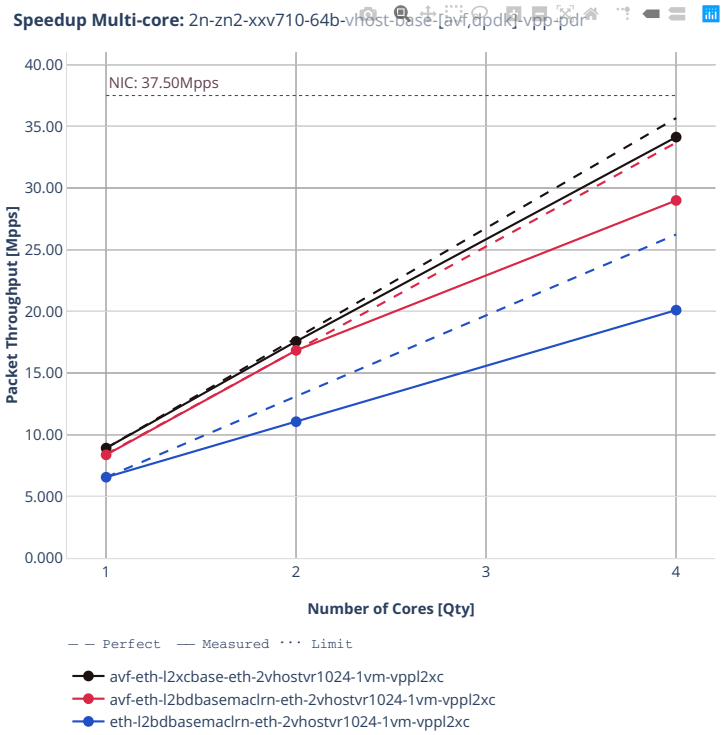
64b-vhost-base-testpmd




Speedup Multi-core: 2n-zn2-xxv710-64b-vhost-base-[avf,dpdk]-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2vhostvr1024-1vm
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm

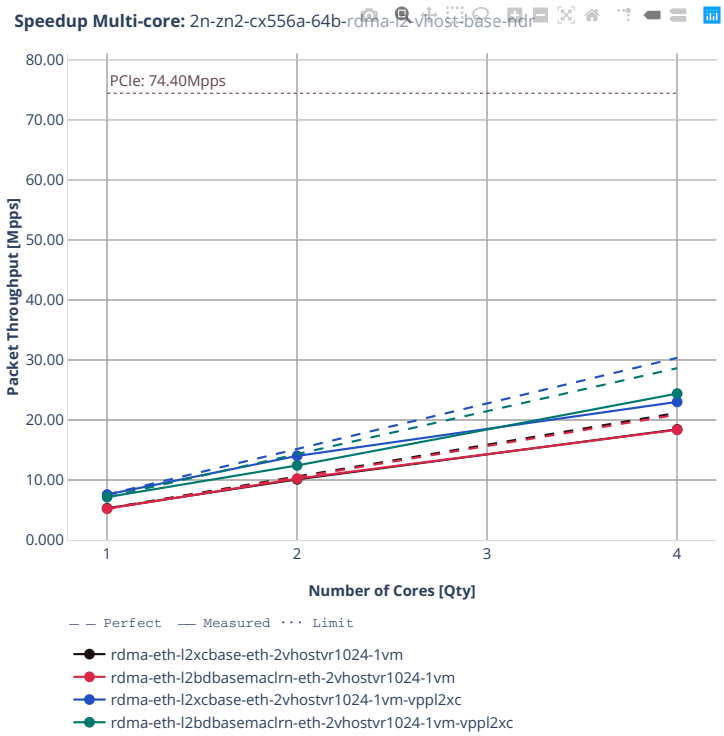
64b-vhost-base-vpp

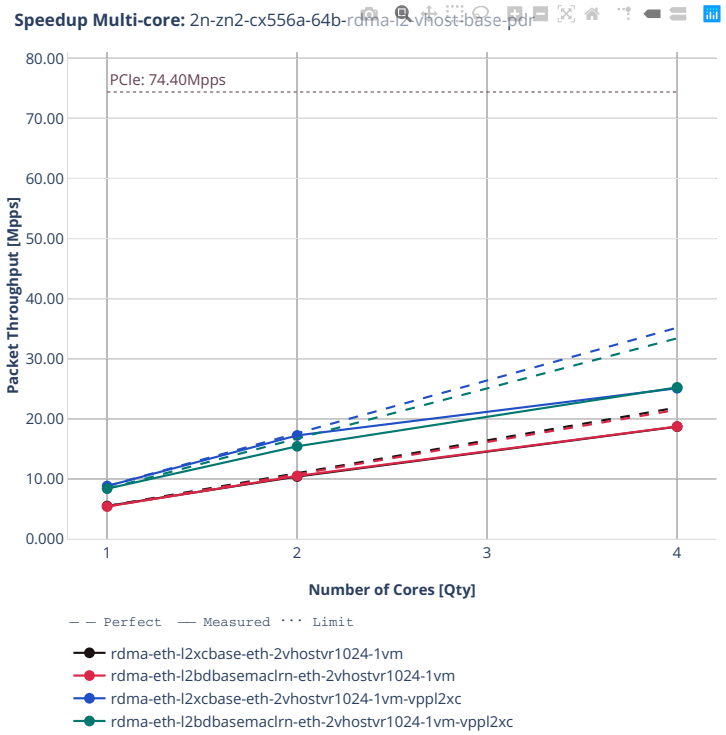



Speedup Multi-core: 2n-zn2-xxv710-64b-vhost-base-[avf,dpdk]-vpp-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2n-zn2-cx556a

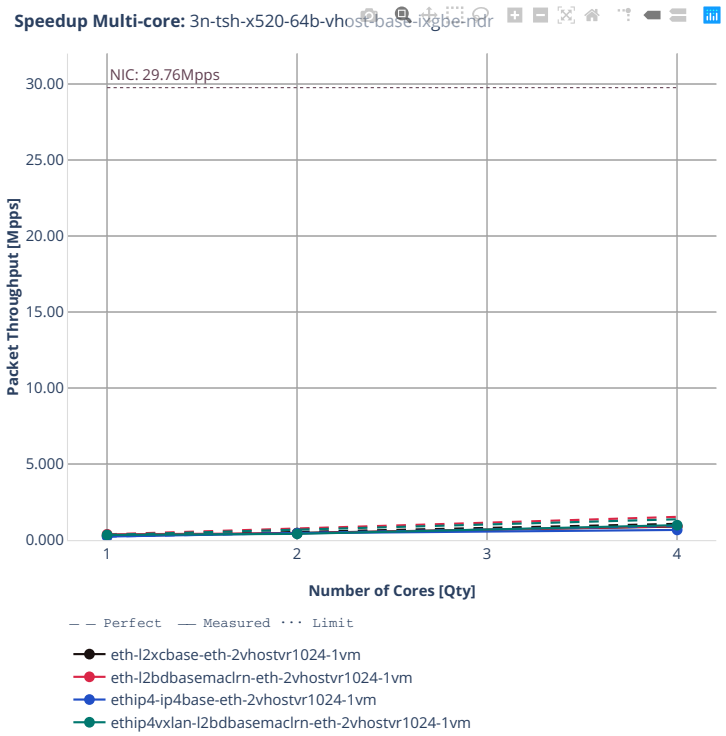
64b-vhost-base-rdma-core

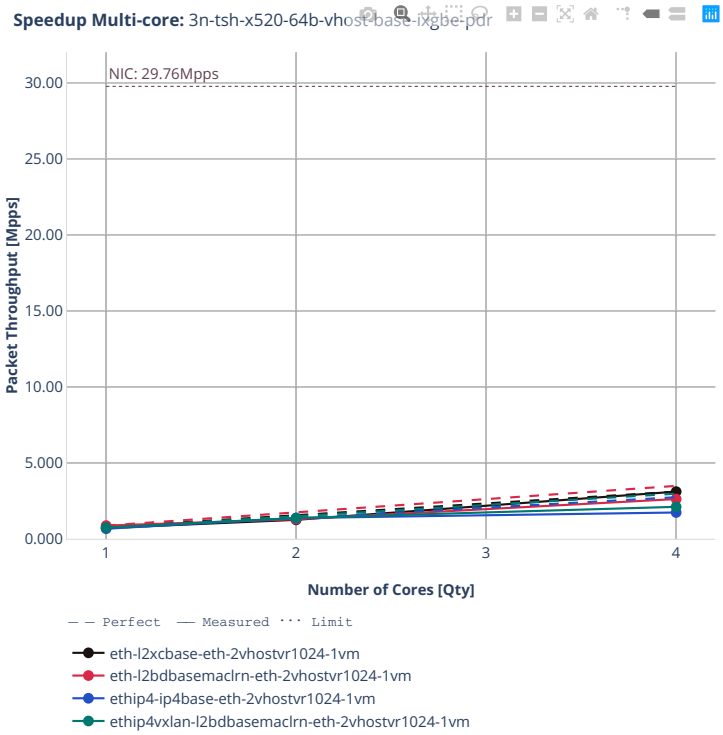



Speedup Multi-core: 2n-zn2-cx556a-64b-rdma-l2-vhost-base-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-eth-l2xcbase-eth-2vhostvr1024-1vm
rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
rdma-eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
rdma-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

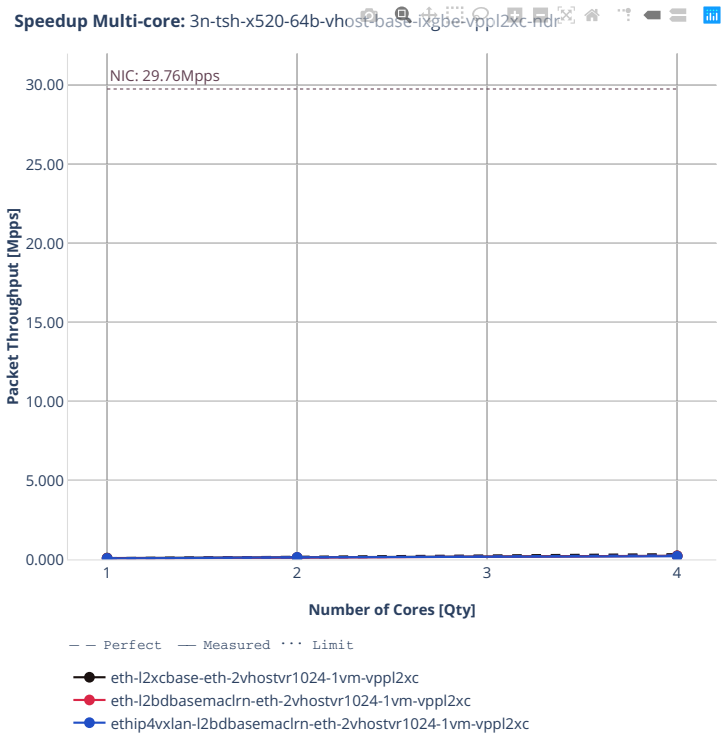
3n-tsh-x520

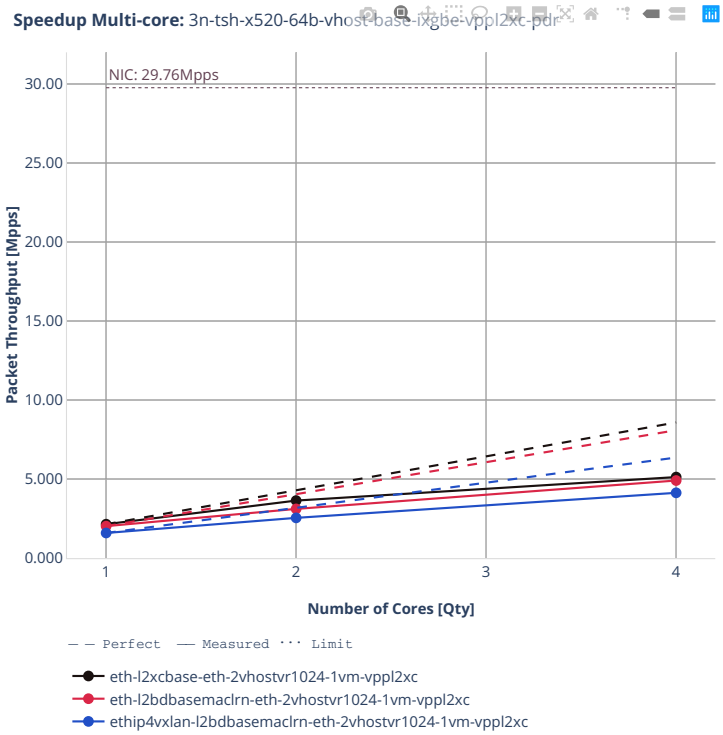
64b-vhost-base-ixgbe




Speedup Multi-core: 3n-tsh-x520-64b-vhost-base-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-eth-2vhostvr1024-1vm
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
ethip4-ip4base-eth-2vhostvr1024-1vm
ethip4vxlan-l2bdbasemaclrn-eth-2vhostvr1024-1vm

64b-vhost-base-ixgbe-vppl2xc




Speedup Multi-core: 3n-tsh-x520-64b-vhost-base-ixgbe-vppl2xc-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-eth-2vhostvr1024-1vm-vppl2xc
eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
ethip4vxlan-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

2.4.8 LXC/DRC Container Memif

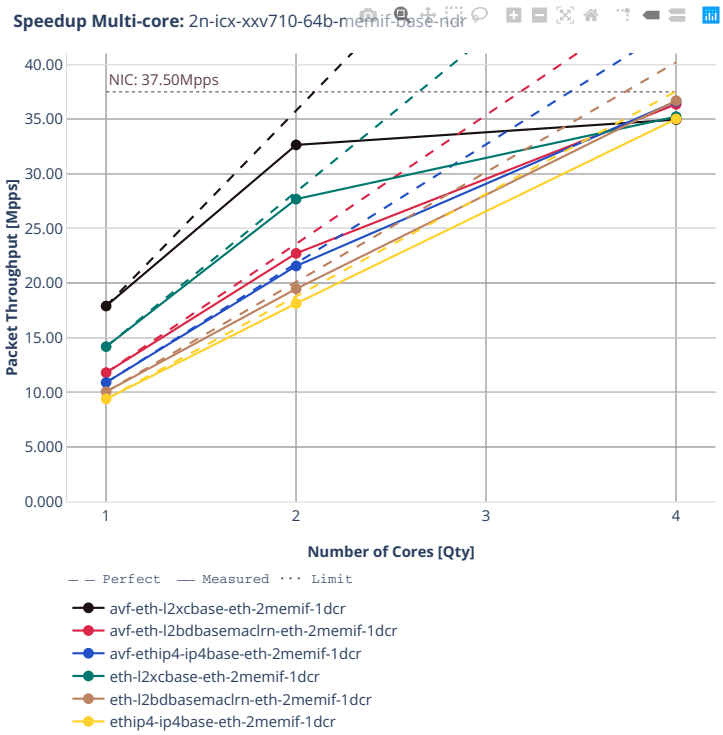
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

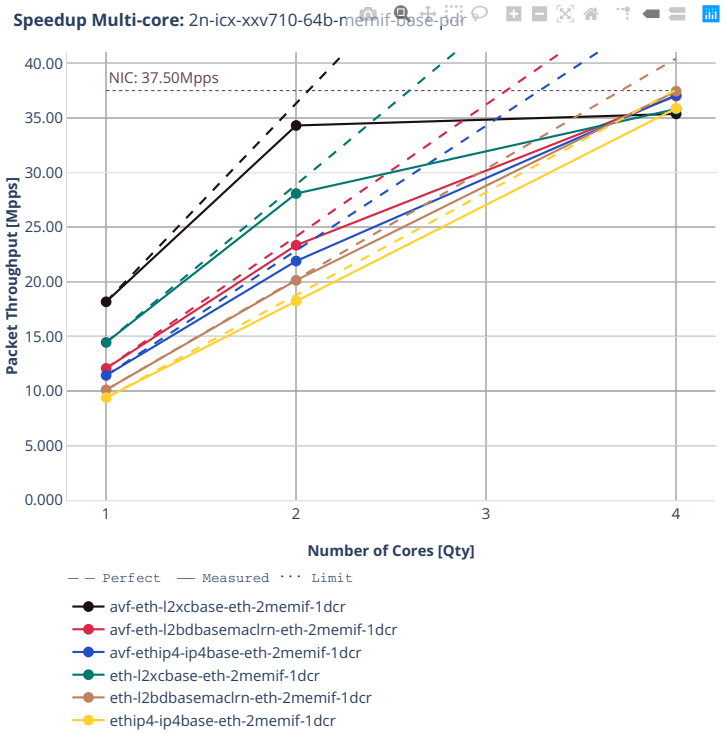
CSIT source code for the test cases used for plots can be found in CSIT git repository[140].

[140] https://git.fd.io/csit/tree/tests/vpp/perf/container_memif?h=rls2210

2n-icx-xxv710

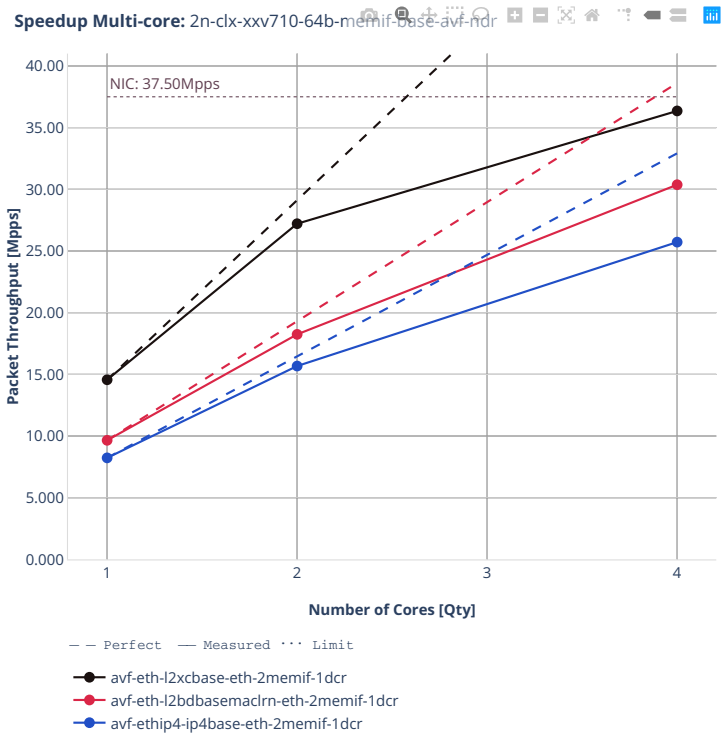
64b-memif-base

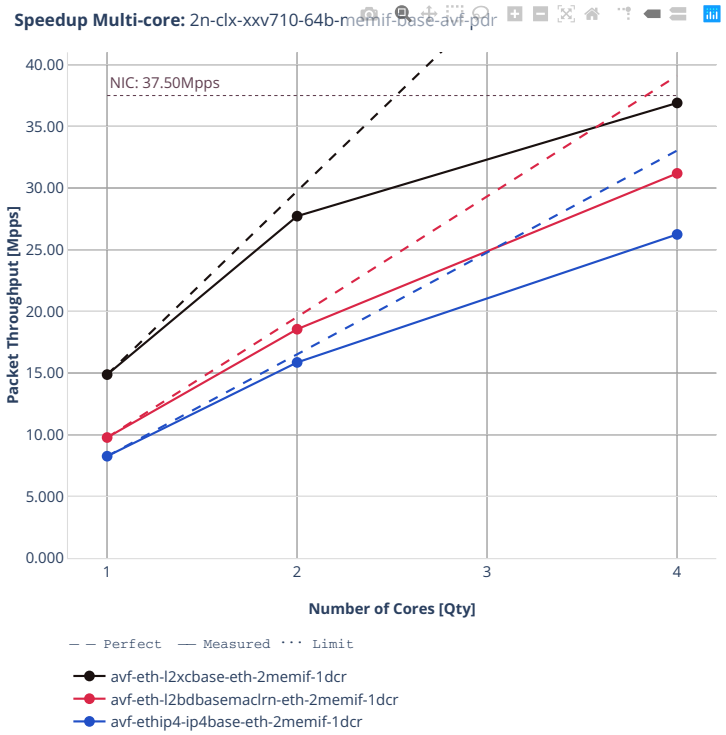



Speedup Multi-core: 2n-icx-xxv710-64b-memif-base-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2memif-1dcr
avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
avf-ethip4-ip4base-eth-2memif-1dcr
eth-l2xcbase-eth-2memif-1dcr
eth-l2bdbasemaclrn-eth-2memif-1dcr
ethip4-ip4base-eth-2memif-1dcr

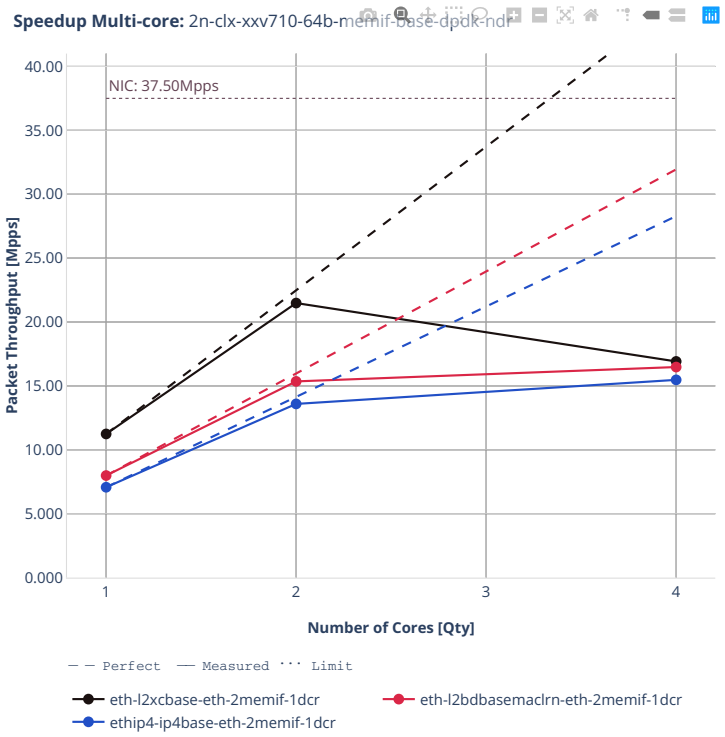
2n-clx-xxv710

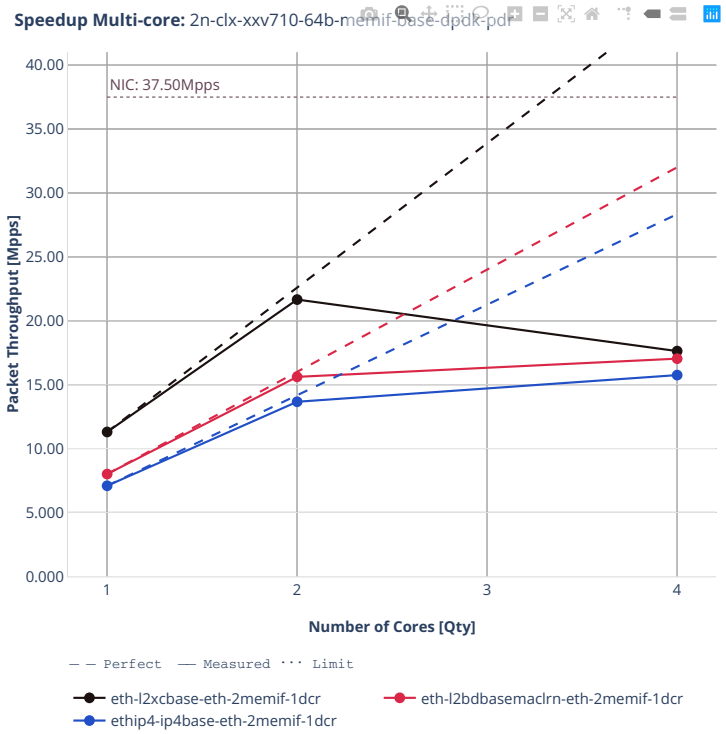
64b-memif-base-avf




Speedup Multi-core: 2n-clx-xxv710-64b-memif-base-avf-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2memif-1dcr
avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
avf-ethip4-ip4base-eth-2memif-1dcr

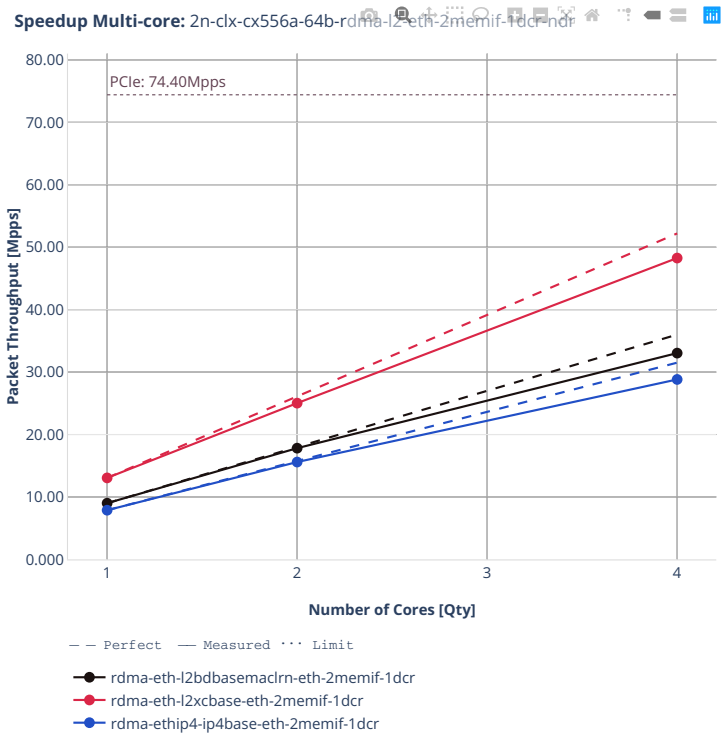
64b-memif-base-dpdk

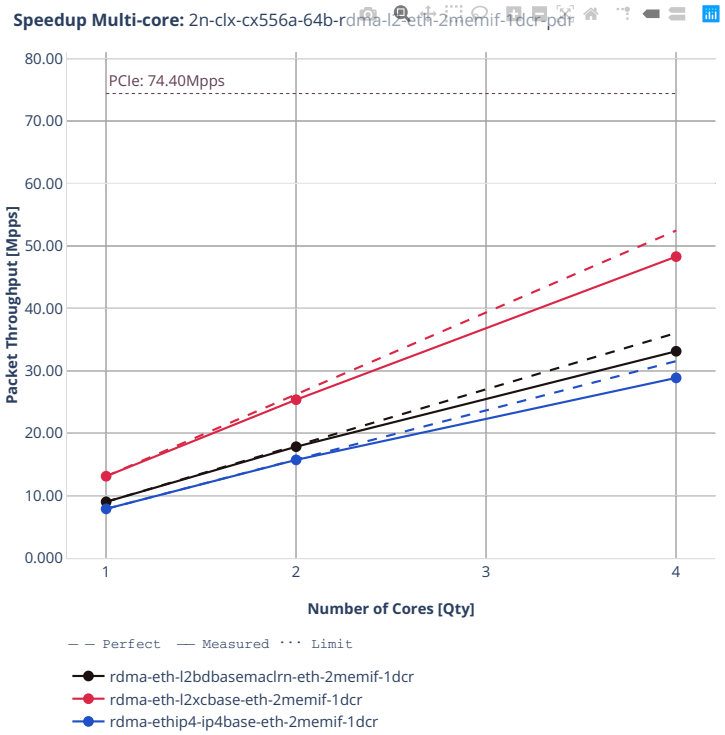



Speedup Multi-core: 2n-clx-xxv710-64b-memif-base-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-eth-2memif-1dcr
eth-l2bdbasemaclrn-eth-2memif-1dcr
ethip4-ip4base-eth-2memif-1dcr

2n-clx-cx556a

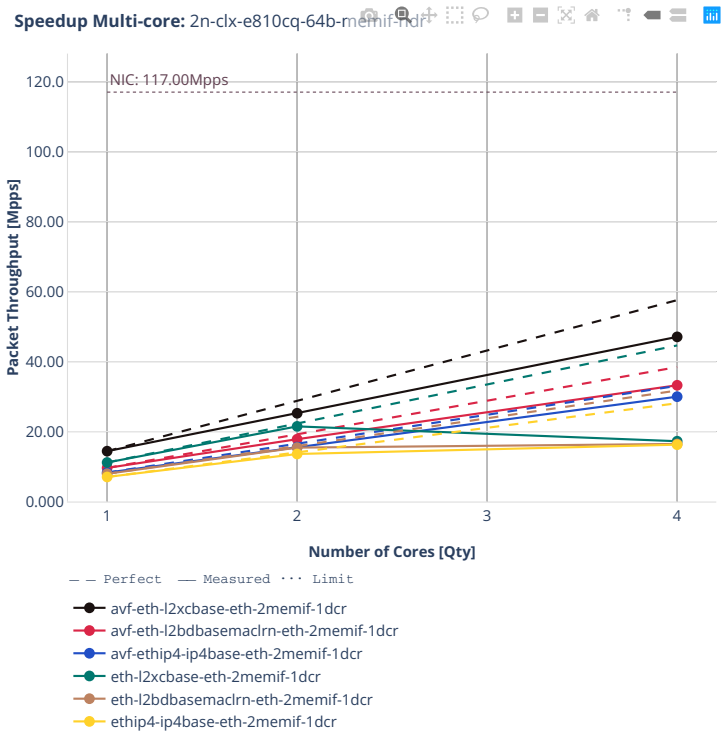
64b-memif-base-rdma-core

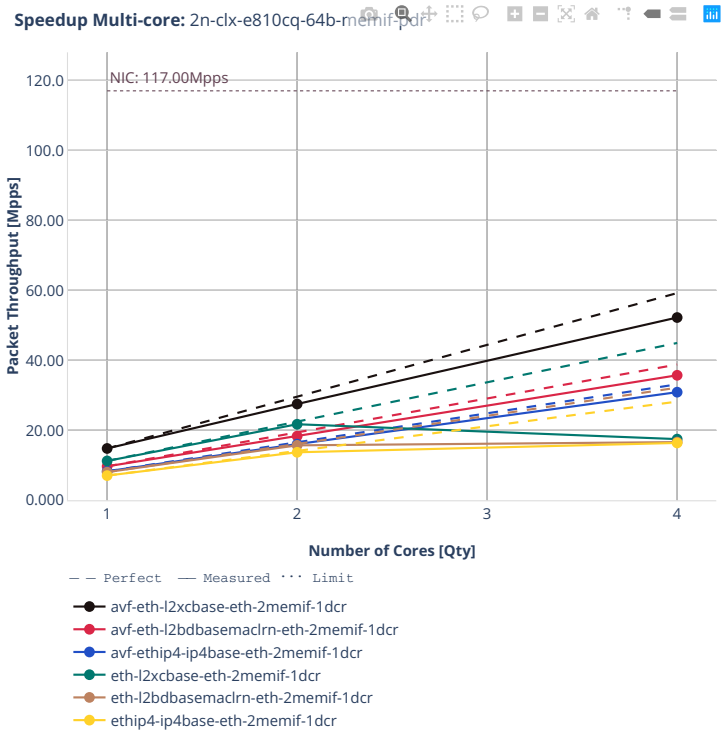



Speedup Multi-core: 2n-clx-cx556a-64b-rdma-l2-eth-2memif-1dcr-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-eth-l2bdbasemaclrn-eth-2memif-1dcr
rdma-eth-l2xcbase-eth-2memif-1dcr
rdma-ethip4-ip4base-eth-2memif-1dcr

2n-clx-e810cq

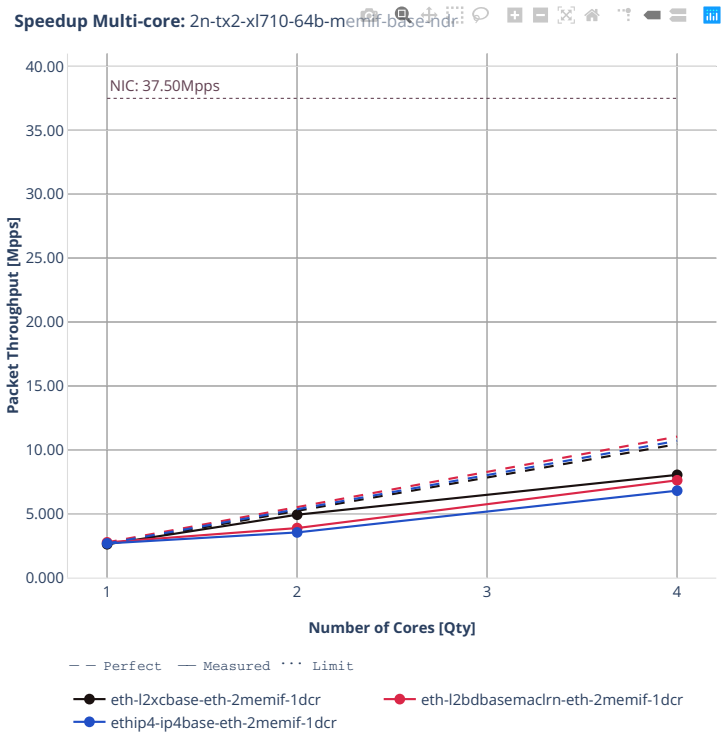
64b-memif-base

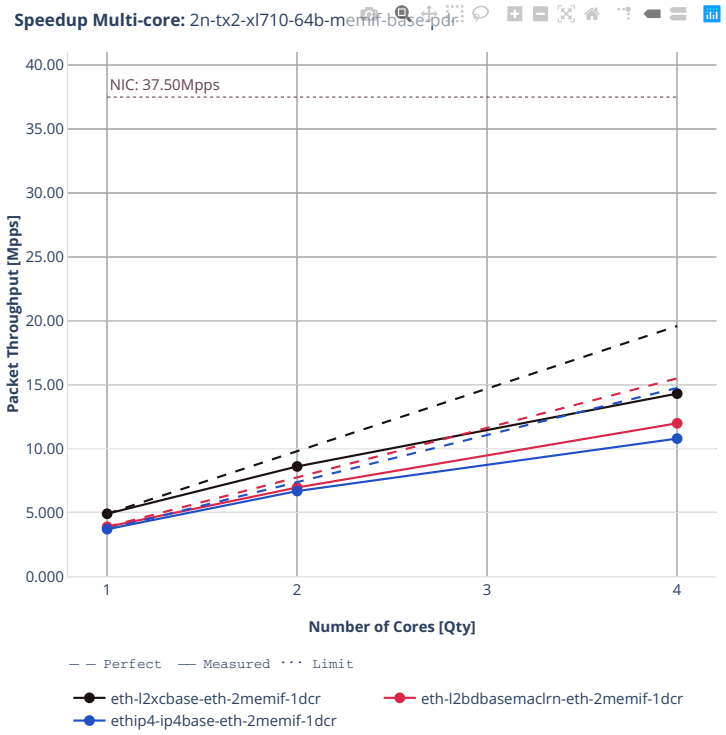



Speedup Multi-core: 2n-clx-e810cq-64b-memif-pdr
NIC: 117.00Mpps
120.0
100.0
80.00
60.00
40.00
20.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2memif-1dcr
avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
avf-ethip4-ip4base-eth-2memif-1dcr
eth-l2xcbase-eth-2memif-1dcr
eth-l2bdbasemaclrn-eth-2memif-1dcr
ethip4-ip4base-eth-2memif-1dcr

2n-tx2-xl710

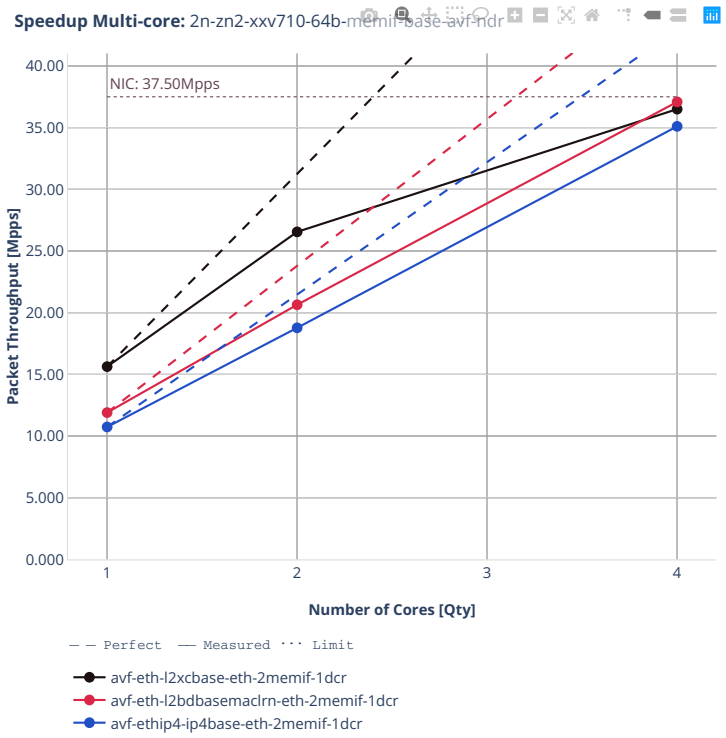
64b-memif-base-dpdk

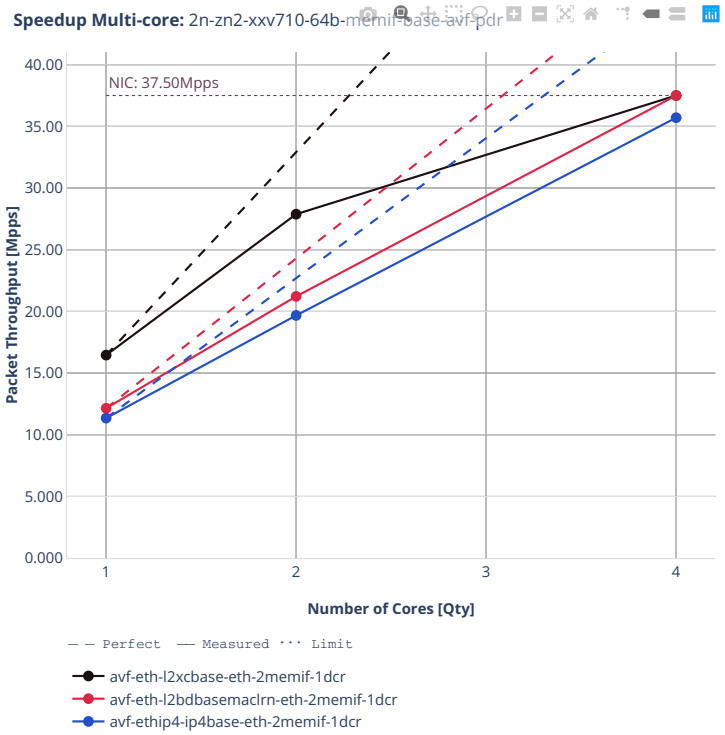



Speedup Multi-core: 2n-tx2-xl710-64b-memif-base-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-eth-2memif-1dcr
eth-l2bdbasemaclrn-eth-2memif-1dcr
ethip4-ip4base-eth-2memif-1dcr

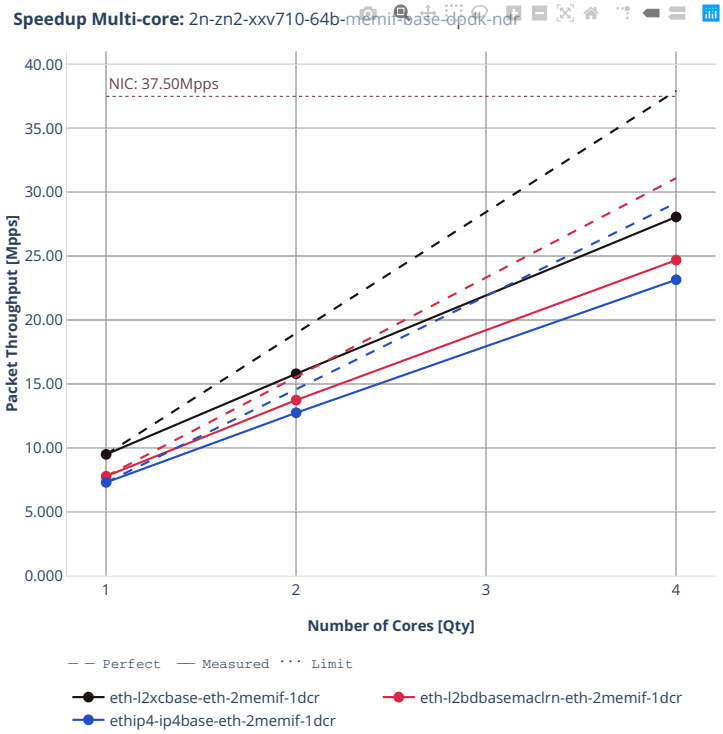
2n-zn2-xxv710

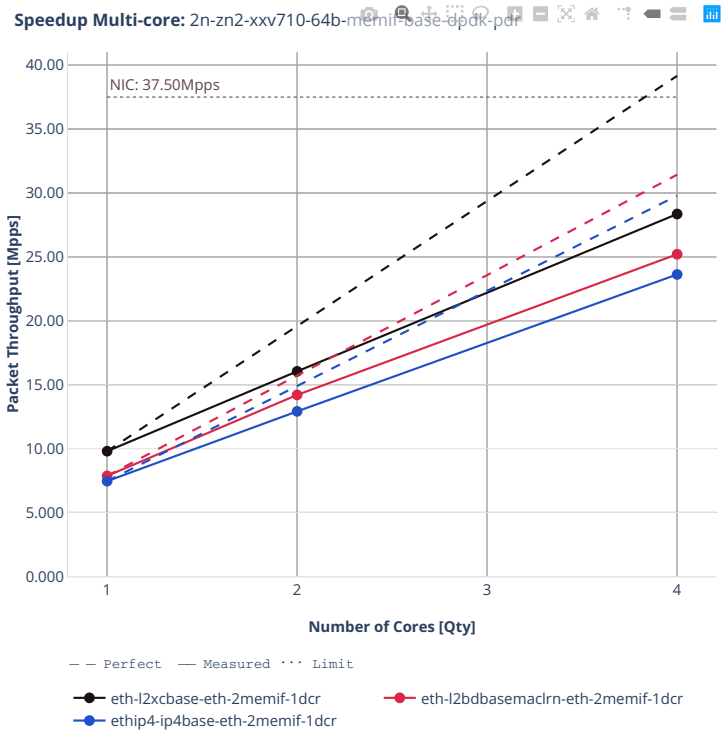
64b-memif-base-avf




Speedup Multi-core: 2n-zn2-xxv710-64b-memif-base-avf-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-eth-l2xcbase-eth-2memif-1dcr
avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
avf-ethip4-ip4base-eth-2memif-1dcr

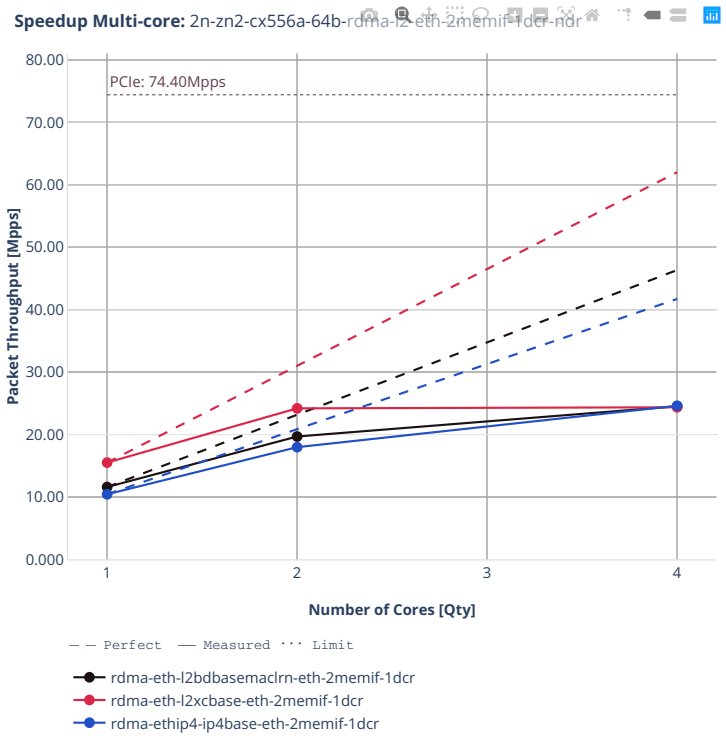
64b-memif-base-dpdk

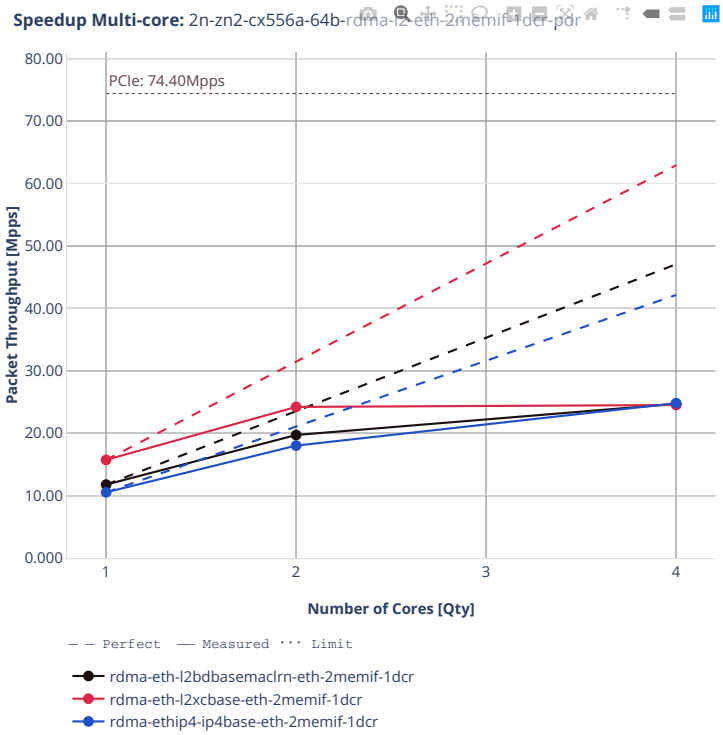



Speedup Multi-core: 2n-zn2-xxv710-64b-memif-base-dpdk-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-eth-2memif-1dcr
eth-l2bdbasemaclrn-eth-2memif-1dcr
ethip4-ip4base-eth-2memif-1dcr

2n-zn2-cx556a

64b-memif-base-rdma-core




Speedup Multi-core: 2n-zn2-cx556a-64b-rdma-l2-eth-2memif-1dcr-pdr
PCIe: 74.40Mpps
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
rdma-eth-l2bdbasemaclrn-eth-2memif-1dcr
rdma-eth-l2xcbase-eth-2memif-1dcr
rdma-ethip4-ip4base-eth-2memif-1dcr

2.4.9 IPSec IPv4 Routing

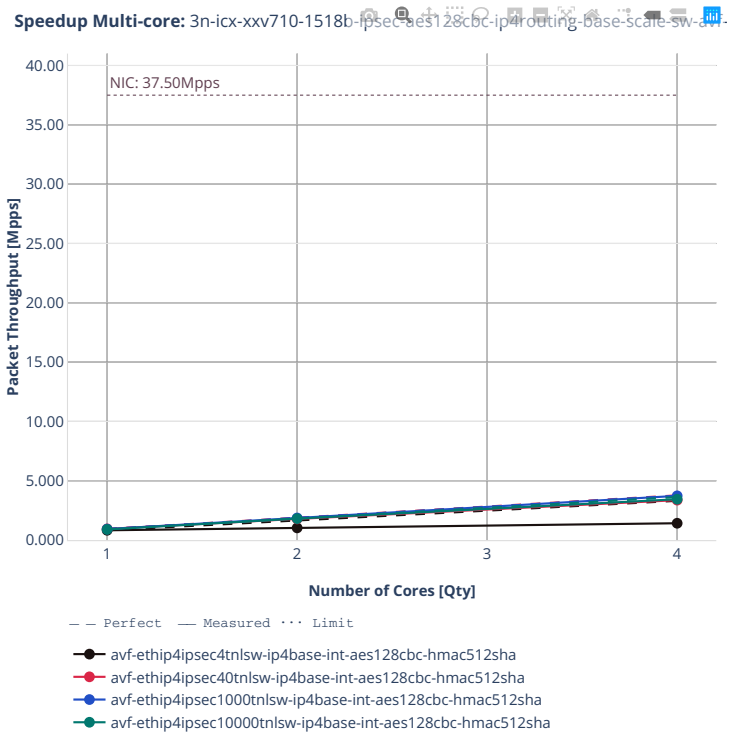
Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. VPP IPSec encryption is accelerated using DPDK cryptodev library driving Intel Quick Assist (QAT) crypto PCIe hardware cards. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

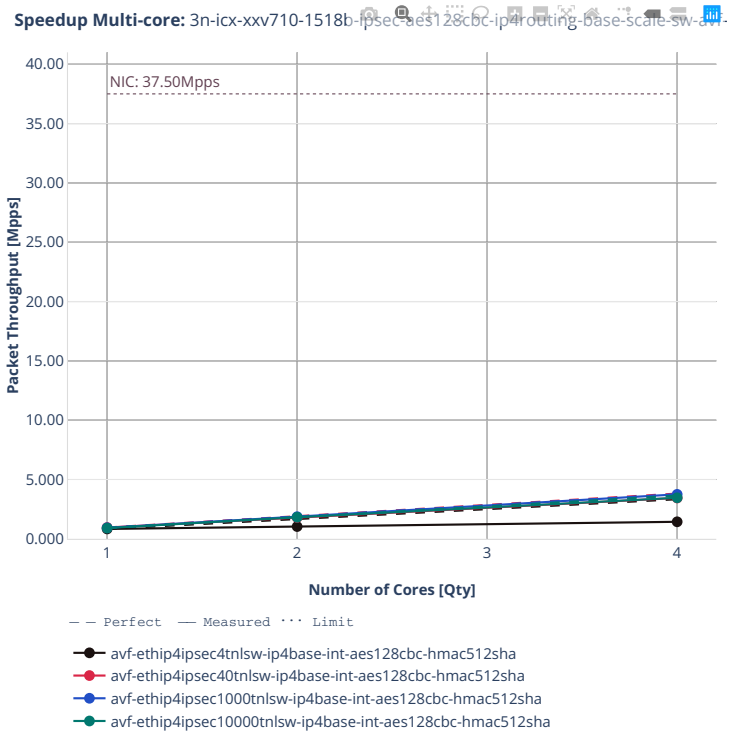
CSIT source code for the test cases used for plots can be found in CSIT git repository[141].

[141] https://git.fd.io/csit/tree/tests/vpp/perf/crypto?h=rls2210

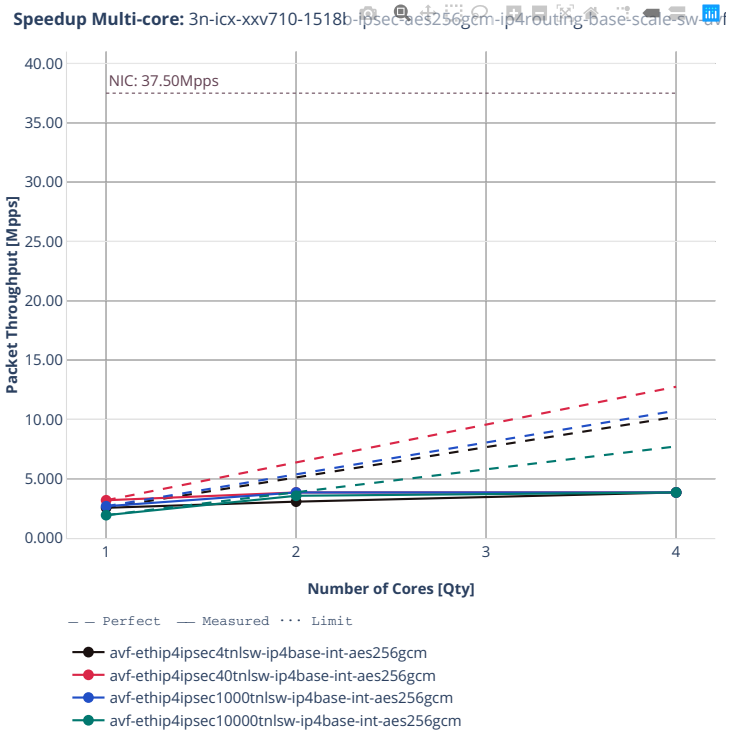
3n-icx-xxv710

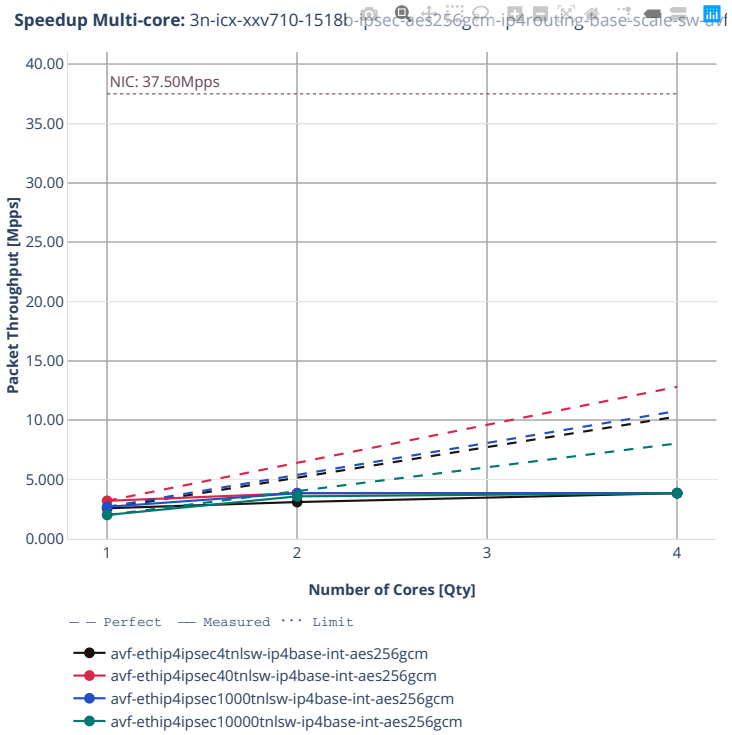
1518b-2t1c-ipsec-aes128cbc-ip4routing-base-scale-sw-avf



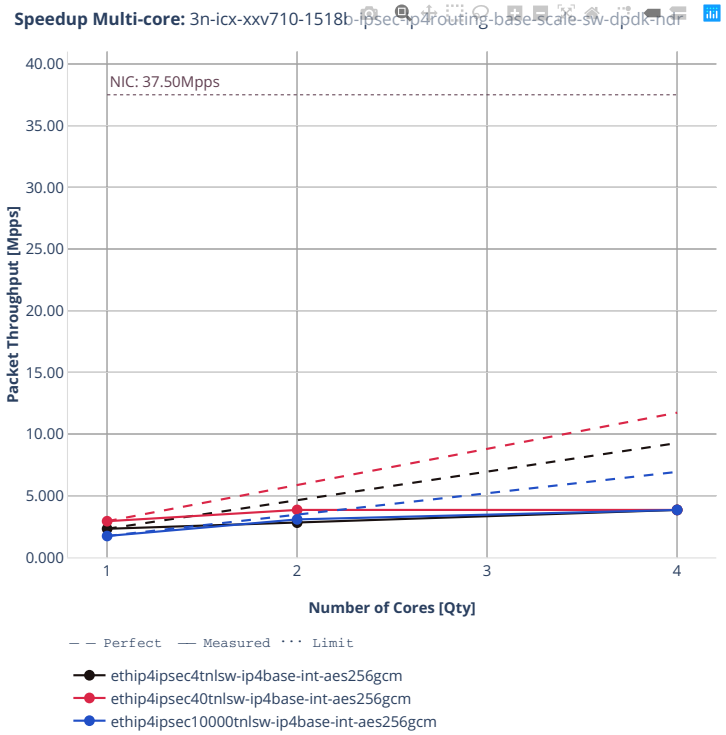

Speedup Multi-core: 3n-icx-xxv710-1518b-ipsec-aes128cbc-ip4routing-base-scale-sw-avf-
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
avf-ethip4ipsec40tnlsw-ip4base-int-aes128cbc-hmac512sha
avf-ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
avf-ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha

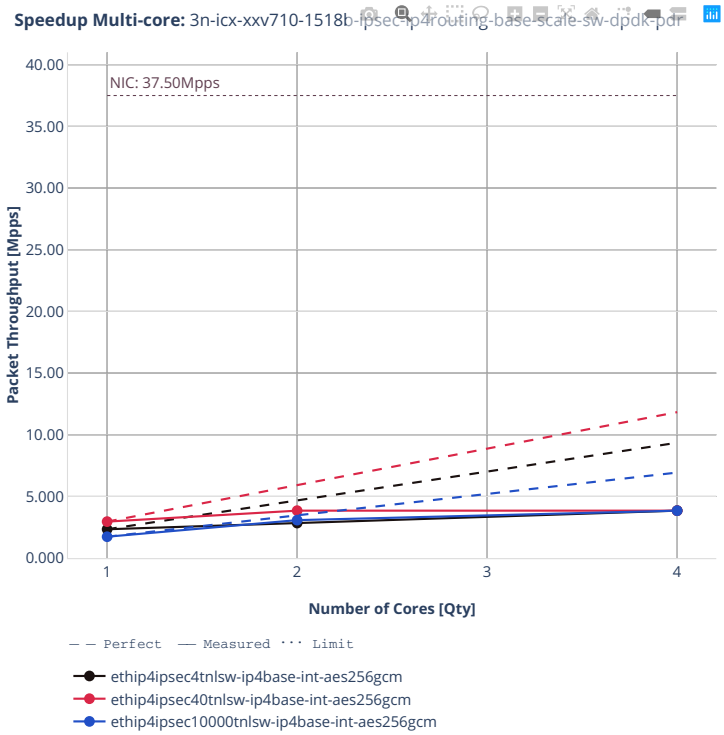
1518b-2t1c-ipsec-aes256gcm-ip4routing-base-scale-sw-avf



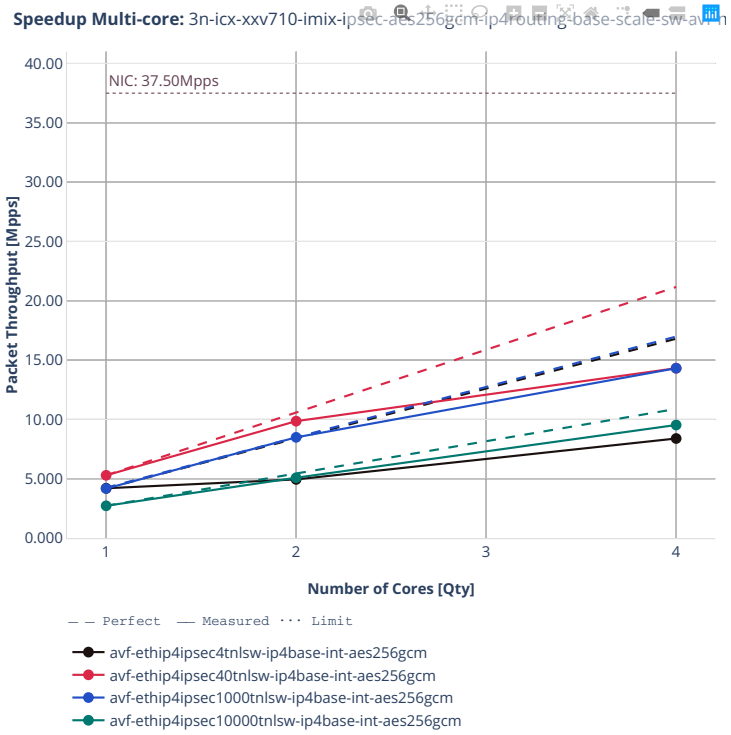

Speedup Multi-core: 3n-icx-xxv710-1518b-ipsec-aes256gcm-ip4routing-base-scale-sw-ull
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4ipsec4tnlsw-ip4base-int-aes256gcm
avf-ethip4ipsec40tnlsw-ip4base-int-aes256gcm
avf-ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
avf-ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

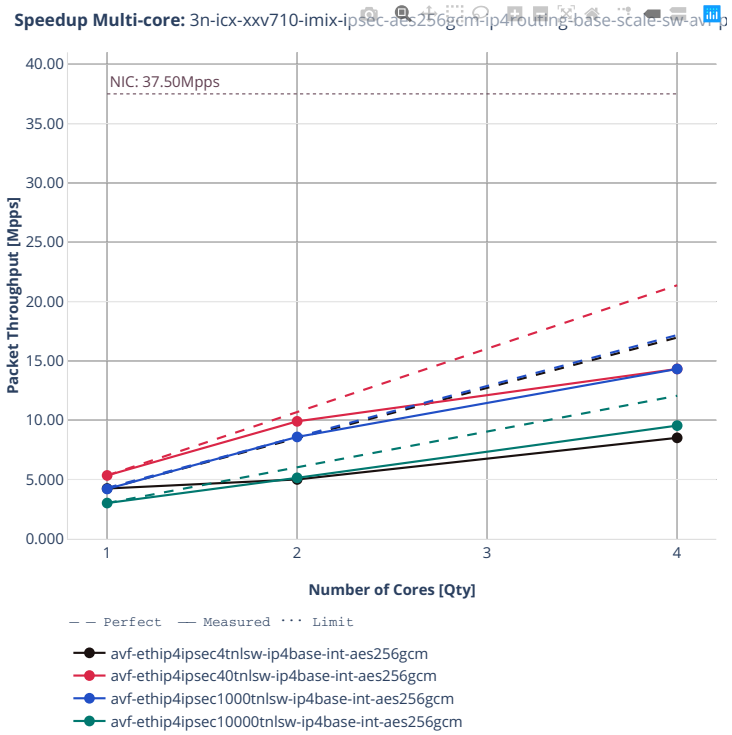
1518b-2t1c-ipsec-ip4routing-base-scale-sw-dpdk



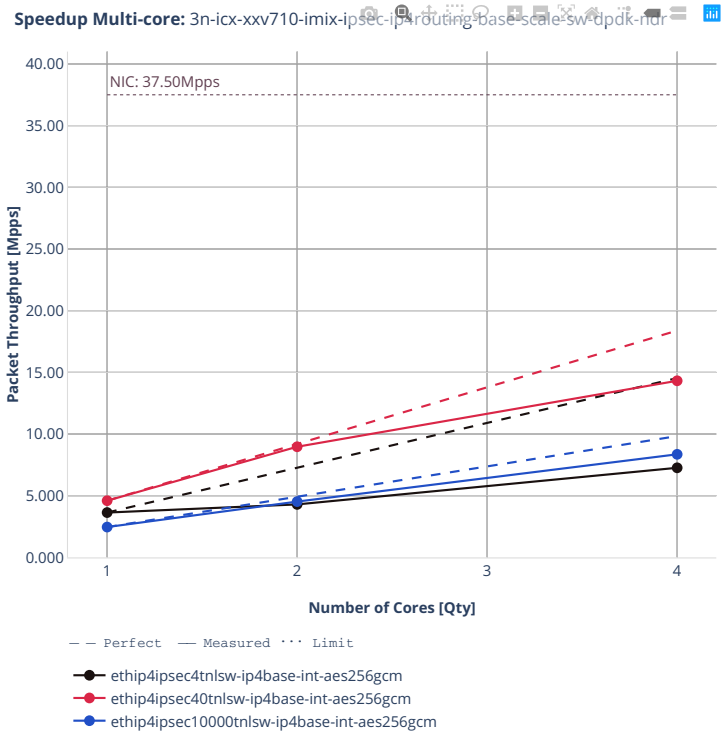

Speedup Multi-core: 3n-icx-xxv710-1518b-ipsec-ip4routing-base-scale-sw-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes256gcm
ethip4ipsec40tnlsw-ip4base-int-aes256gcm
ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

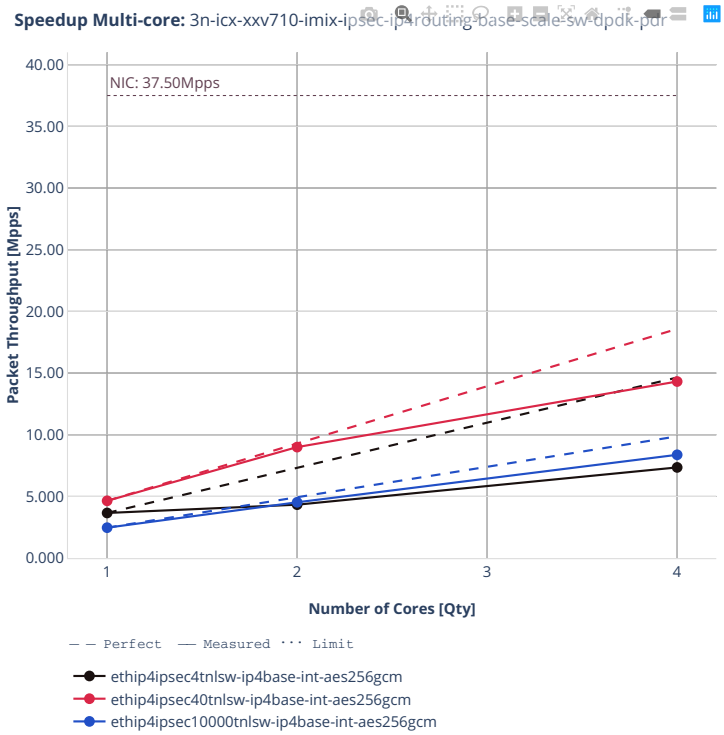
imix-2t1c-ipsec-aes256gcm-ip4routing-base-scale-sw-avf




Speedup Multi-core: 3n-icx-xxv710-imix-ipsec-aes256gcm-ip4routing-base-scale-sw-avf
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
avf-ethip4ipsec4tnlsw-ip4base-int-aes256gcm
avf-ethip4ipsec40tnlsw-ip4base-int-aes256gcm
avf-ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
avf-ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

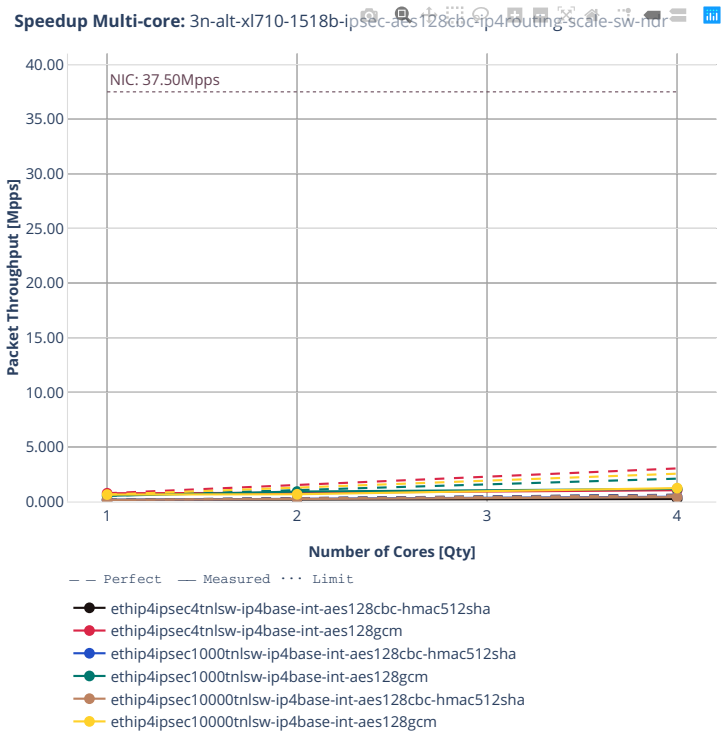
imix-2t1c-ipsec-ip4routing-base-scale-sw-dpdk

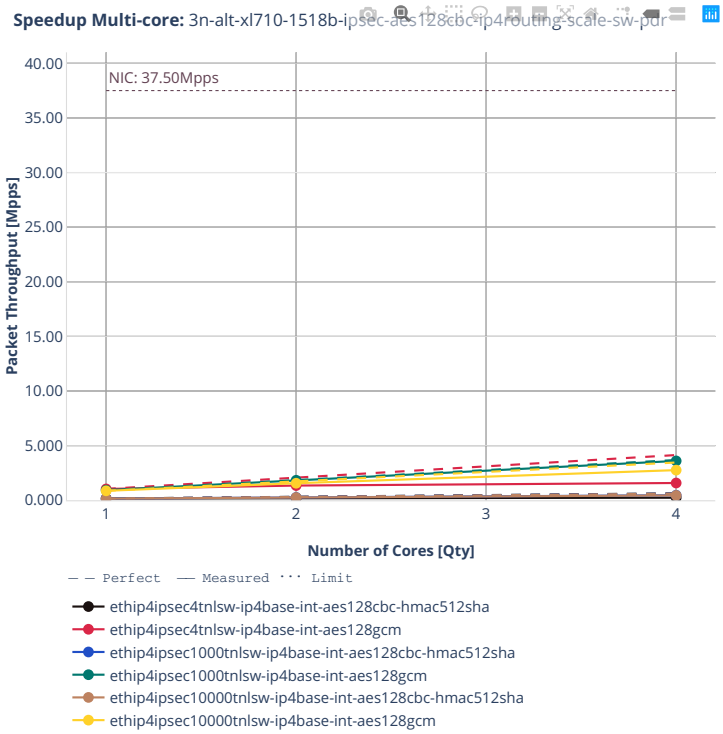



Speedup Multi-core: 3n-icx-xxv710-imix-ipsec-ip4routing-base-scale-sw-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes256gcm
ethip4ipsec40tnlsw-ip4base-int-aes256gcm
ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

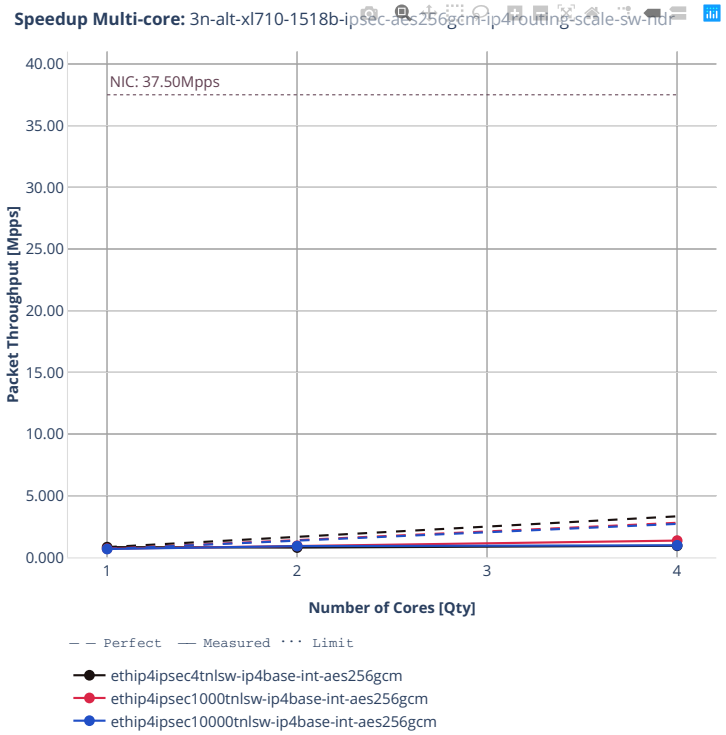
3n-alt-xl710

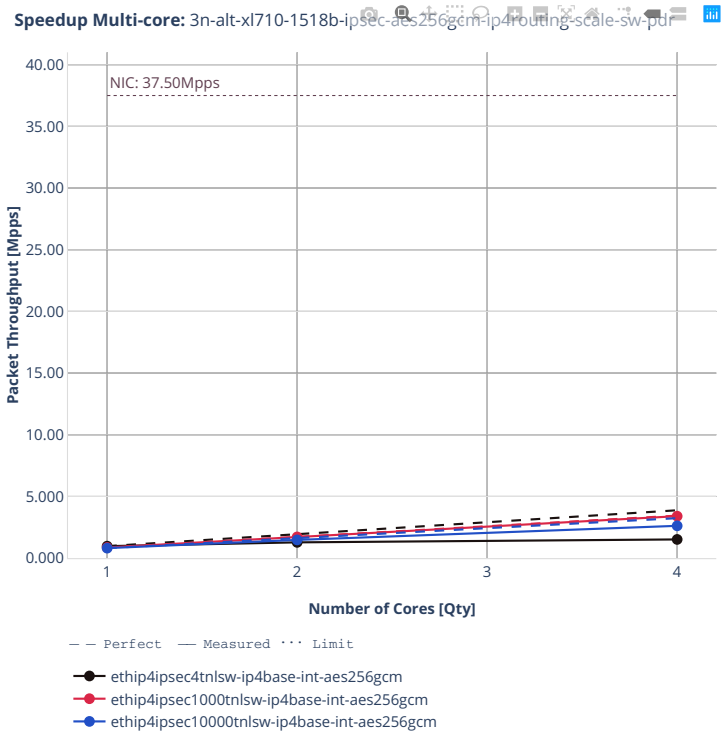
1518b-ipsec-aes128cbc-ip4routing-scale-sw



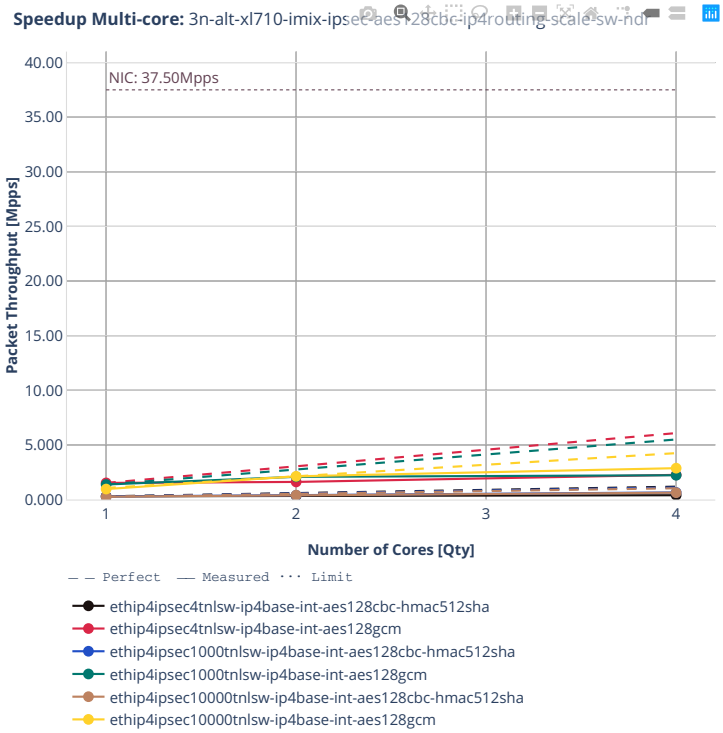

Speedup Multi-core: 3n-alt-xl710-1518b-ipsec-aes128cbc-ip4routing-scale-sw-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec4tnlsw-ip4base-int-aes128gcm
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes128gcm
ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec10000tnlsw-ip4base-int-aes128gcm

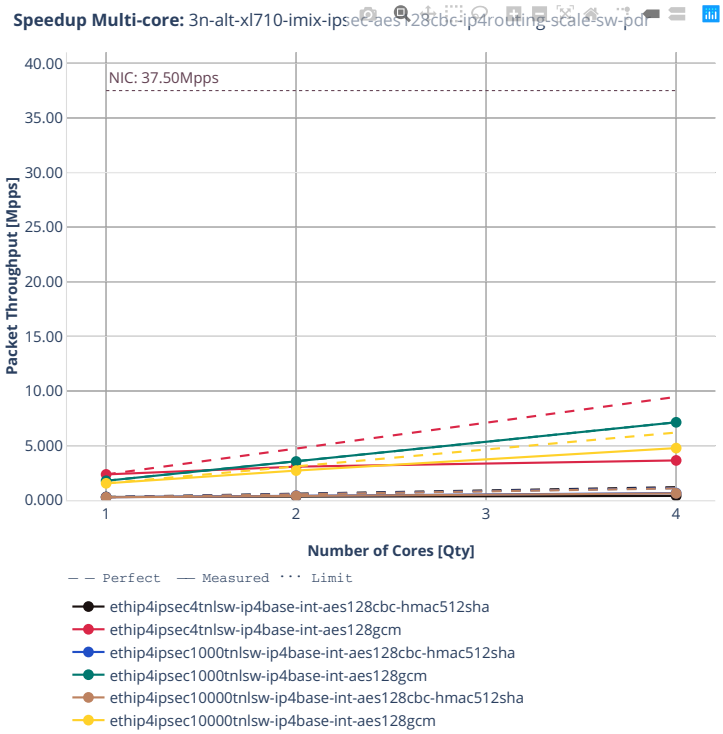
1518b-ipsec-aes256gcm-ip4routing-scale-sw



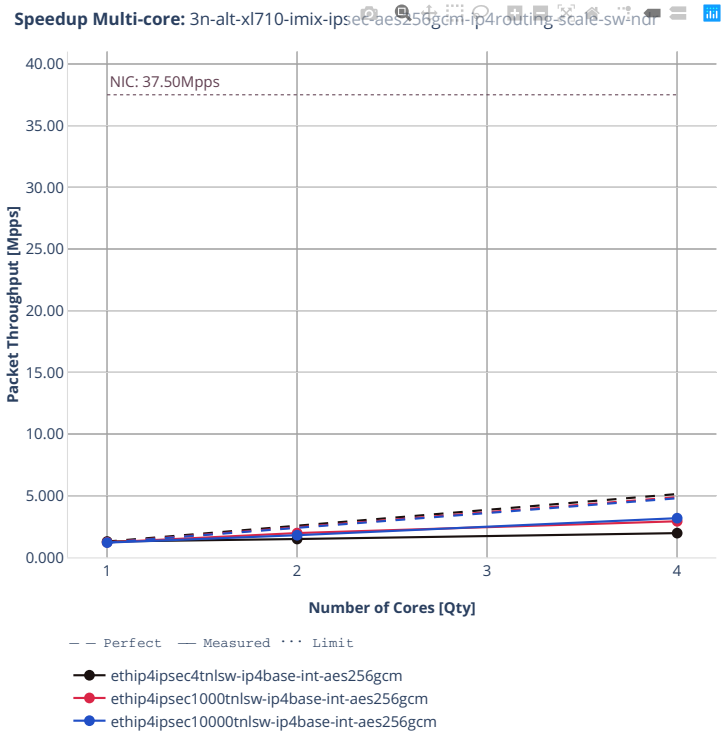

Speedup Multi-core: 3n-alt-xl710-1518b-ipsec-aes256gcm-ip4routing-scale-sw-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes256gcm
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

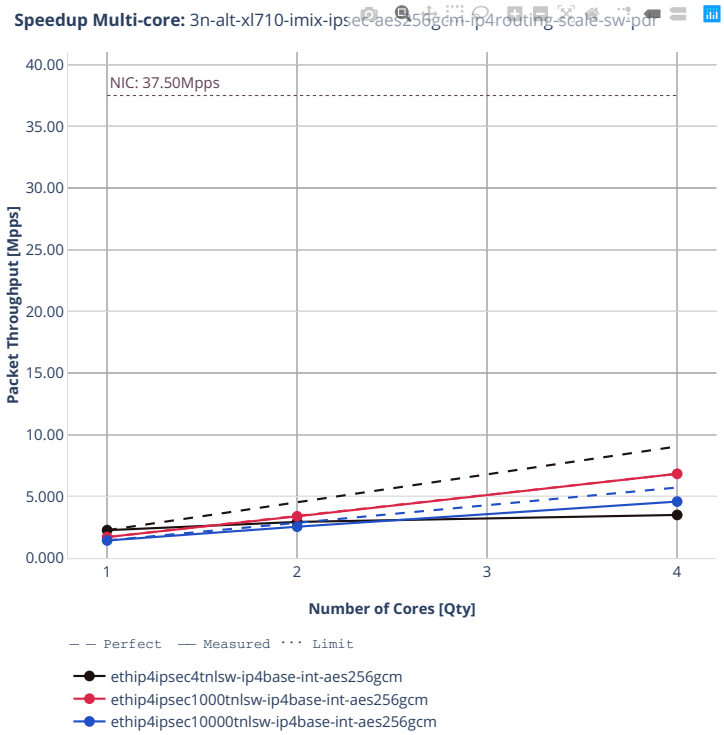
imix-ipsec-aes128cbc-ip4routing-scale-sw



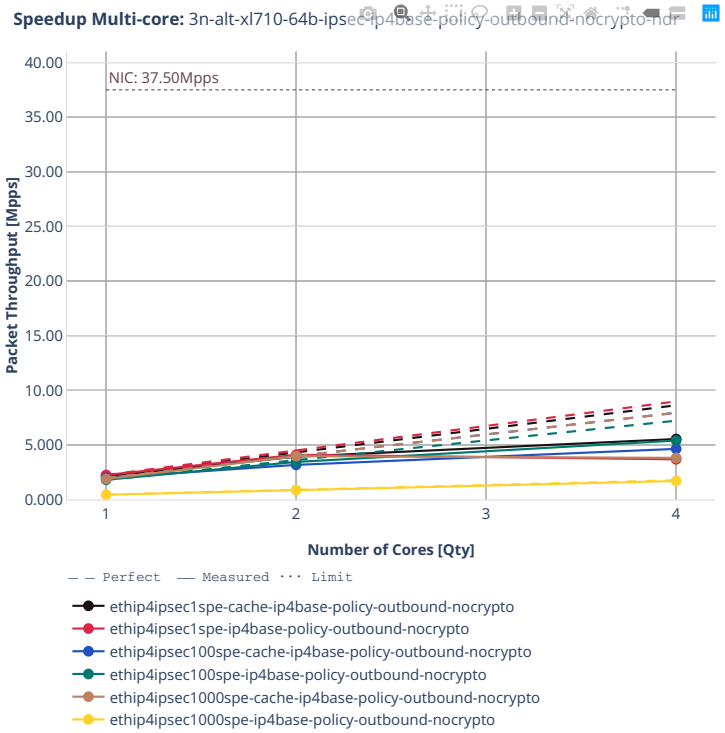

Speedup Multi-core: 3n-alt-xl710-imix-ipsec-aes128cbc-ip4routing-scale-sw-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec4tnlsw-ip4base-int-aes128gcm
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes128gcm
ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec10000tnlsw-ip4base-int-aes128gcm

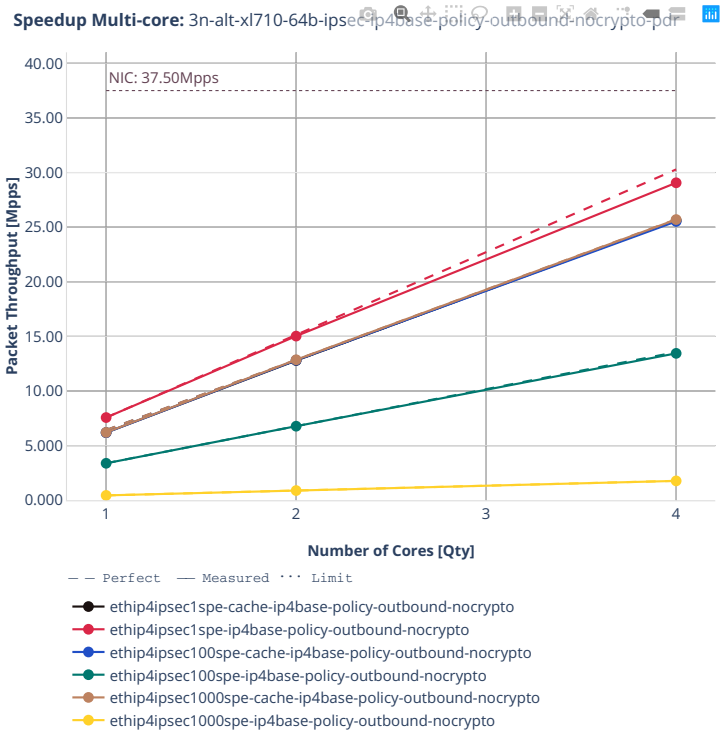
imix-ipsec-aes256gcm-ip4routing-scale-sw



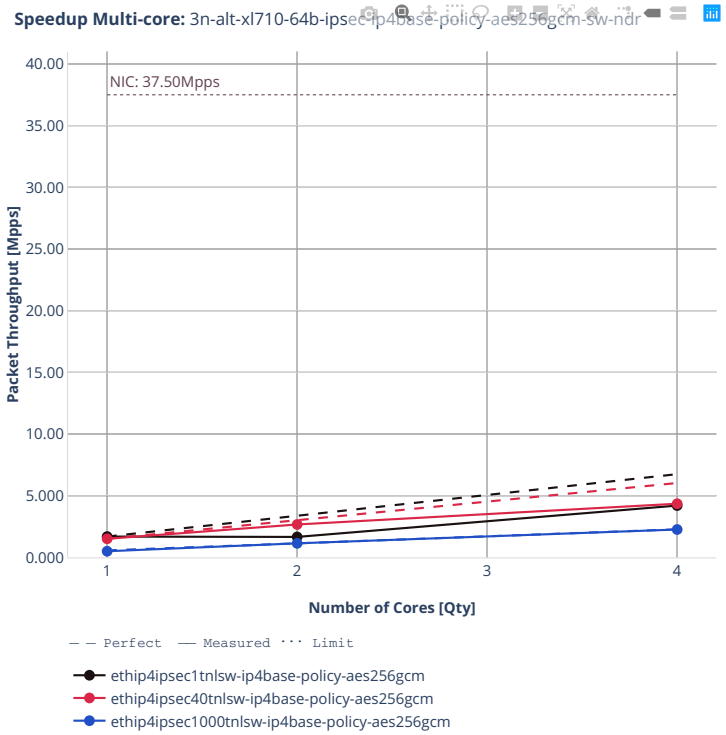

Speedup Multi-core: 3n-alt-xl710-imix-ipsec-aes256gcm-ip4routing-scale-sw-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes256gcm
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
ethip4ipsec10000tnlsw-ip4base-int-aes256gcm

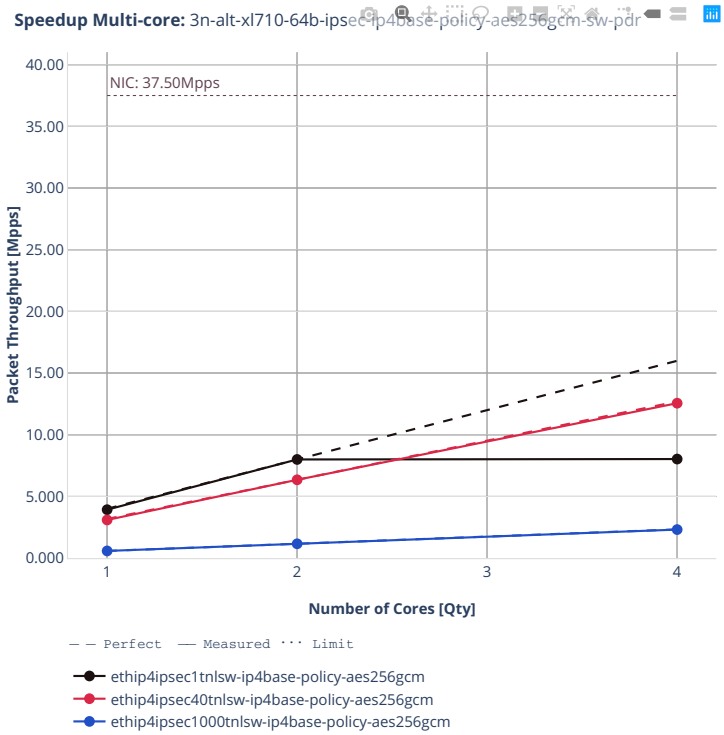
64b-ipsec-ip4base-policy-outbound-nocrypto




Speedup Multi-core: 3n-alt-xl710-64b-ipsec-ip4base-policy-outbound-nocrypto-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec1spe-cache-ip4base-policy-outbound-nocrypto
ethip4ipsec1spe-ip4base-policy-outbound-nocrypto
ethip4ipsec100spe-cache-ip4base-policy-outbound-nocrypto
ethip4ipsec100spe-ip4base-policy-outbound-nocrypto
ethip4ipsec1000spe-cache-ip4base-policy-outbound-nocrypto
ethip4ipsec1000spe-ip4base-policy-outbound-nocrypto

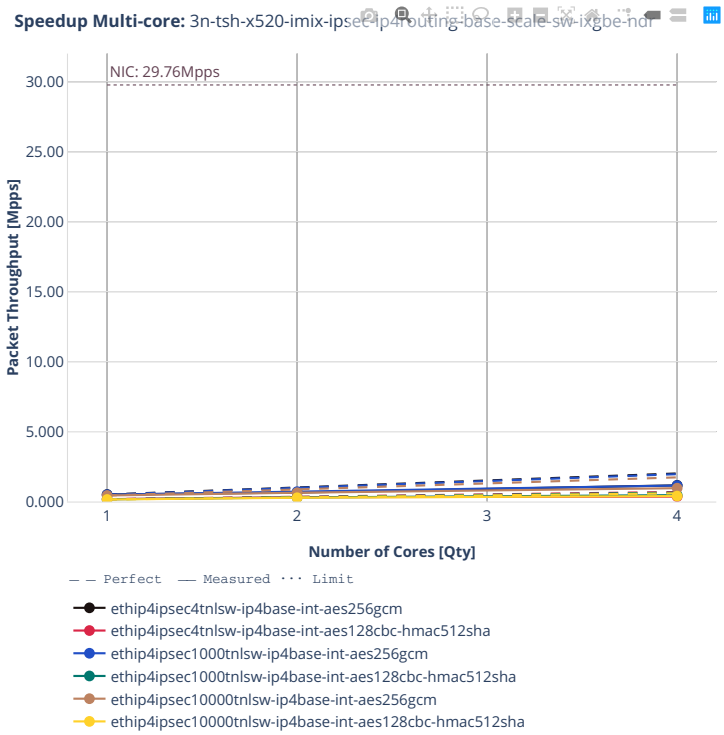
64b-ipsec-ip4base-policy-aes256gcm-sw

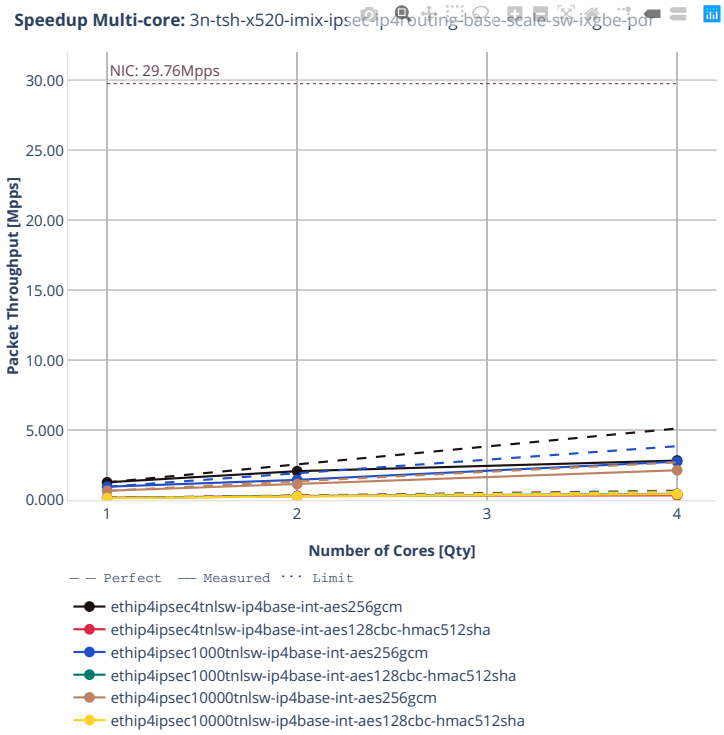



Speedup Multi-core: 3n-alt-xl710-64b-ipsec-ip4base-policy-aes256gcm-sw-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec1tnlsw-ip4base-policy-aes256gcm
ethip4ipsec40tnlsw-ip4base-policy-aes256gcm
ethip4ipsec1000tnlsw-ip4base-policy-aes256gcm

3n-tsh-x520

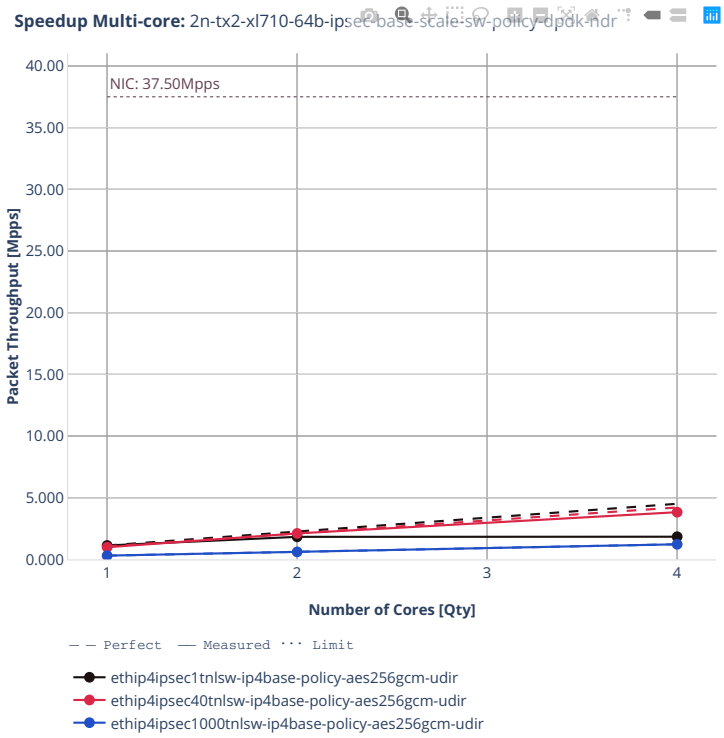
imix-ipsec-ip4routing-base-scale-sw-ixgbe

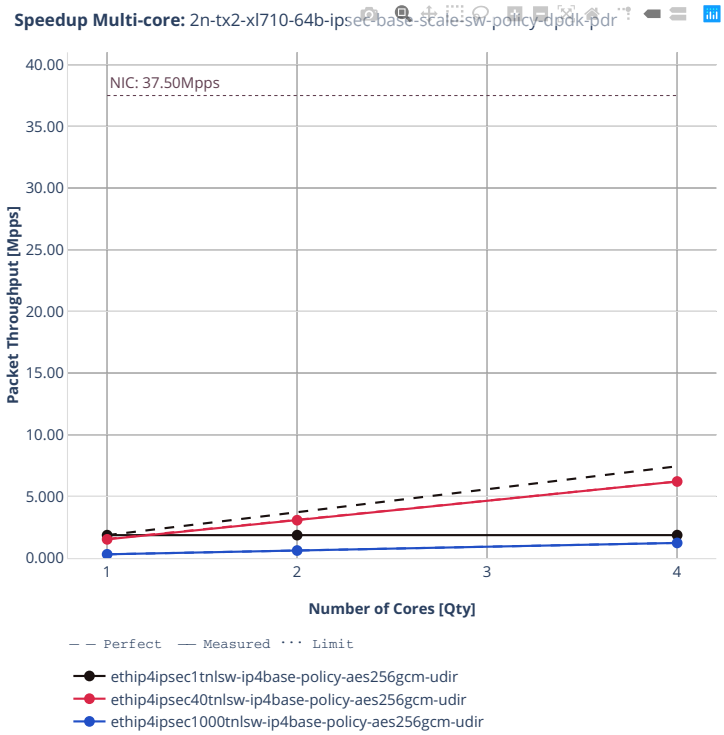



Speedup Multi-core: 3n-tsh-x520-imix-ipsec-ip4routing-base-scale-sw-ixgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes256gcm
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec10000tnlsw-ip4base-int-aes256gcm
ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha

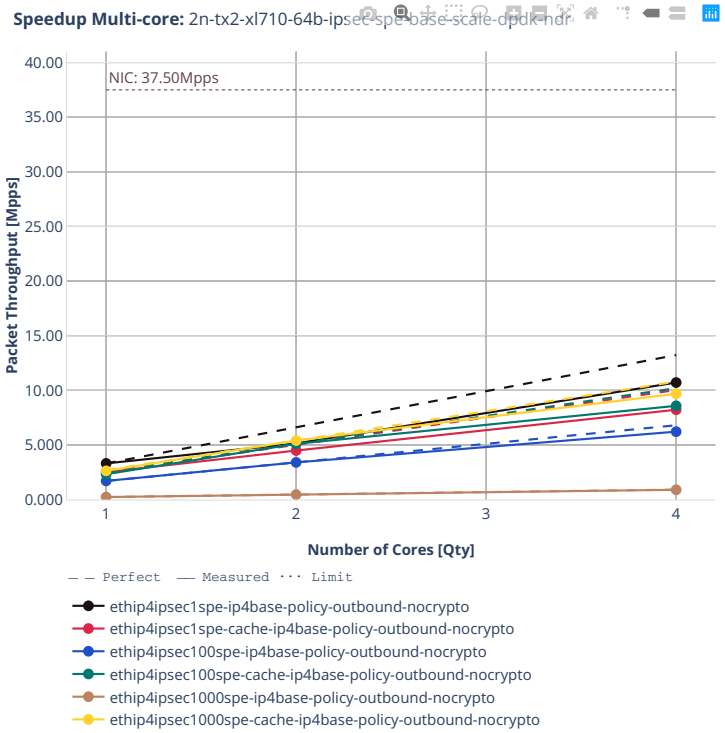
2n-tx2-xl710

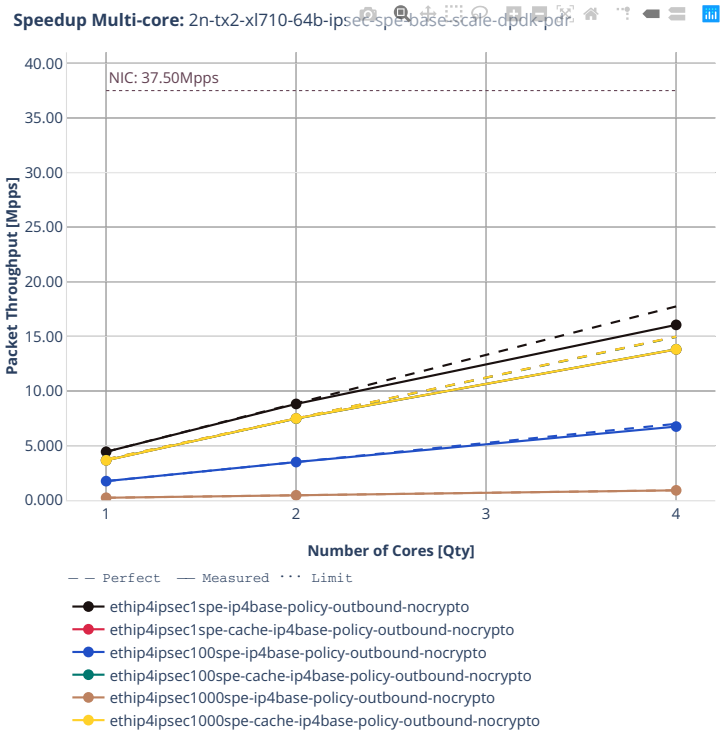
64b-ipsec-spe-ip4routing-base-scale



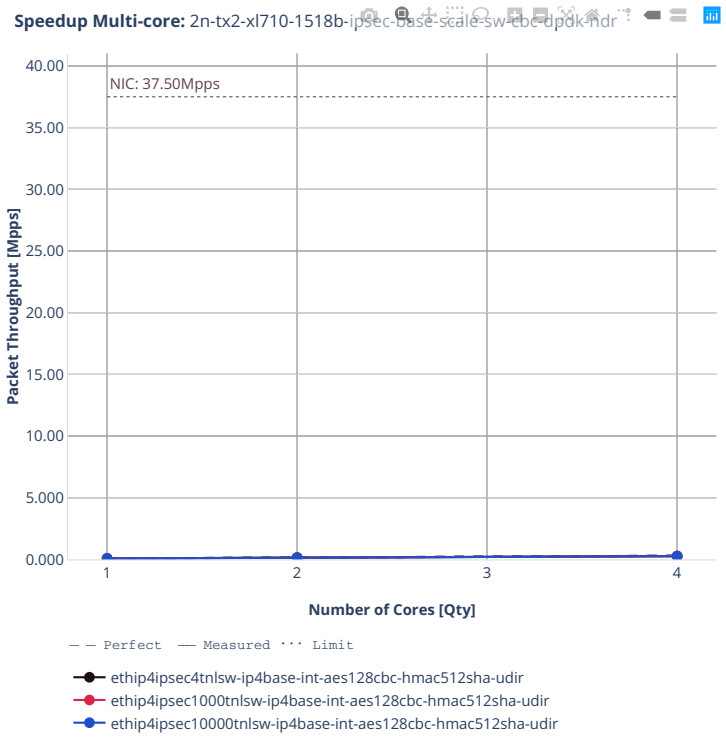

Speedup Multi-core: 2n-tx2-xl710-64b-ipsec-base-scale-sw-policy-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec1tnlsw-ip4base-policy-aes256gcm-udir
ethip4ipsec40tnlsw-ip4base-policy-aes256gcm-udir
ethip4ipsec1000tnlsw-ip4base-policy-aes256gcm-udir

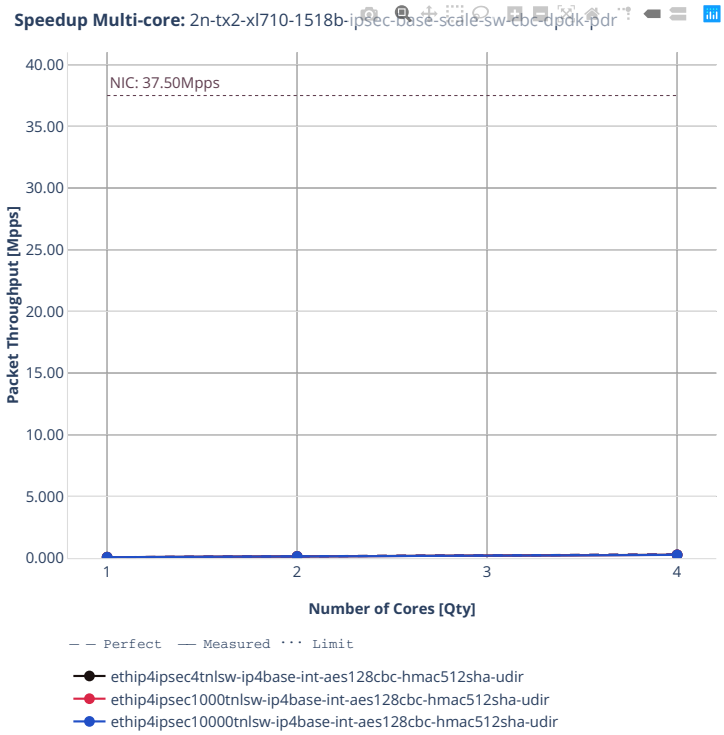
64b-ipsec-ip4routing-base-scale-sw



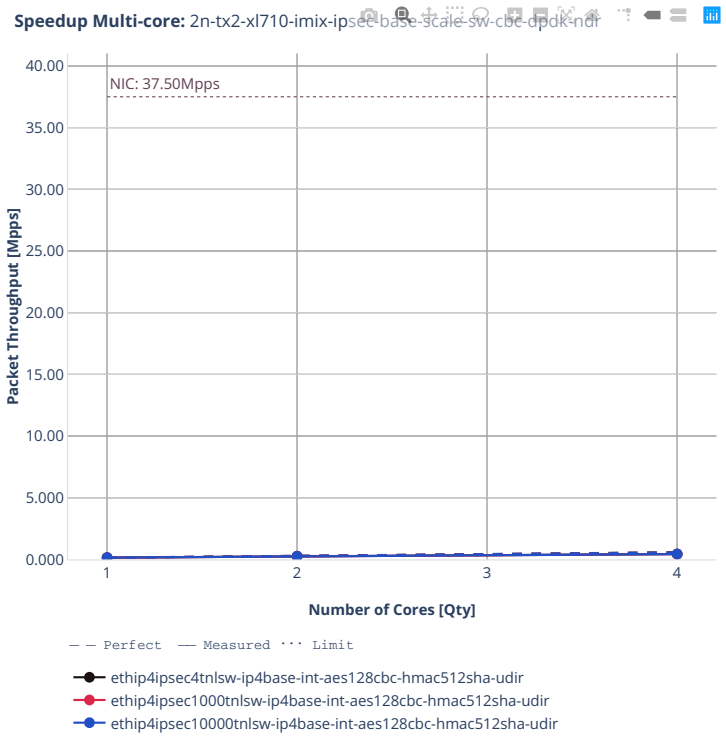

Speedup Multi-core: 2n-tx2-xl710-64b-ipsec-spe-base-scale-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec1spe-ip4base-policy-outbound-nocrypto
ethip4ipsec1spe-cache-ip4base-policy-outbound-nocrypto
ethip4ipsec100spe-ip4base-policy-outbound-nocrypto
ethip4ipsec100spe-cache-ip4base-policy-outbound-nocrypto
ethip4ipsec1000spe-ip4base-policy-outbound-nocrypto
ethip4ipsec1000spe-cache-ip4base-policy-outbound-nocrypto

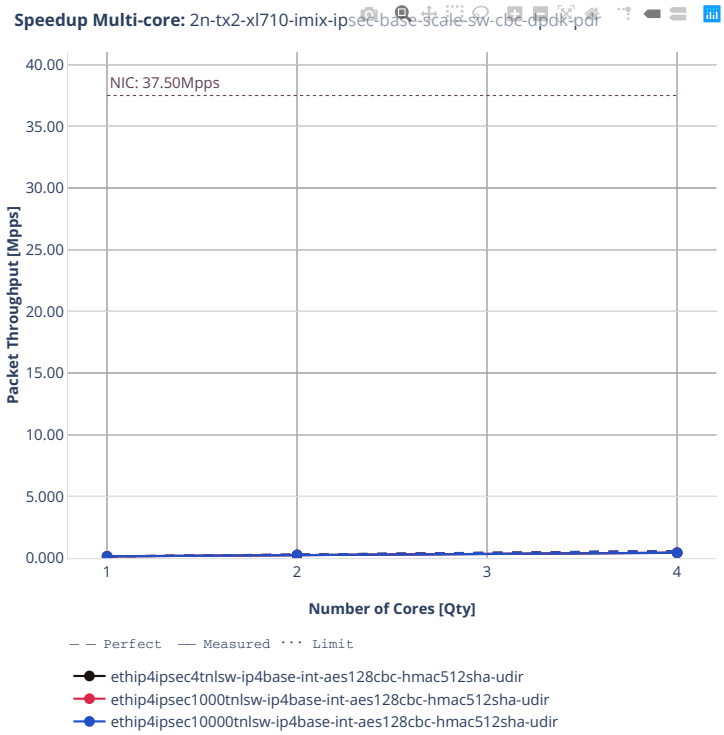
1518b-ipsec-ip4routing-base-scale-sw-cbc



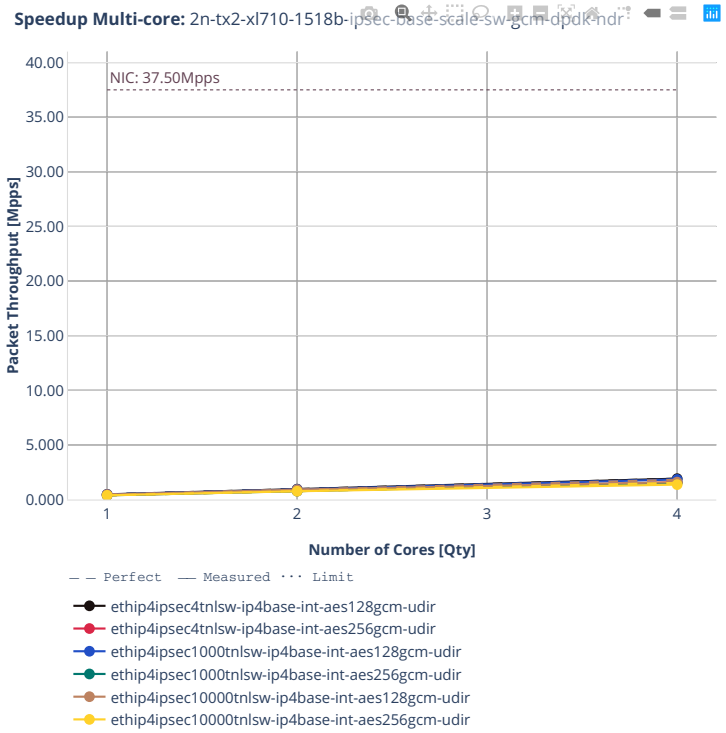

Speedup Multi-core: 2n-tx2-xl710-1518b-ipsec-base-scale-sw-cbc-dpdk-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir

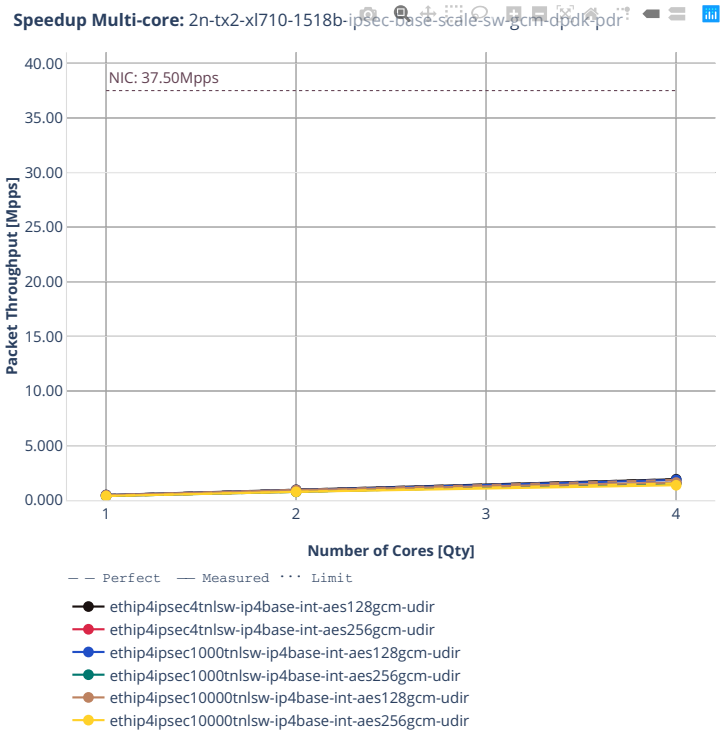
imix-ipsec-ip4routing-base-scale-sw-cbc



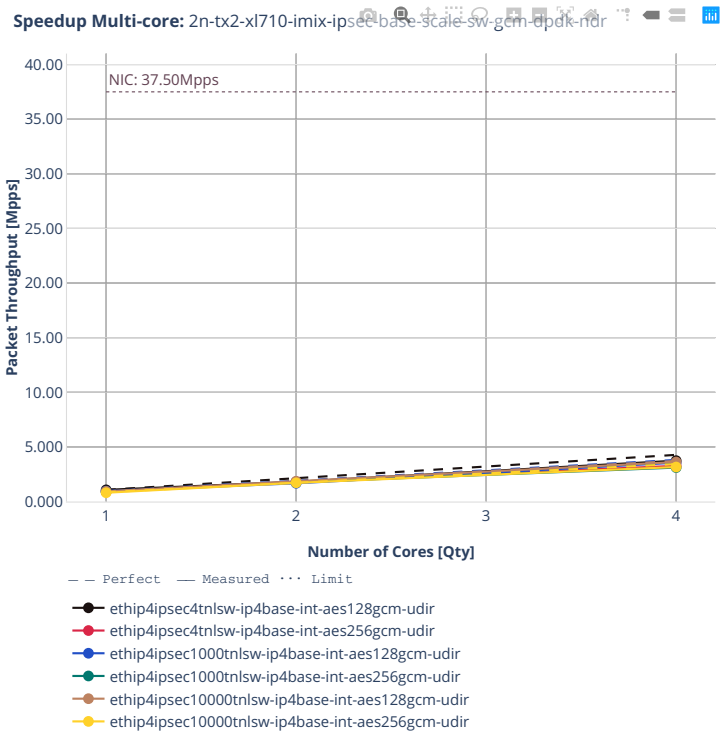

Speedup Multi-core: 2n-tx2-xl710-imix-ipsec-base-scale-sw-cbc-dpdk-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir
ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha-udir

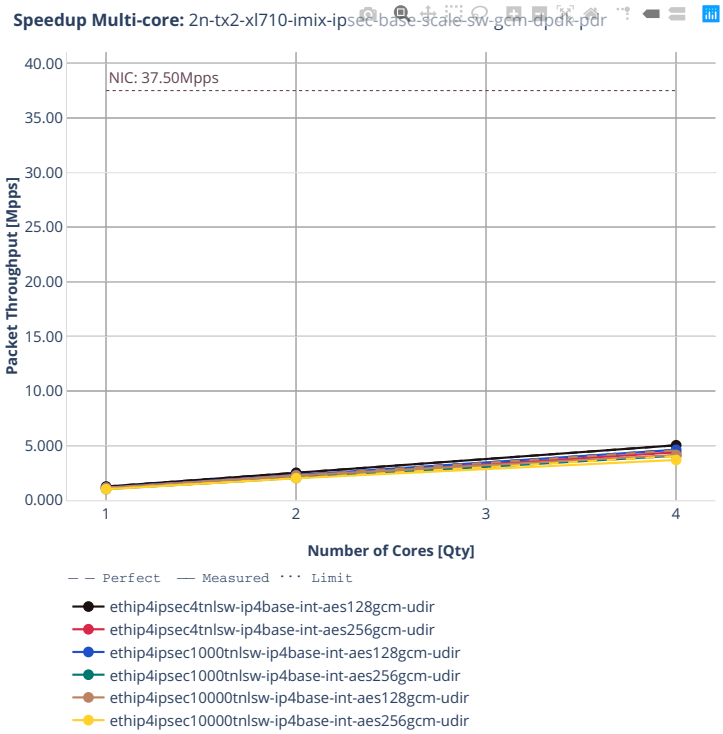
1518b-ipsec-ip4routing-base-scale-sw-gcm




Speedup Multi-core: 2n-tx2-xl710-1518b-ipsec-base-scale-sw-gcm-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128gcm-udir
ethip4ipsec4tnlsw-ip4base-int-aes256gcm-udir
ethip4ipsec1000tnlsw-ip4base-int-aes128gcm-udir
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm-udir
ethip4ipsec10000tnlsw-ip4base-int-aes128gcm-udir
ethip4ipsec10000tnlsw-ip4base-int-aes256gcm-udir

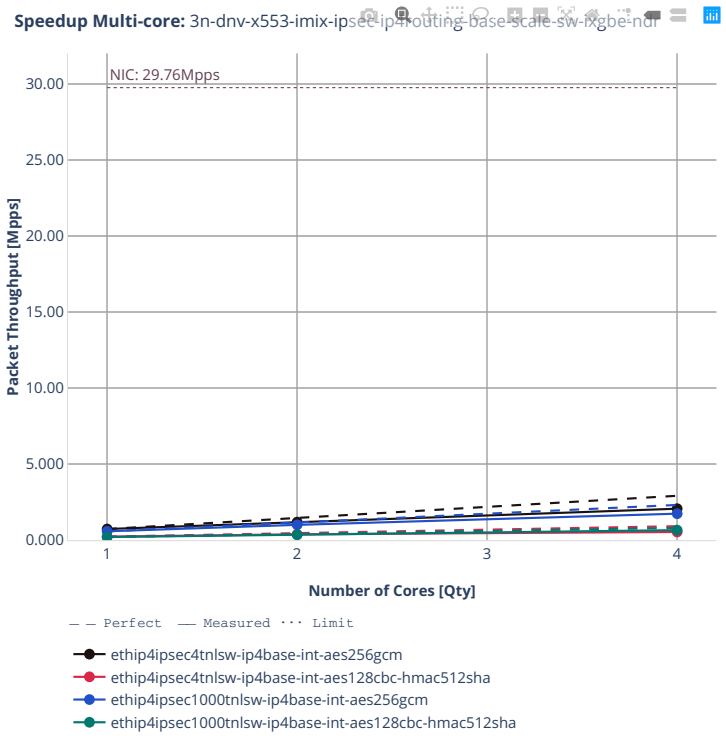
imix-ipsec-ip4routing-base-scale-sw-gcm

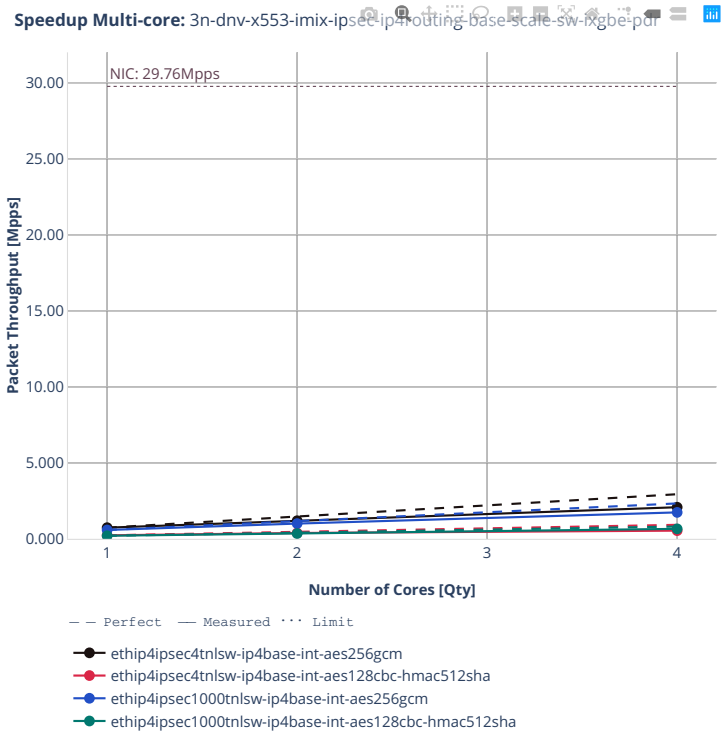



Speedup Multi-core: 2n-tx2-xl710-imix-ipsec-base-scale-sw-gcm-dpdk-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128gcm-udir
ethip4ipsec4tnlsw-ip4base-int-aes256gcm-udir
ethip4ipsec1000tnlsw-ip4base-int-aes128gcm-udir
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm-udir
ethip4ipsec10000tnlsw-ip4base-int-aes128gcm-udir
ethip4ipsec10000tnlsw-ip4base-int-aes256gcm-udir

3n-dnv-x553

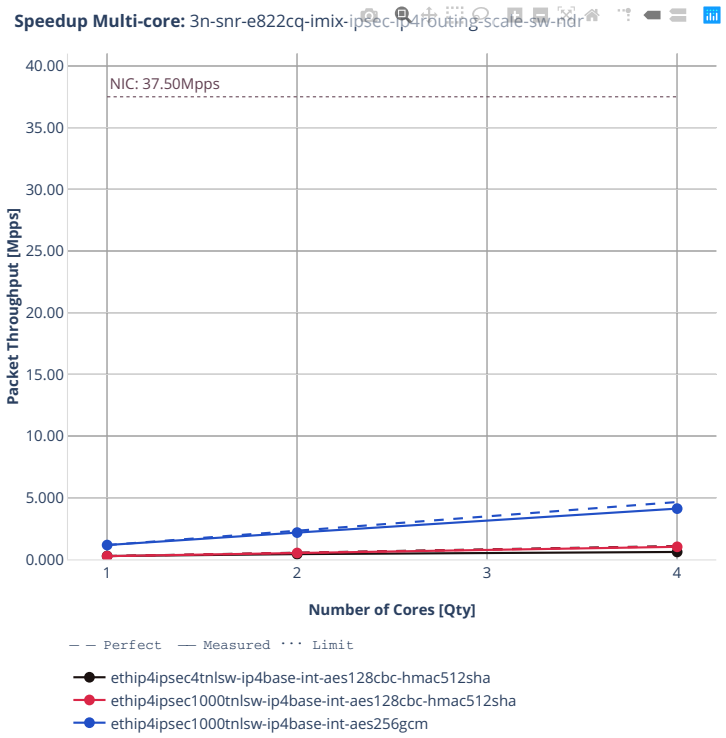
imix-ipsec-ip4routing-base-scale-sw-ixgbe

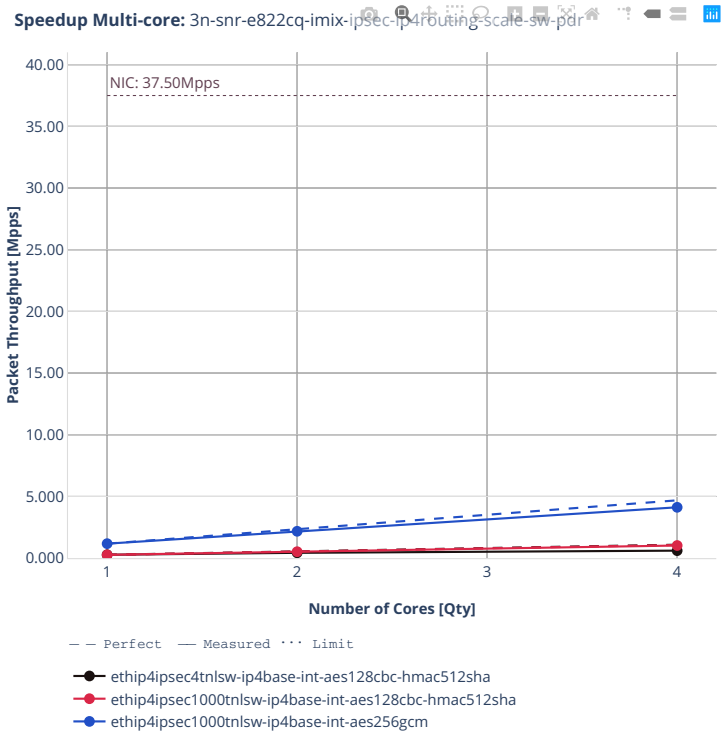



Speedup Multi-core: 3n-dnv-x553-imix-ipsec-ip4routing-base-scale-sw-nxgbe-pdr
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes256gcm
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha

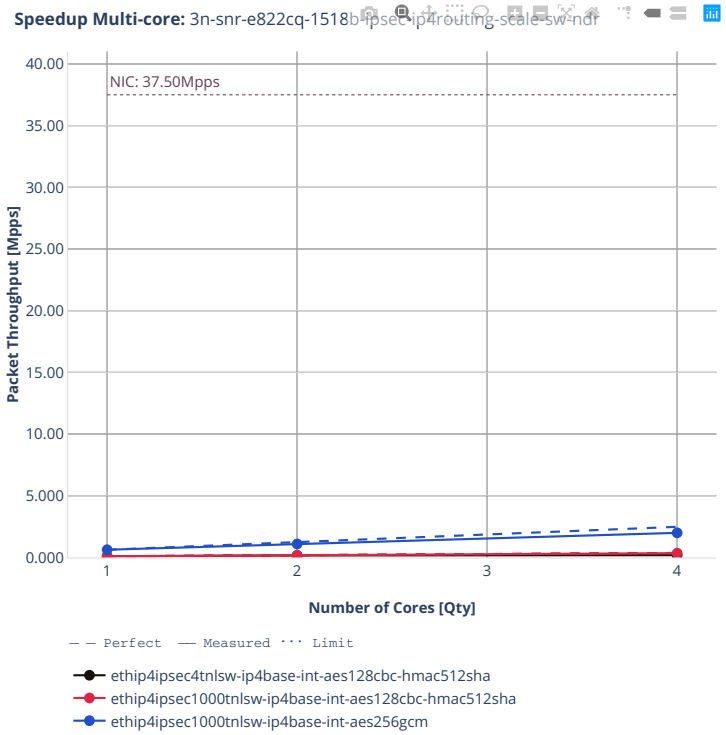
3n-snr-e822cq

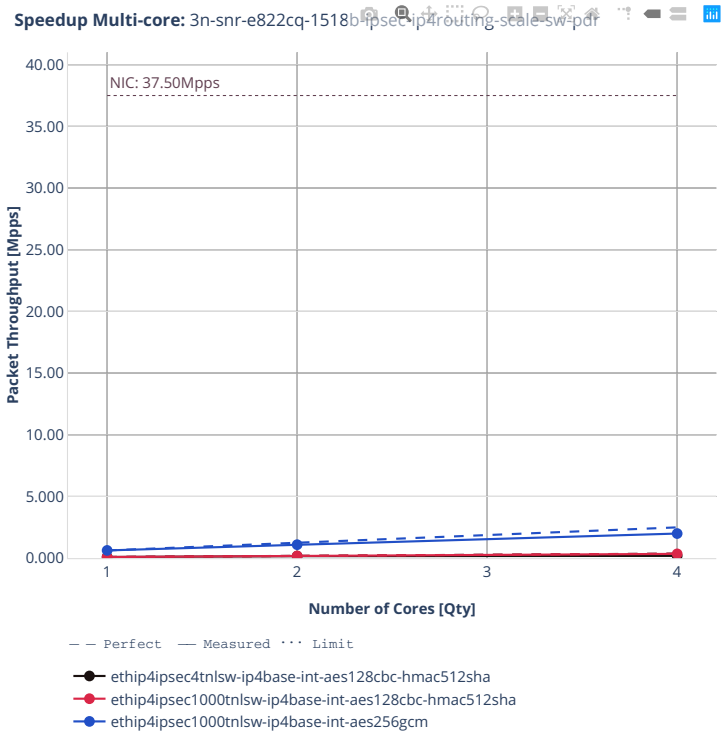
imix-ipsec-ip4routing-scale-sw




Speedup Multi-core: 3n-snr-e822cq-imix-ipsec-ip4routing-scale-sw-pdr
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm

1518b-ipsec-ip4routing-scale-sw




Speedup Multi-core: 3n-snr-e822cq-1518b-ipsec-ip4routing-scale-sw-pdr
NIC: 37.50Mpps
Packet Throughput [Mpps]
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
ethip4ipsec1000tnlsw-ip4base-int-aes256gcm

2.5 Packet Latency

VPP latency results are generated based on the test data obtained from CSIT-2210 NDR-PDR throughput tests executed across physical testbeds hosted in LF FD.io labs: 2n-icx, 3n-icx, 2n-aws, 2n-clx, 2n-zn2, 3n-alt, 3n-tsh, 2n-tx2.

Latency by percentile distribution plots are used to show packet latency percentiles at different packet rate load levels: i) No-Load latency streams only, ii) Low-Load at 10% PDR, iii) Mid-Load at 50% PDR and iv) High-Load at 90% PDR.

For more details, see *Packet Latency* (page 43).

Additional information about graph data:

1. **Graph Title**: describes tested DUT packet path.
2. **X-axis Labels**: percentile of packets.
3. **Y-axis Labels**: measured one-way packet latency values in [uSec].
4. **Graph Legend**: list of latency tests at different packet rate load level.
5. **Hover Information**: packet rate load level, stream direction (East-West, West-East), percentile, one-way latency.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[142], build logs from FD.io vpp performance job 3n-icx[143], build logs from FD.io vpp performance job 2n-aws[144], build logs from FD.io vpp performance job 2n-clx[145], build logs from FD.io vpp performance job 2n-zn2[146], build logs from FD.io vpp performance job 3n-alt[147], build logs from FD.io vpp performance job 3n-tsh[148] and build logs from FD.io vpp performance job 2n-tx2[149] with RF result files csit-vpp-perf-2210-*.zip archived here.

---

[142] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[143] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-icx
[144] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-aws
[145] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx
[146] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-zn2
[147] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-alt
[148] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-tsh
[149] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-tx2

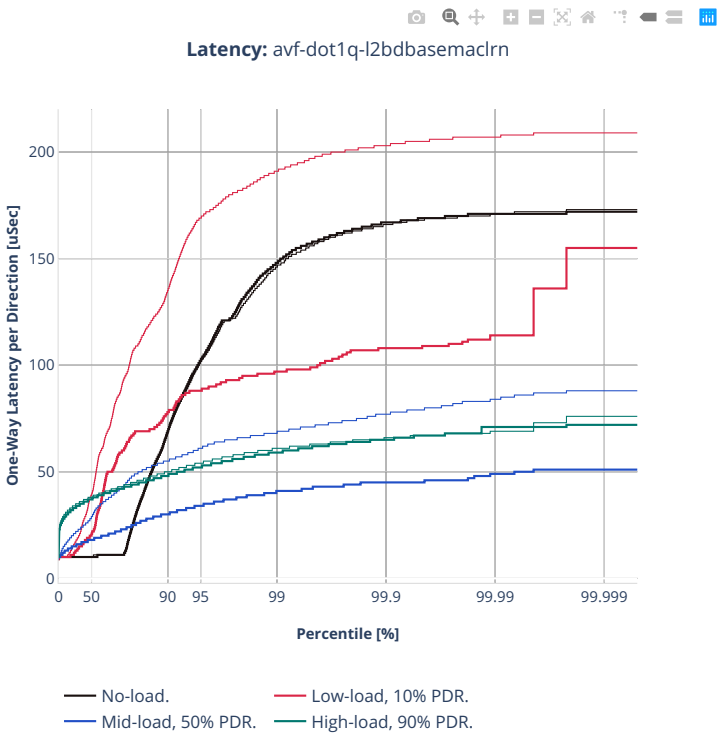
2.5.1 L2 Ethernet Switching

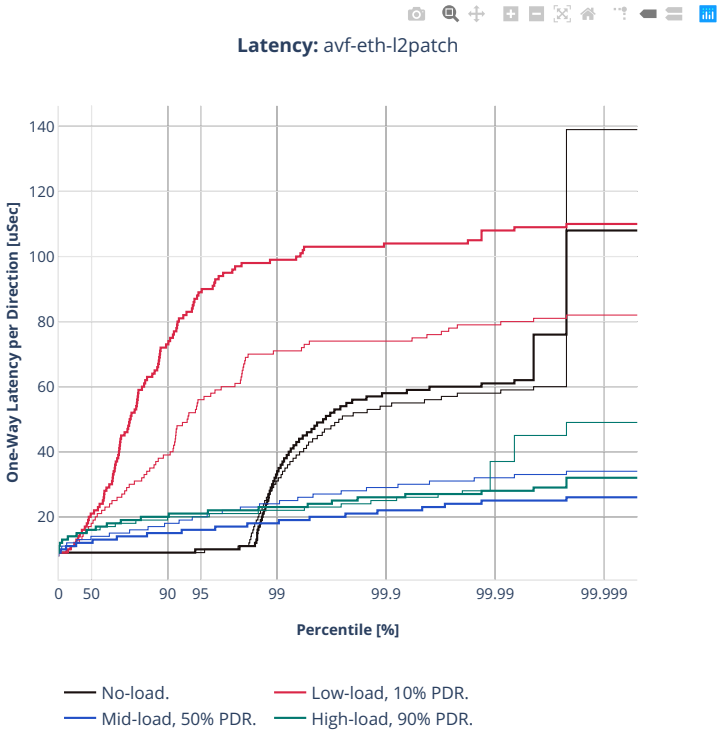
CSIT source code for the test cases used for plots can be found in CSIT git repository[150].

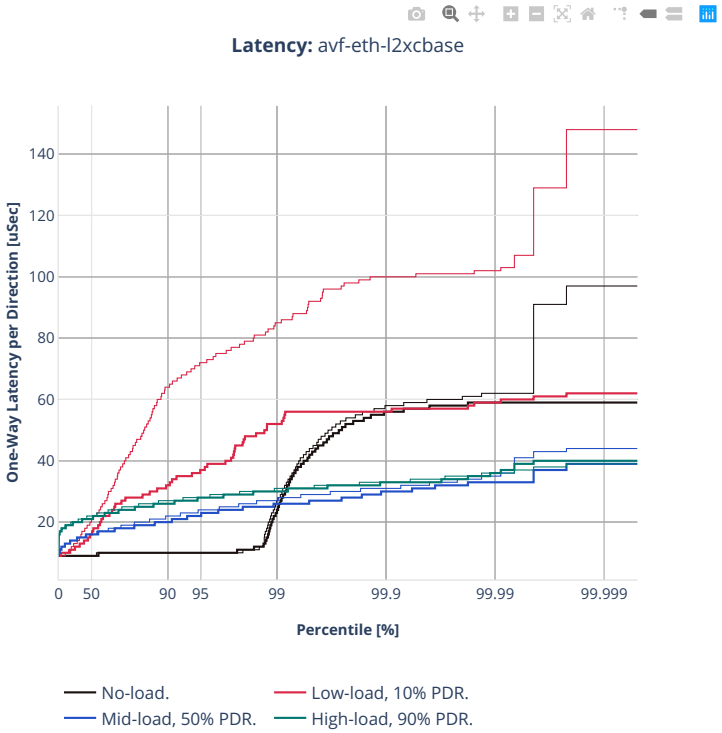
[150] https://git.fd.io/csit/tree/tests/vpp/perf/l2?h=rls2210

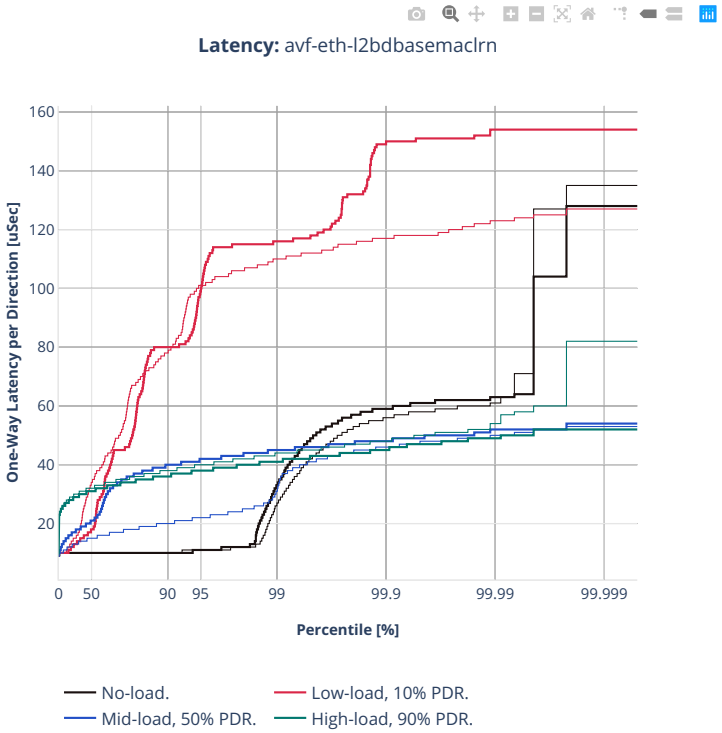
2n-icx-xxv710

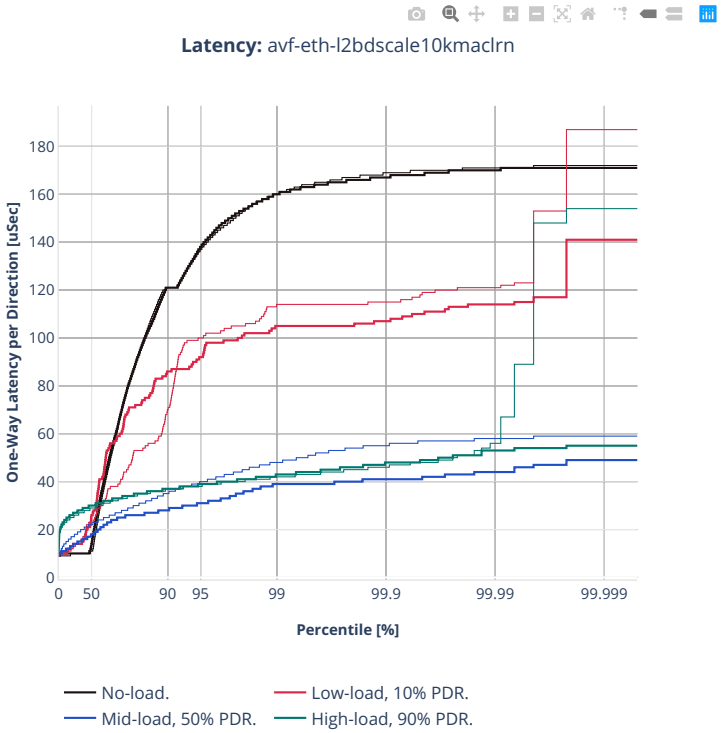
64b-2t1c-l2switching-base-scale-avf

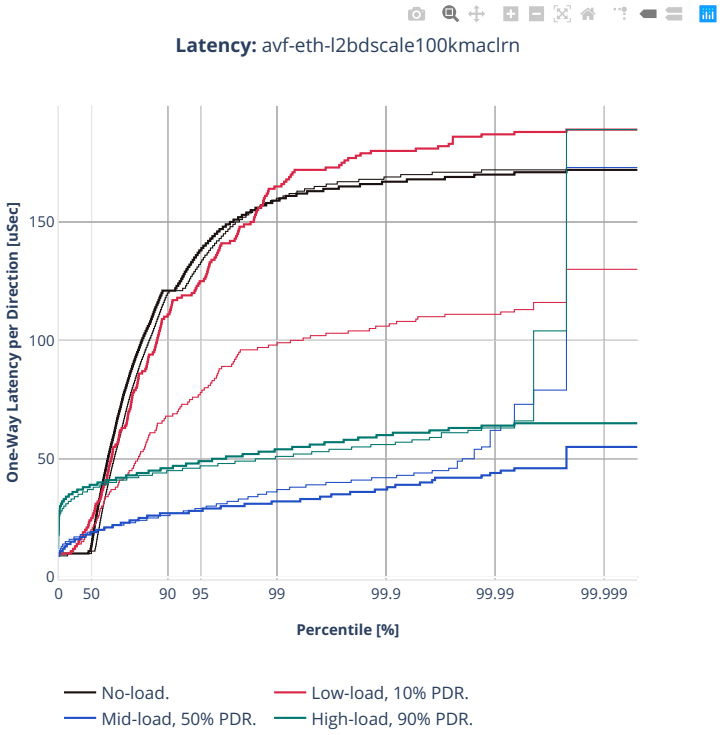


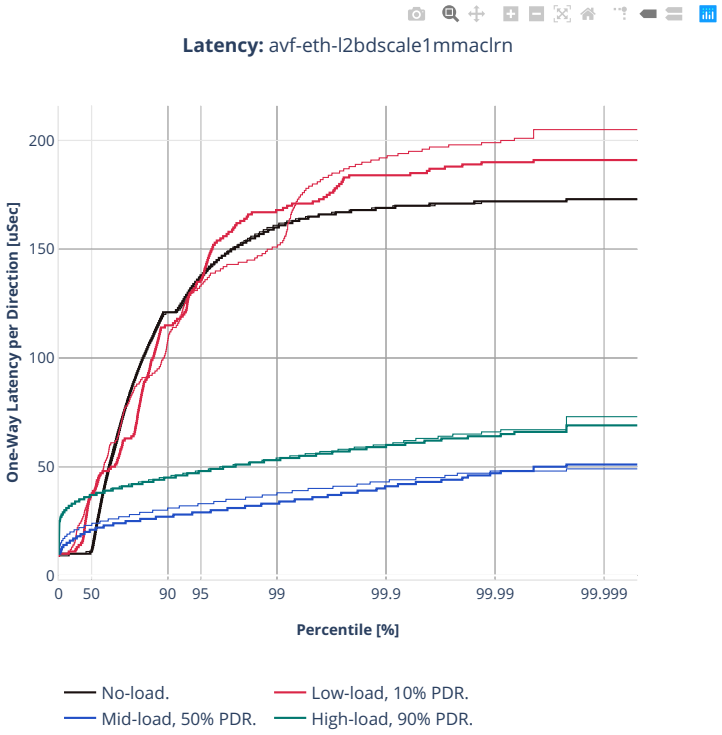

Latency: avf-eth-l2patch
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-eth-l2xcbase
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

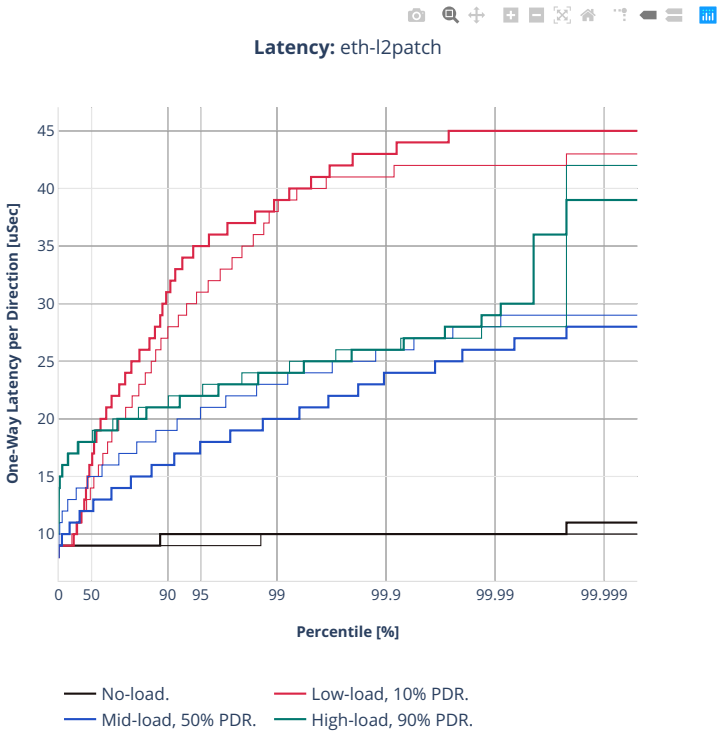

Latency: avf-eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

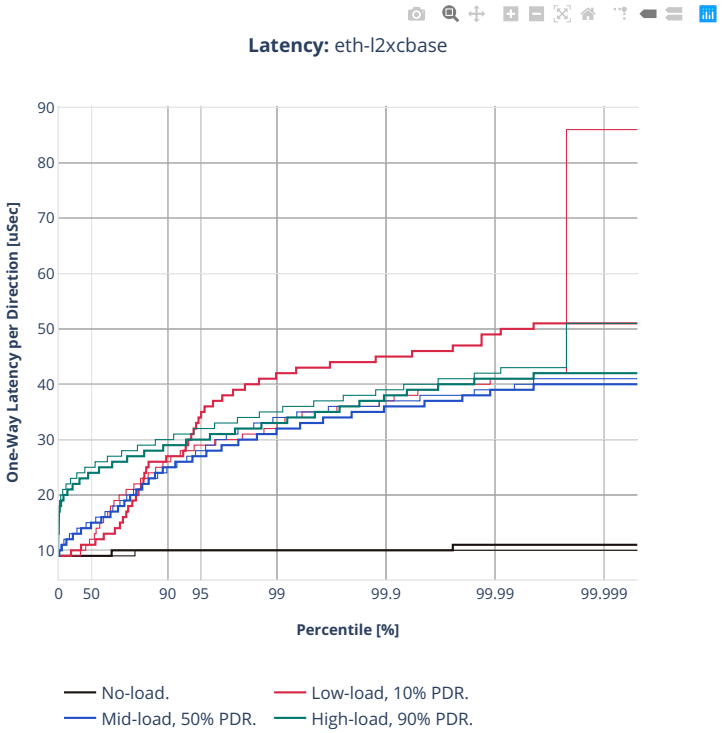

Latency: avf-eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

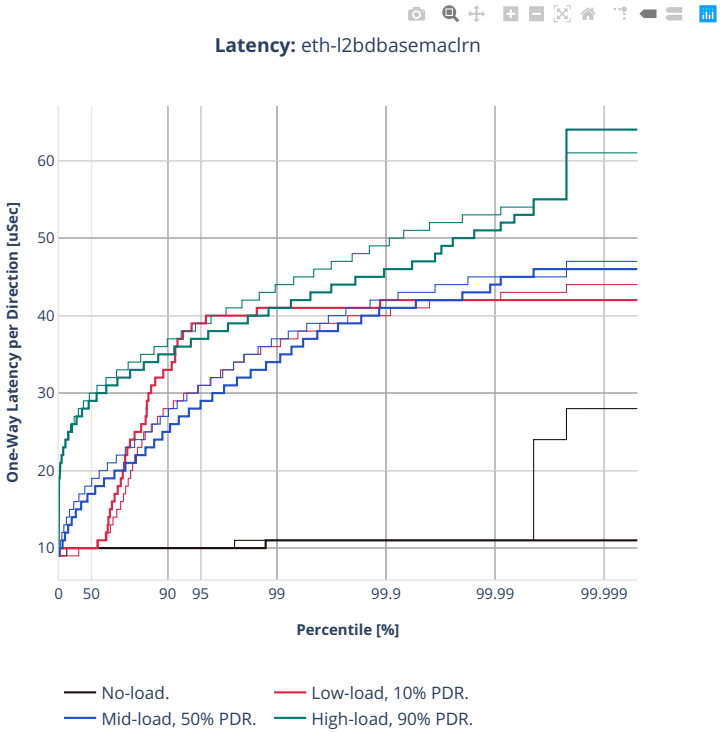

Latency: avf-eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

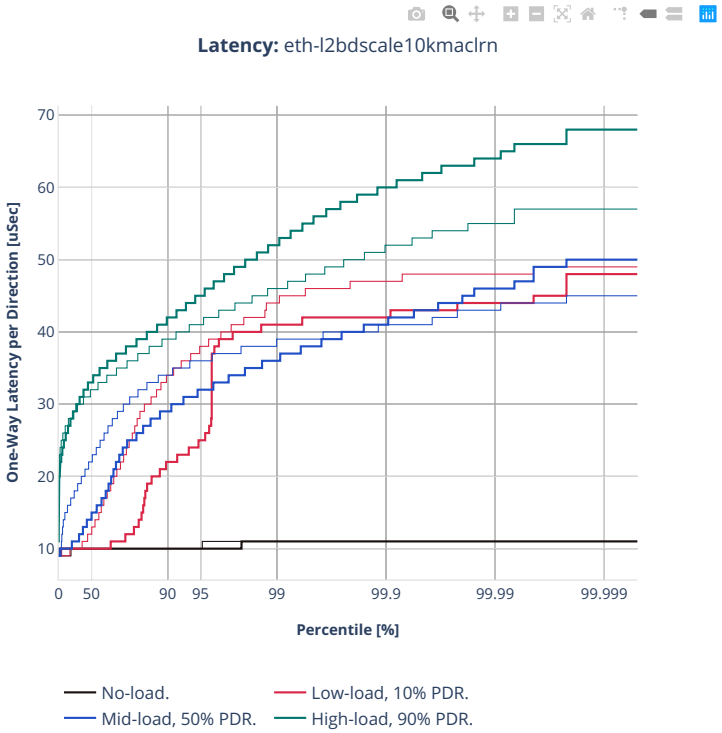

Latency: avf-eth-l2bdscale1mmaclrn
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

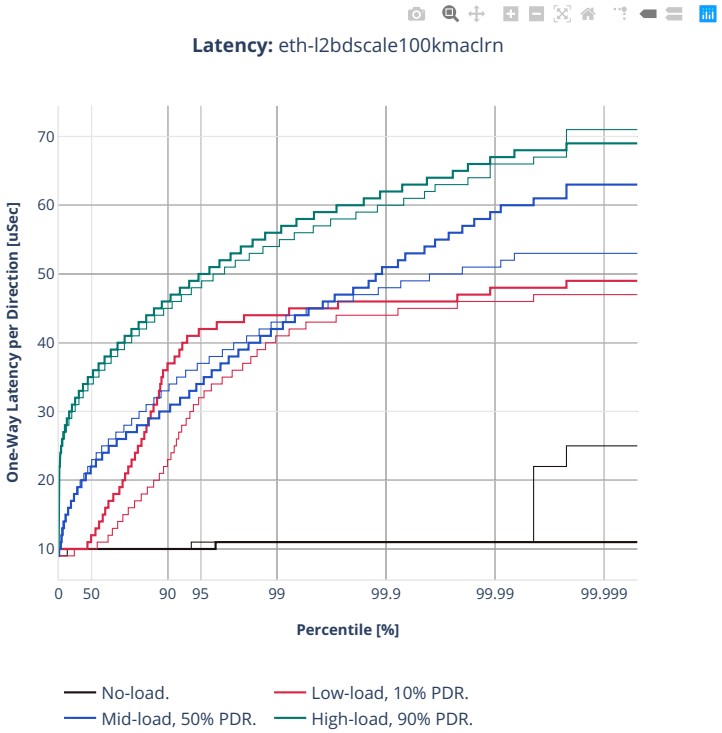
64b-2t1c-l2switching-base-scale-dpdk

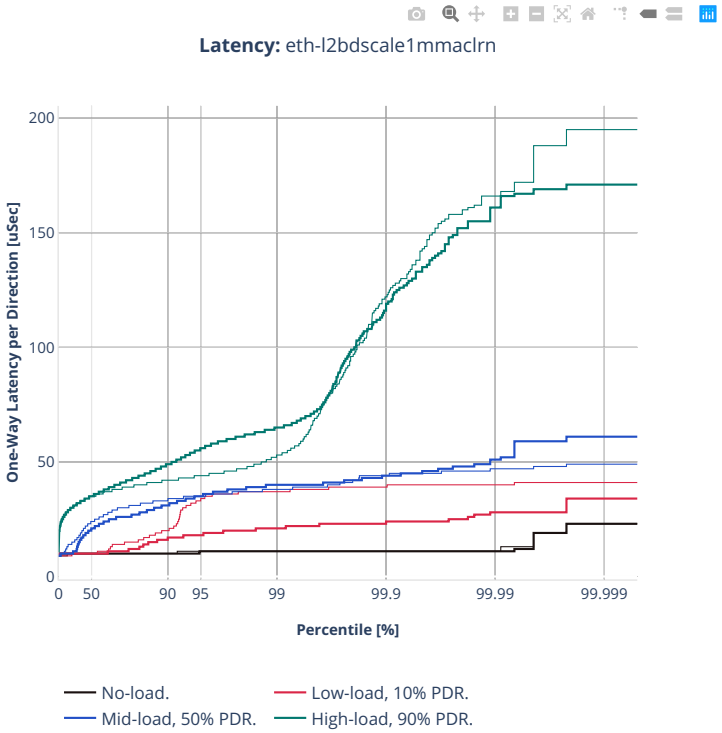



Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

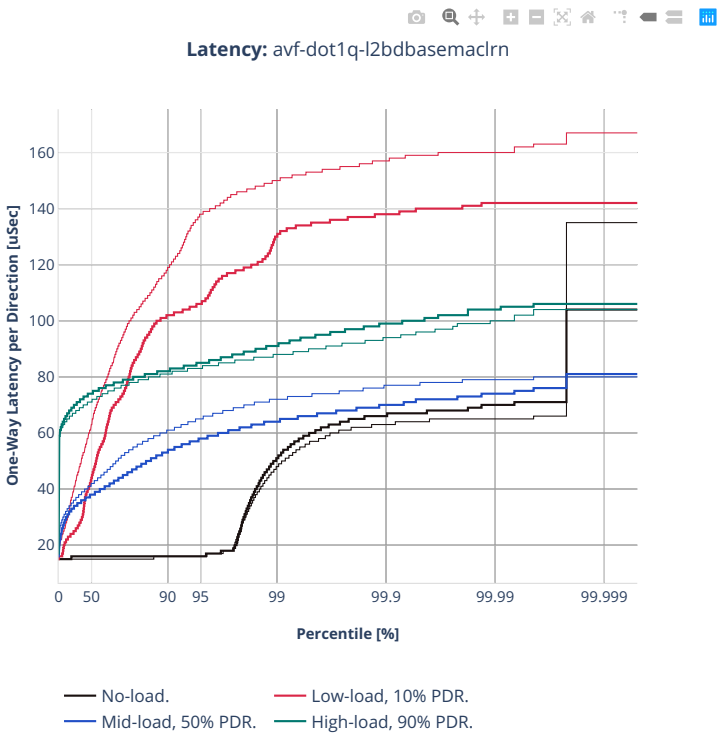

Latency: eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

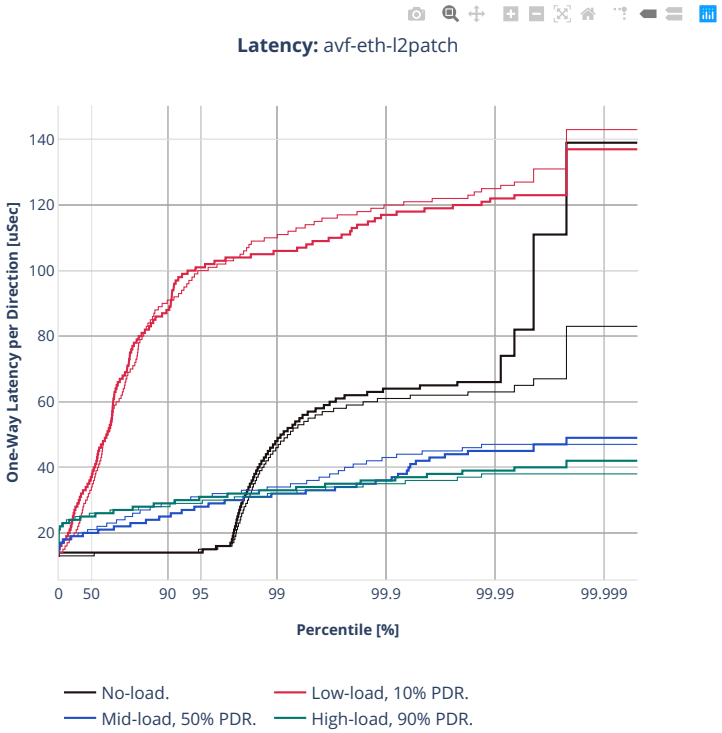

Latency: eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

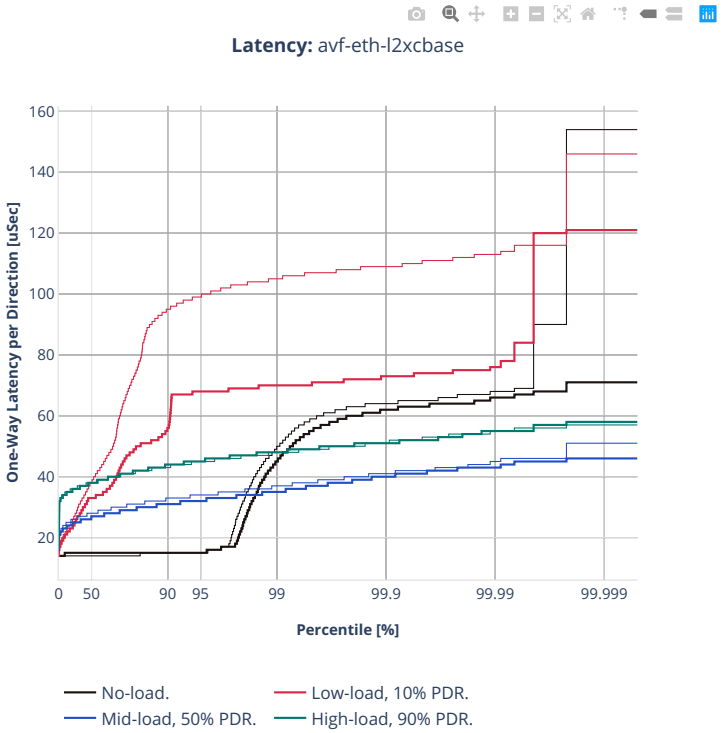

Latency: eth-l2bdscale1mmaclrn
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

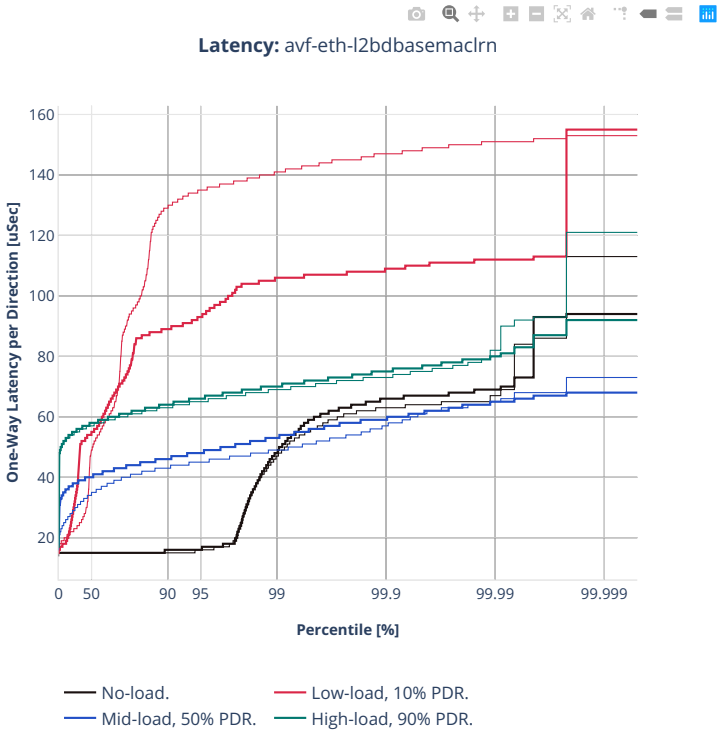
3n-icx-xxv710

64b-2t1c-l2switching-base-avf

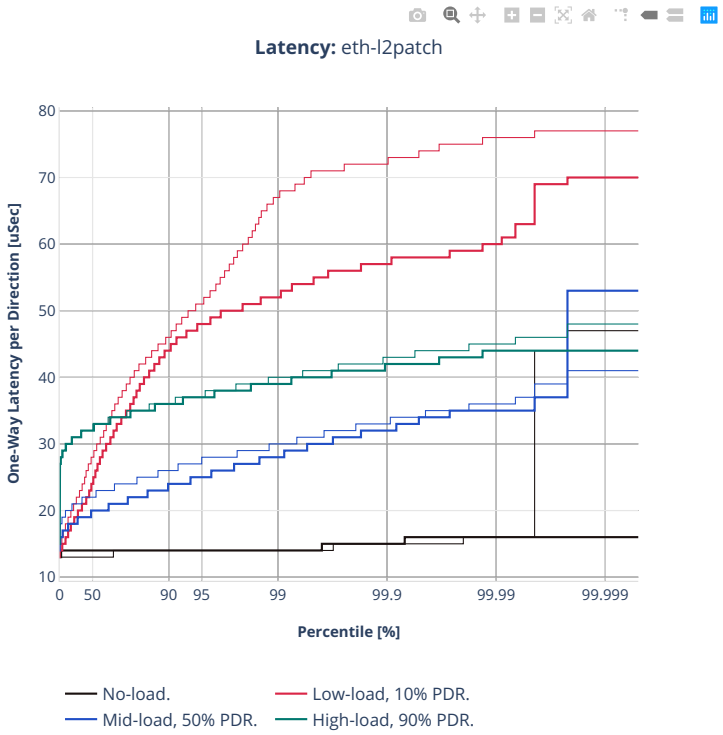


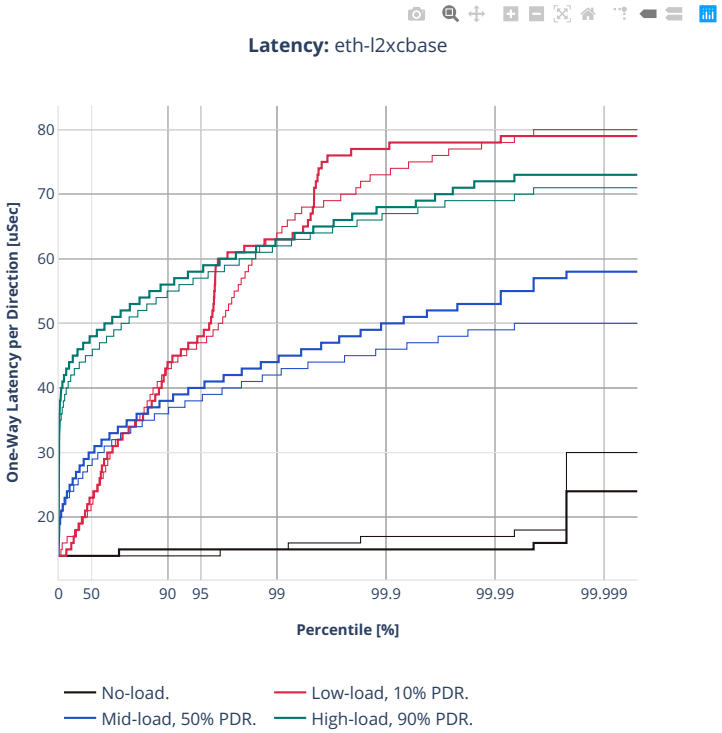

Latency: avf-eth-l2patch
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

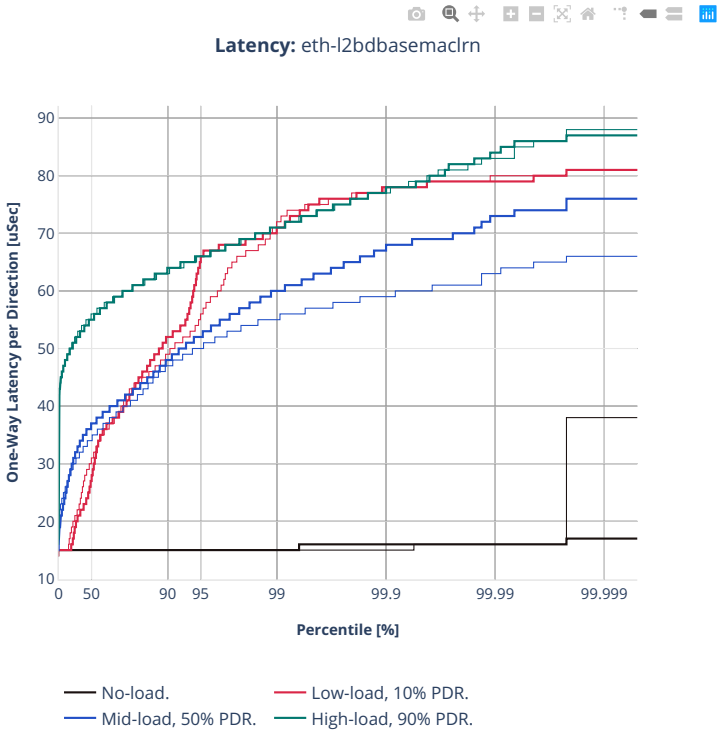

Latency: avf-eth-l2xcbase
One-Way Latency per Direction [uSec]
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-l2switching-base-dpdk

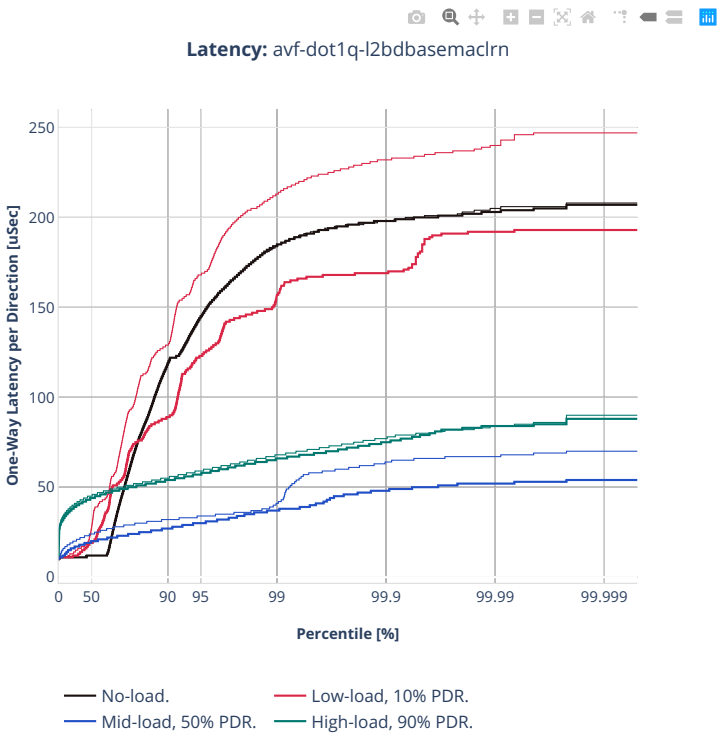


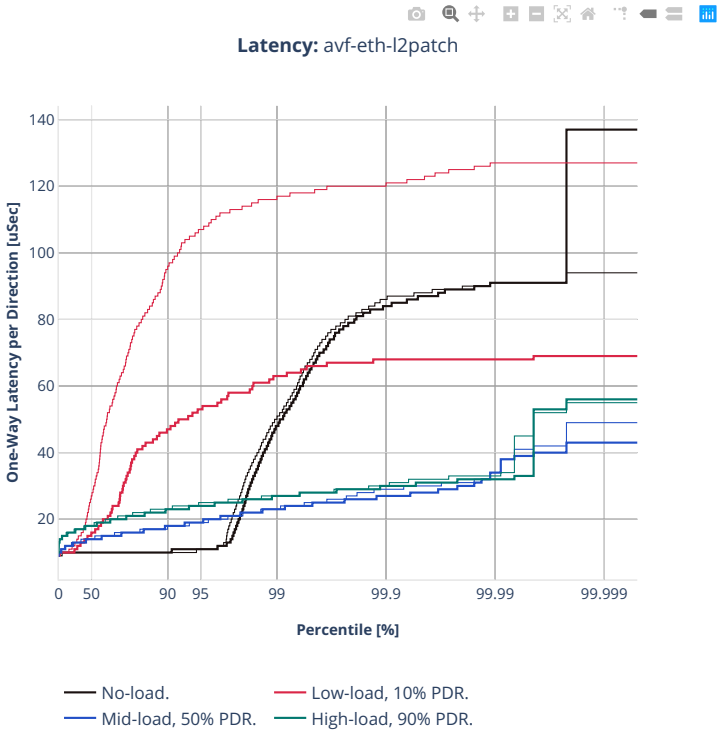

Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

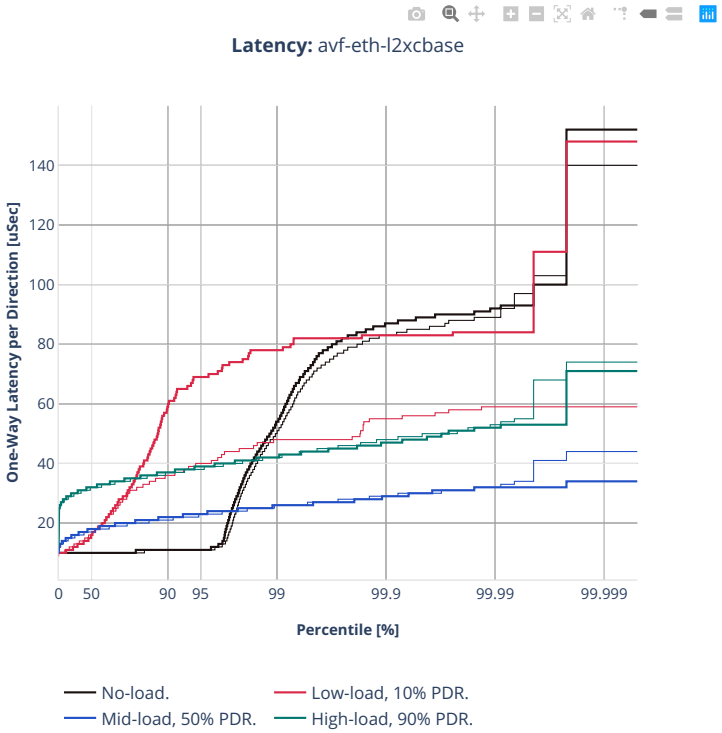

Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

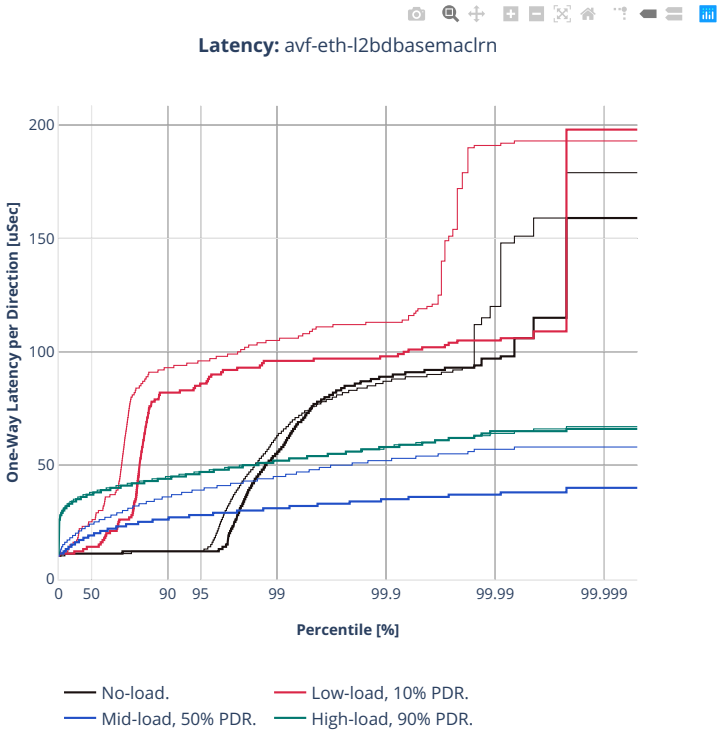
2n-clx-xxv710

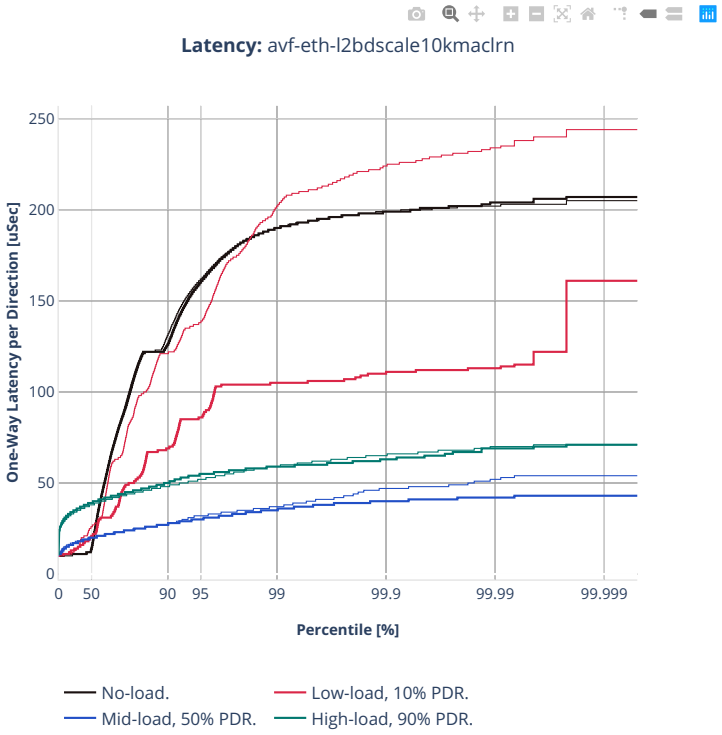
64b-2t1c-l2switching-base-scale-avf

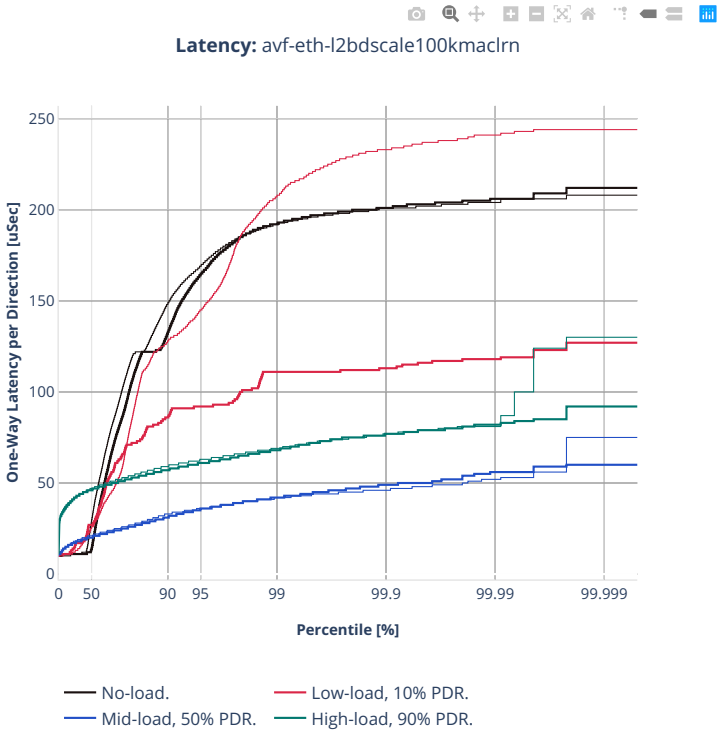



Latency: avf-eth-l2patch
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

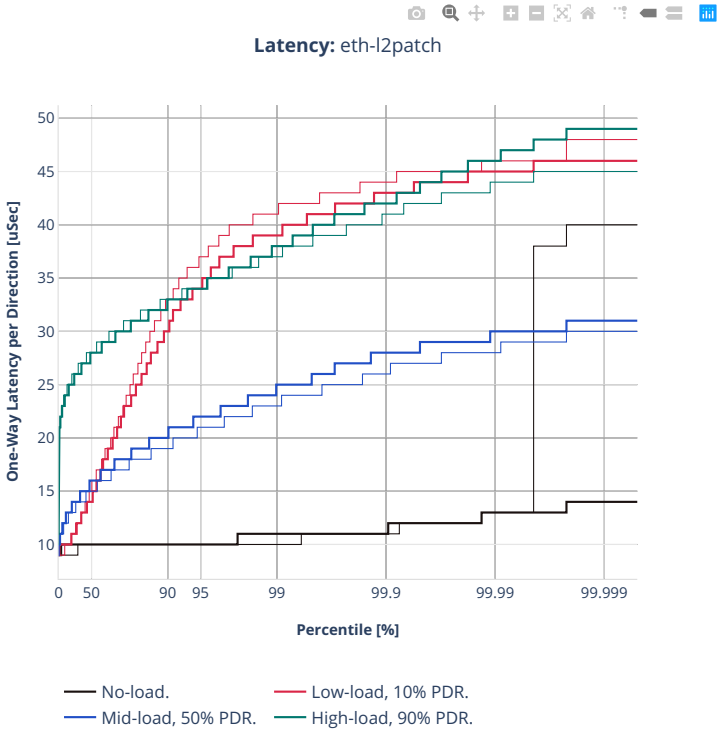

Latency: avf-eth-l2xcbase
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

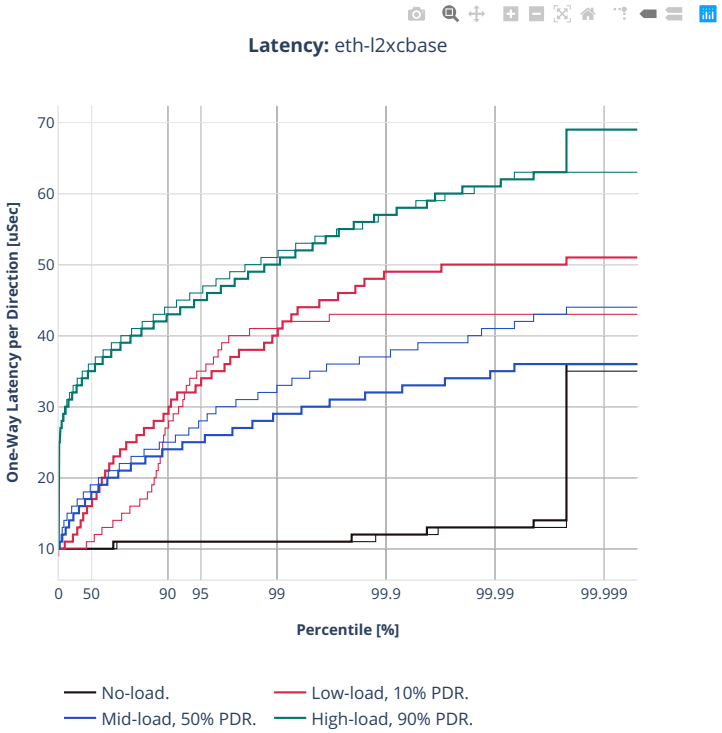

Latency: avf-eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

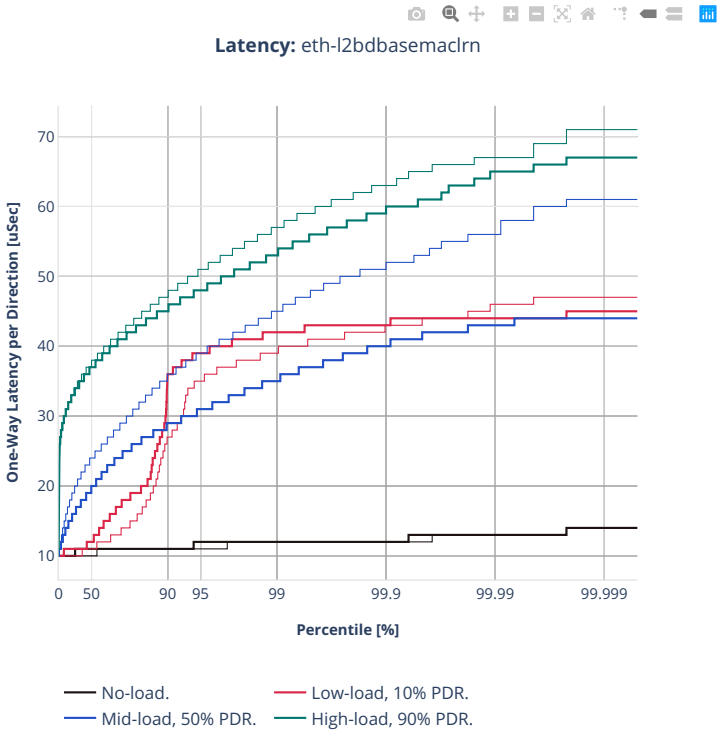

Latency: avf-eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

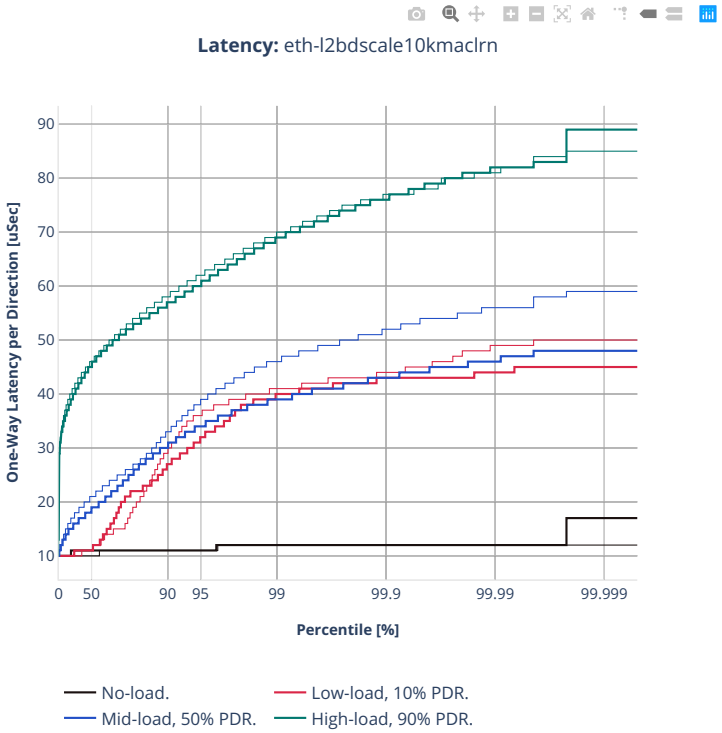

Latency: avf-eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

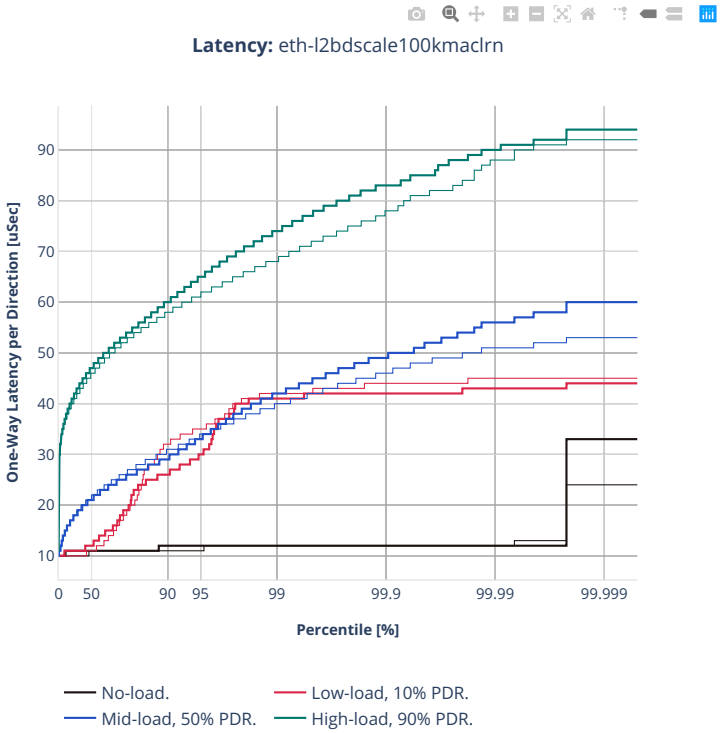
64b-2t1c-l2switching-base-scale-dpdk

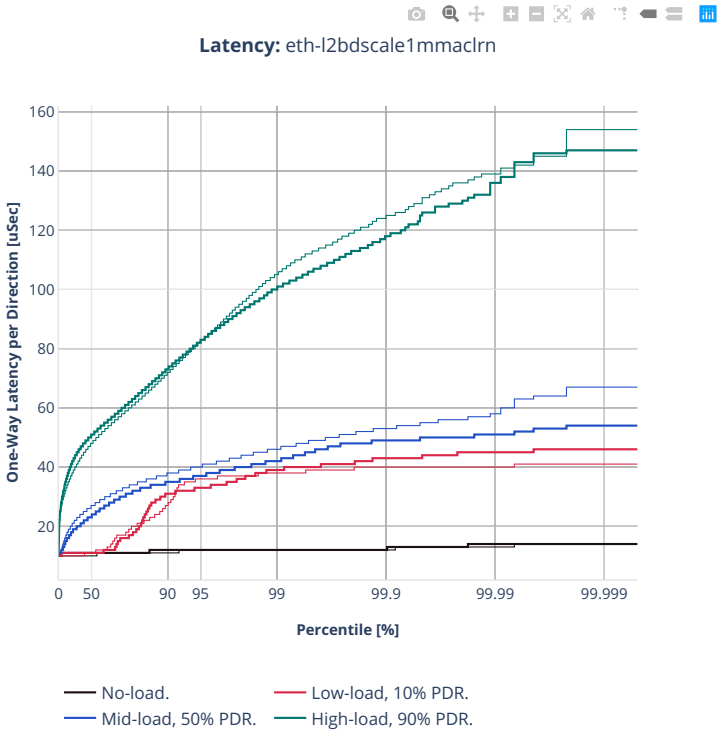



Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

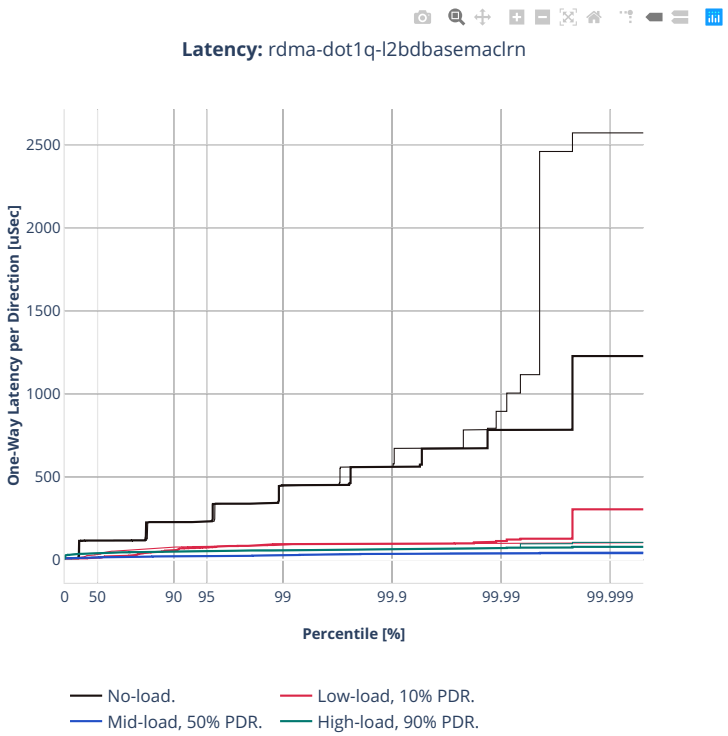

Latency: eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

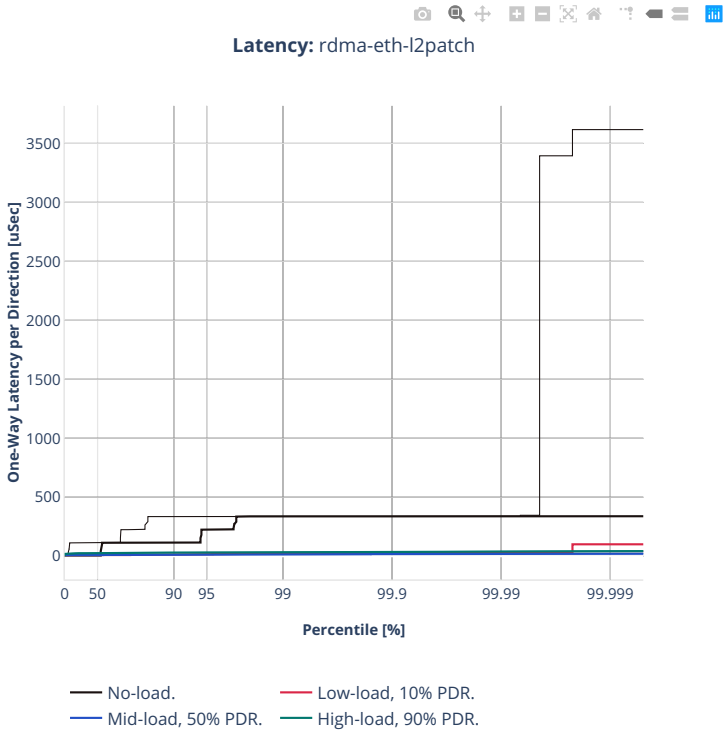

Latency: eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

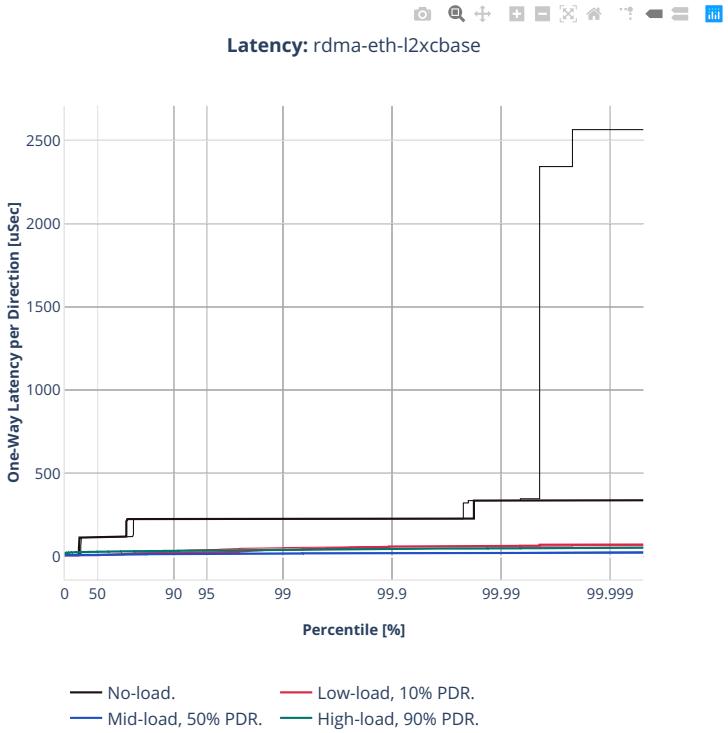

Latency: eth-l2bdscale1mmaclrn
One-Way Latency per Direction [uSec]
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

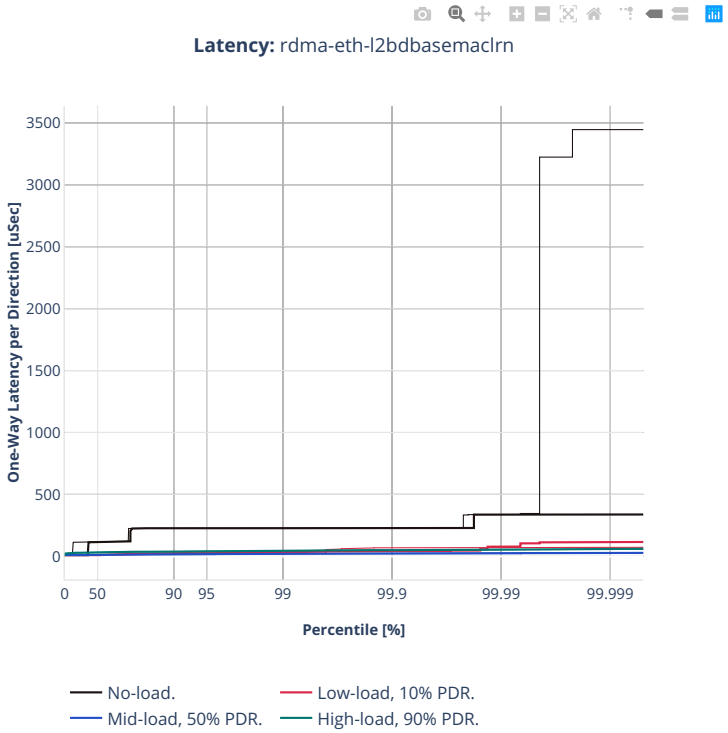
2n-clx-cx556a

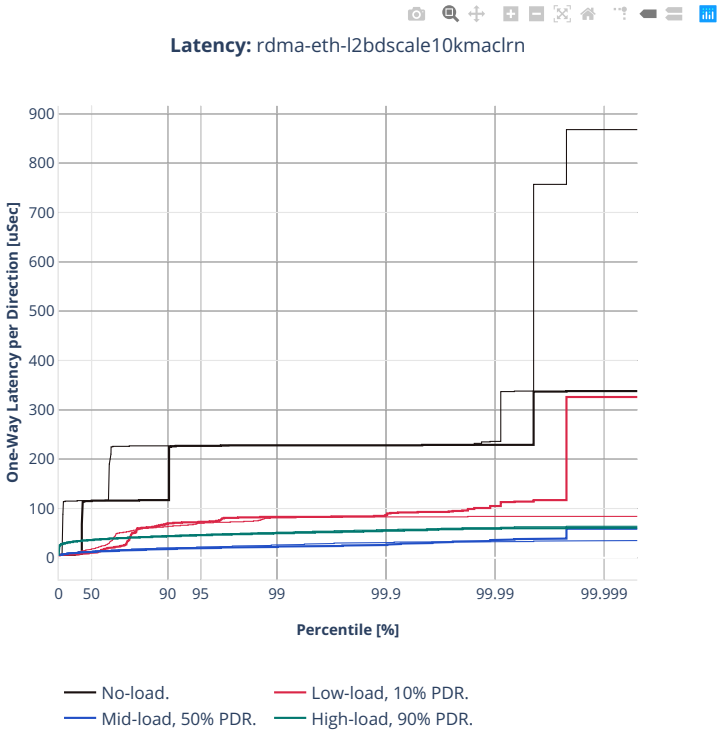
64b-2t1c-l2switching-base-scale-rdma



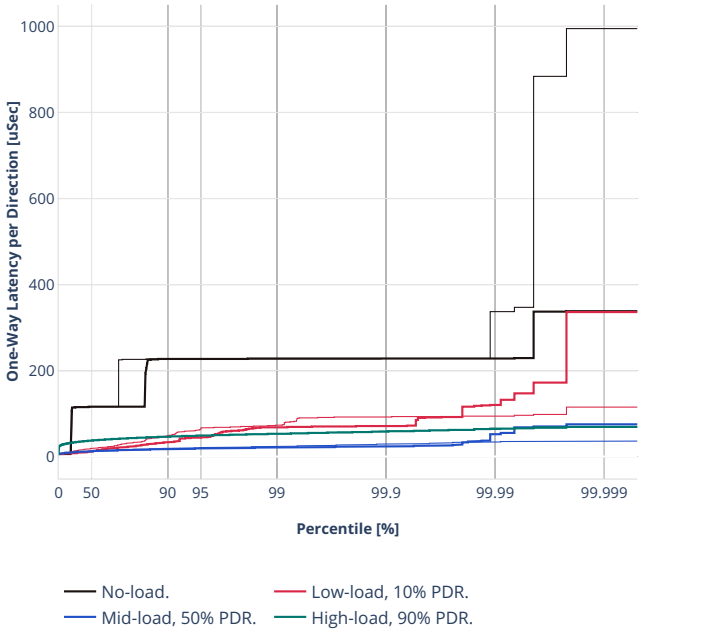

Latency: rdma-eth-l2patch
One-Way Latency per Direction [uSec]
3500
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

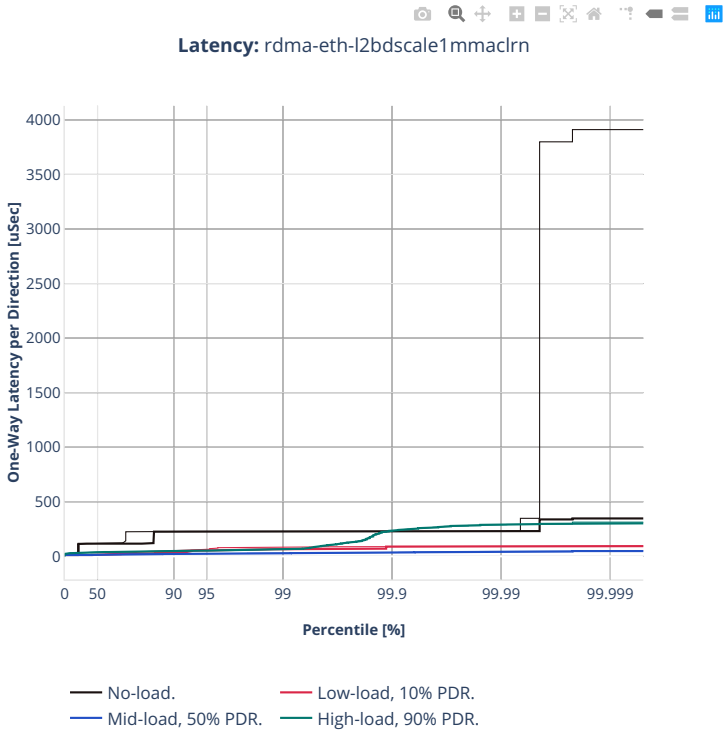

Latency: rdma-eth-l2xcbase
One-Way Latency per Direction [uSec]
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: rdma-eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
3500
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: rdma-eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
900
800
700
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

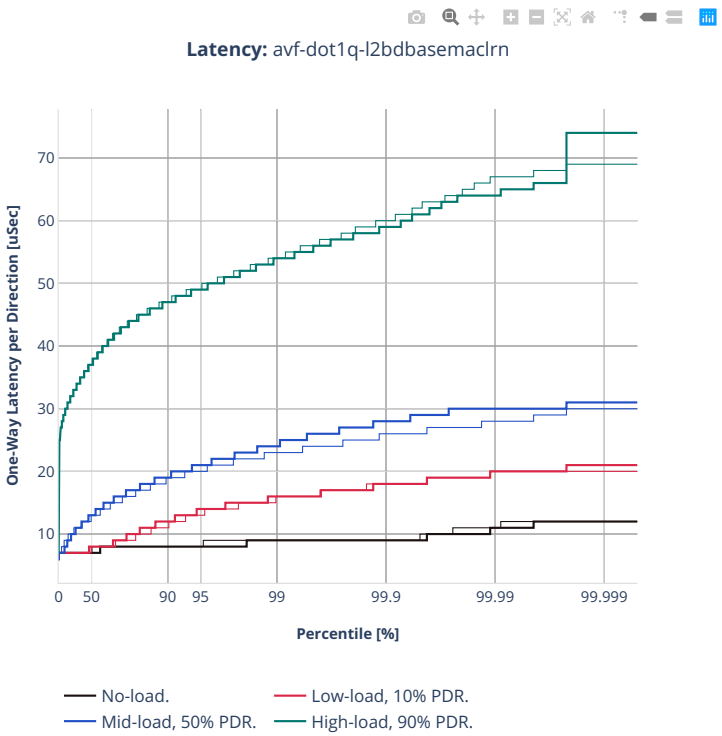
Latency: rdma-eth-l2bdscale100kmaclrn

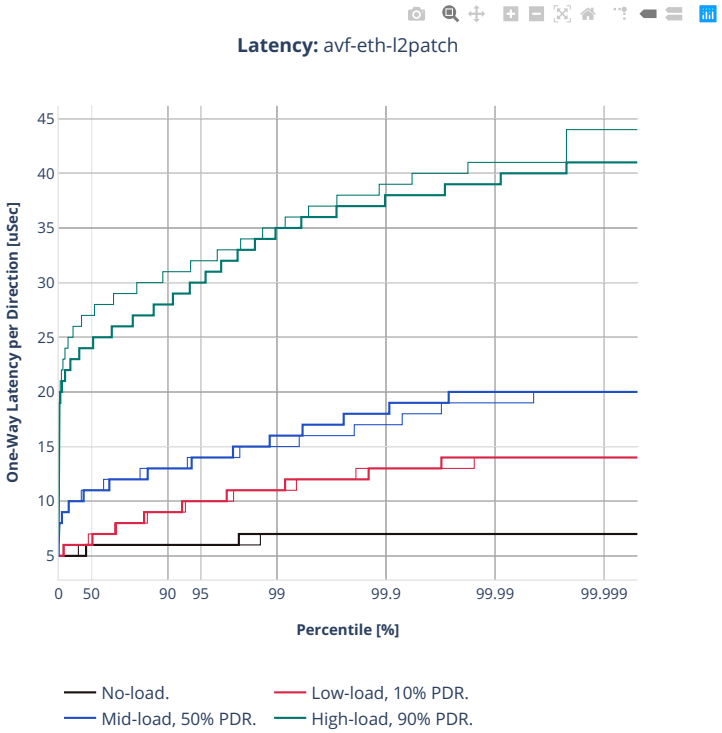


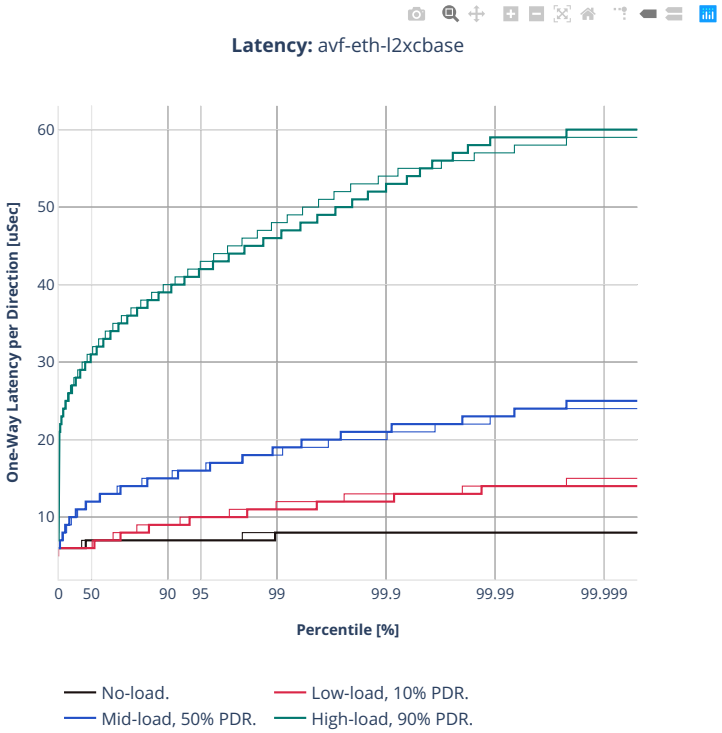

Latency: rdma-eth-l2bdscale1mmaclrn
One-Way Latency per Direction [uSec]
4000
3500
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

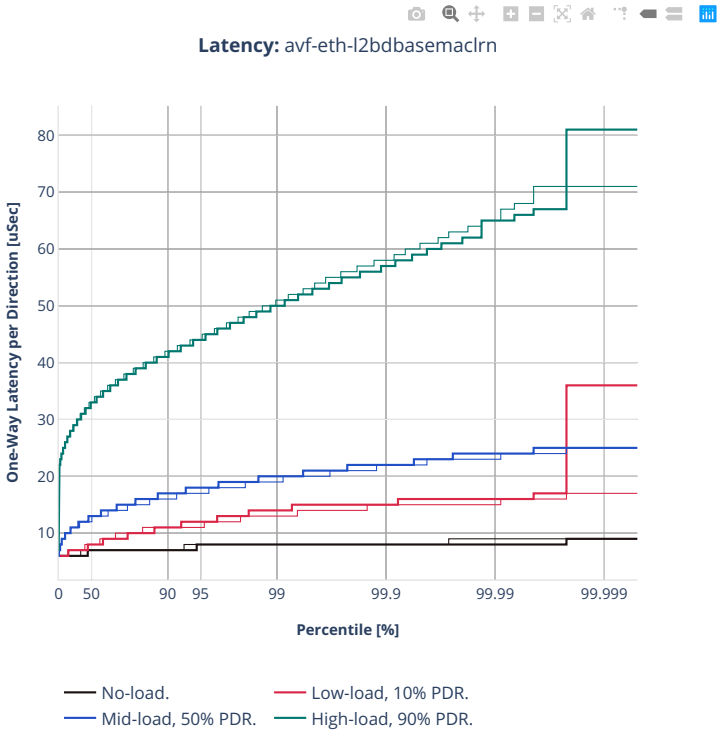
2n-clx-e810cq

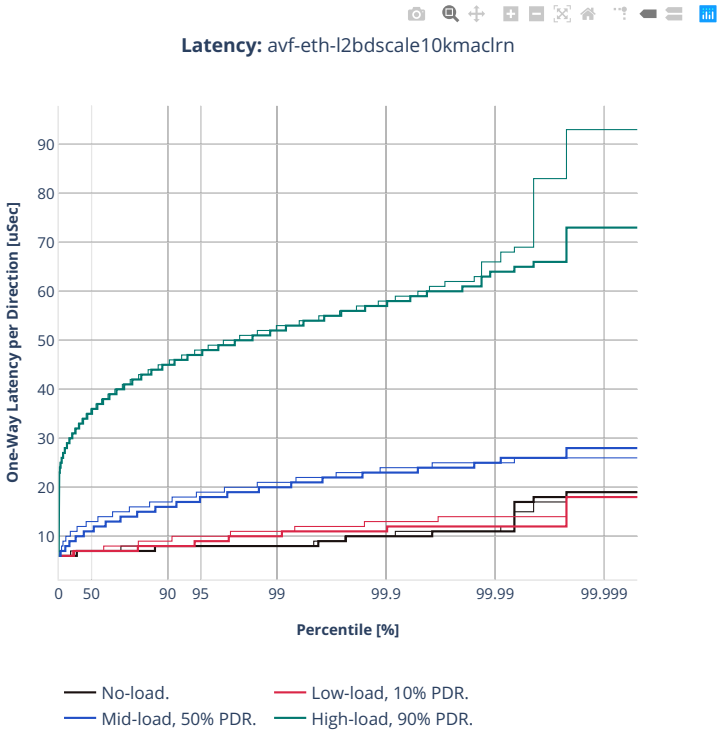
64b-2t1c-l2switching-base-scale-avf

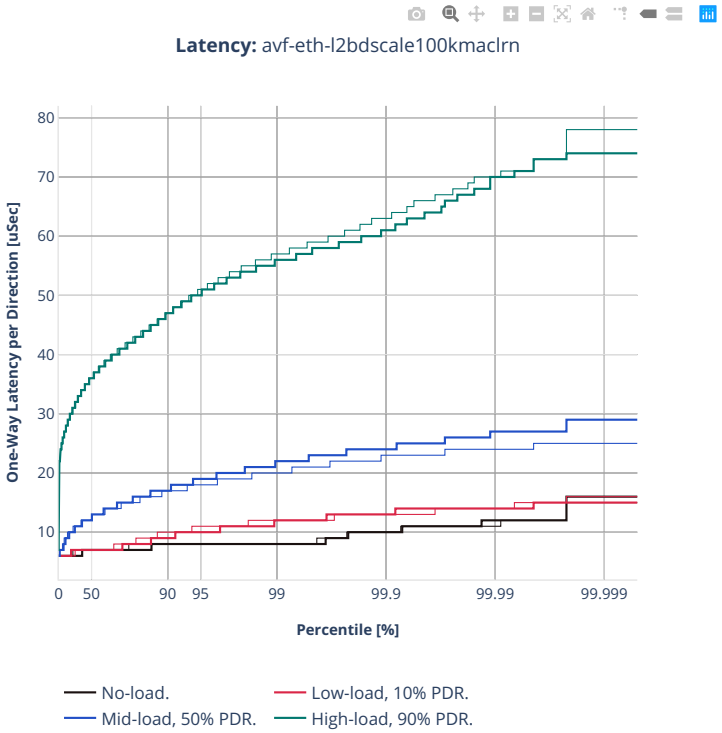


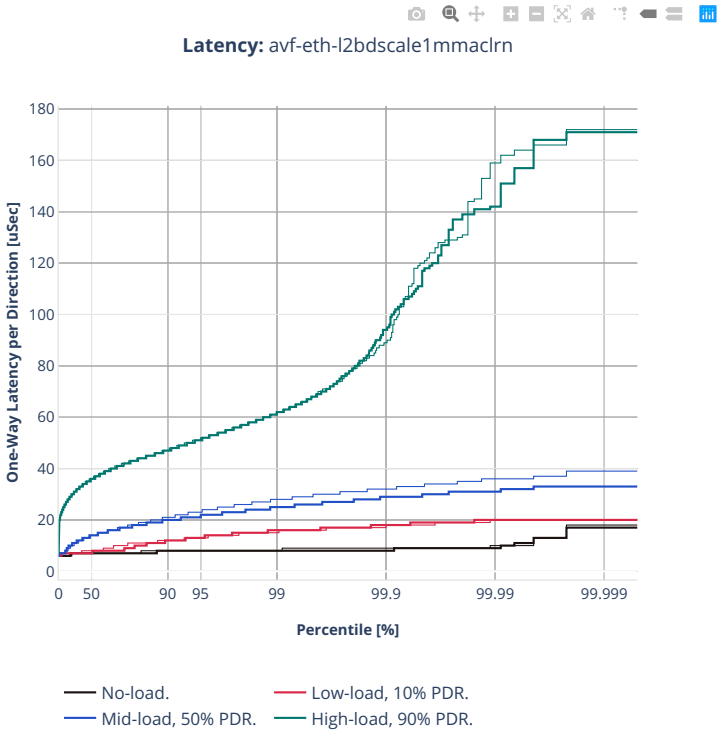

Latency: avf-eth-l2patch
One-Way Latency per Direction [uSec]
45
40
35
30
25
20
15
10
5
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-eth-l2xcbase
One-Way Latency per Direction [uSec]
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

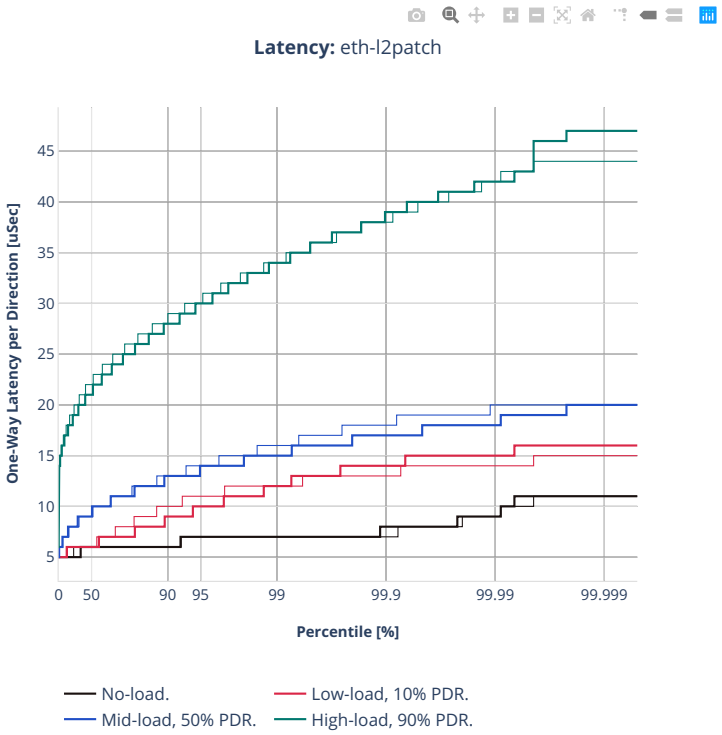

Latency: avf-eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

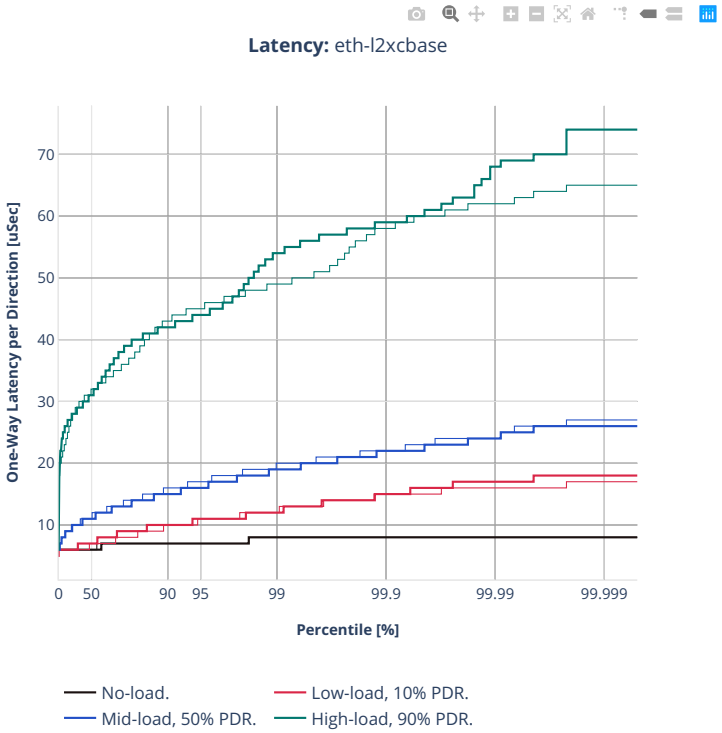

Latency: avf-eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

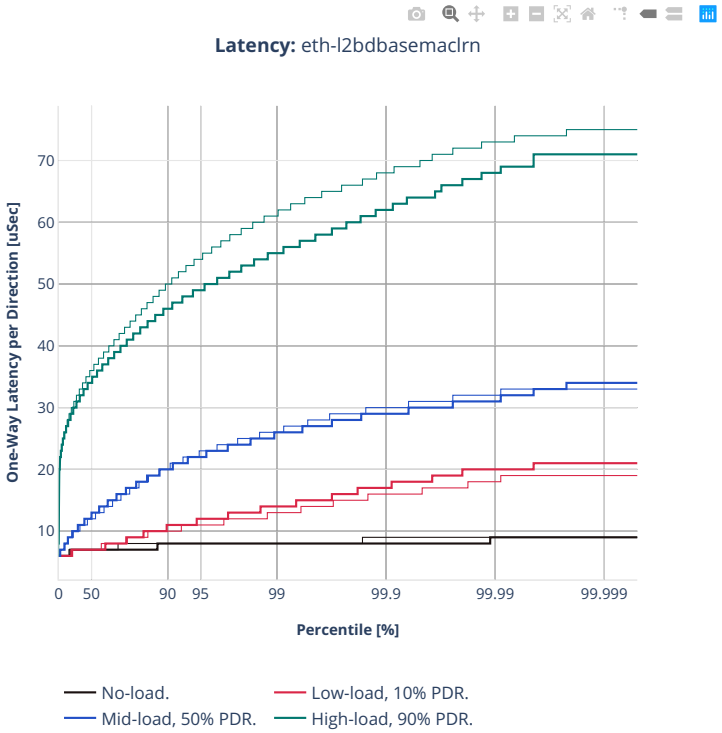

Latency: avf-eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

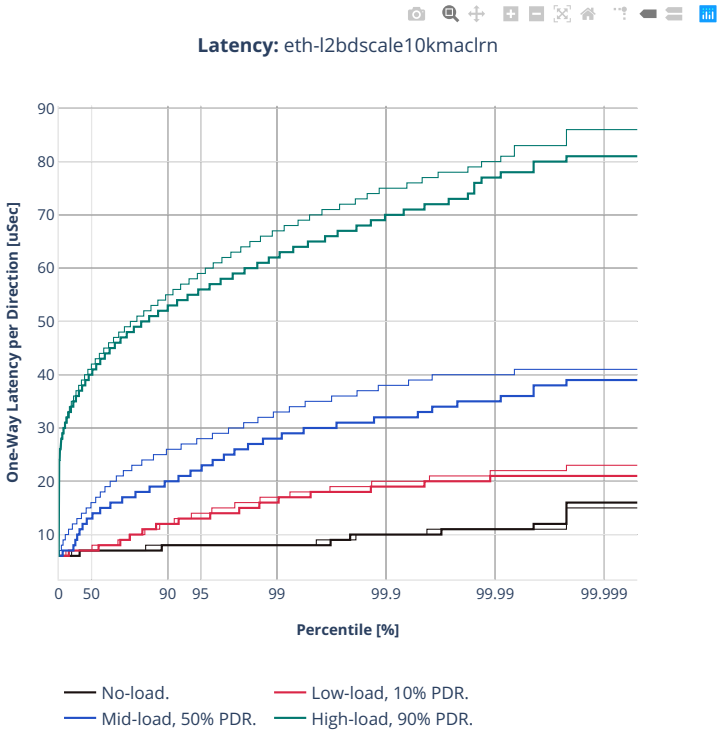

Latency: avf-eth-l2bdscale1mmaclrn
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-l2switching-base-scale-dpdk



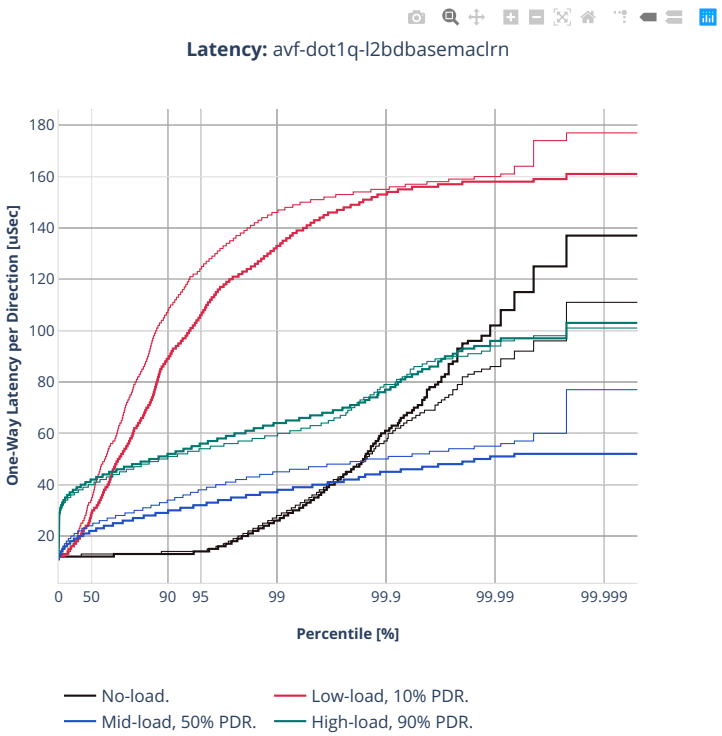

Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

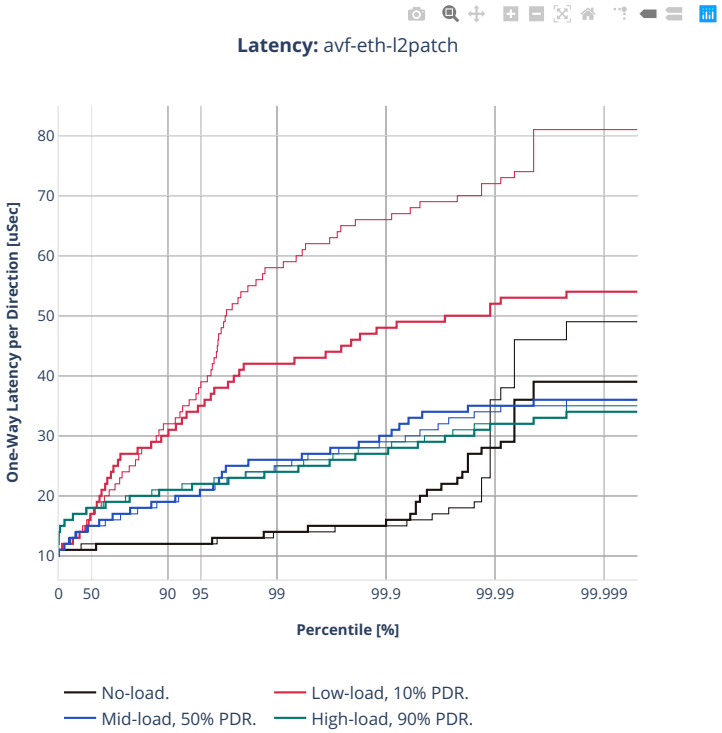

Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

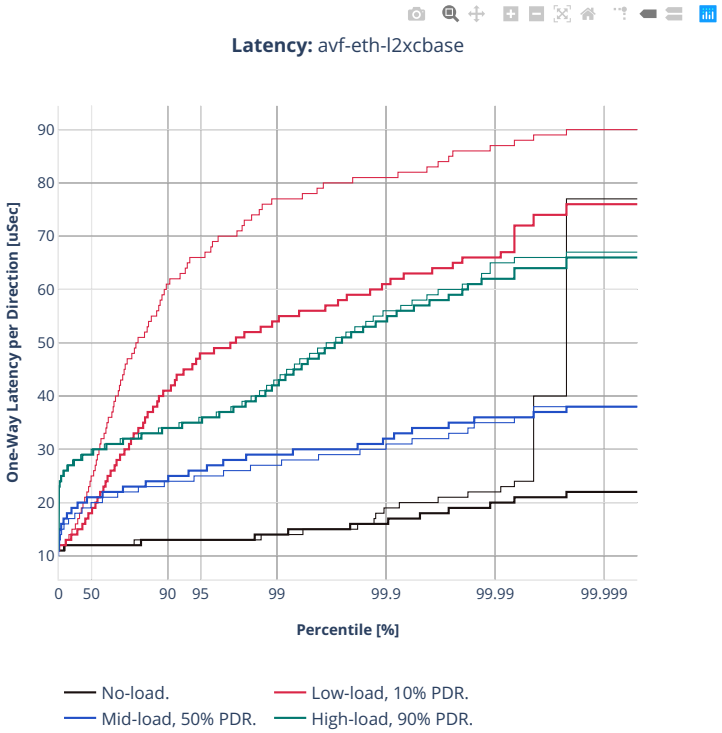

Latency: eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

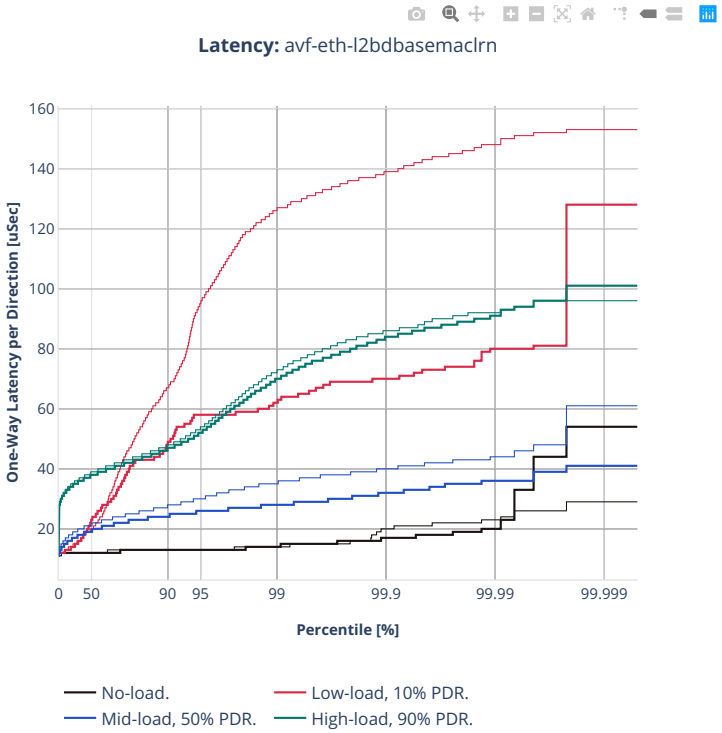
2n-zn2-xxv710

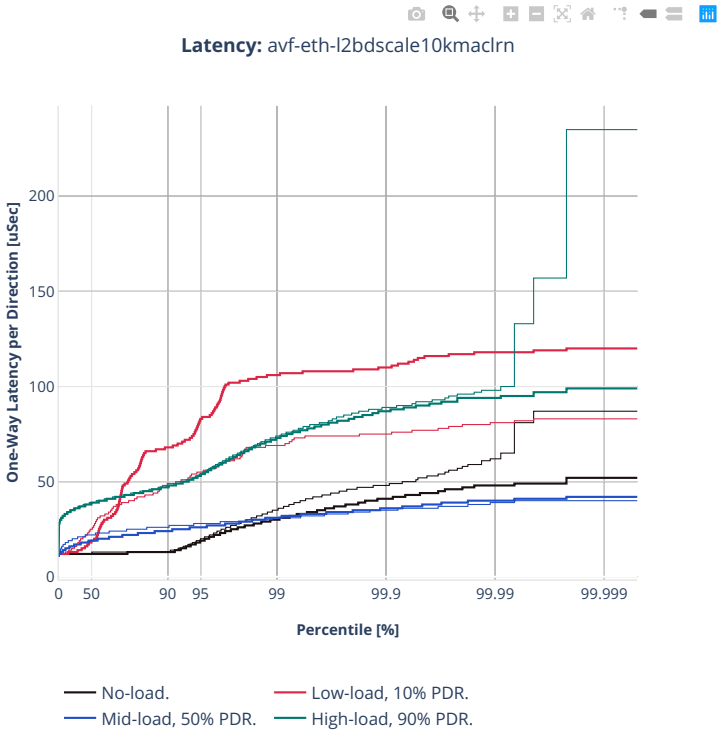
64b-2t1c-l2switching-base-scale-avf

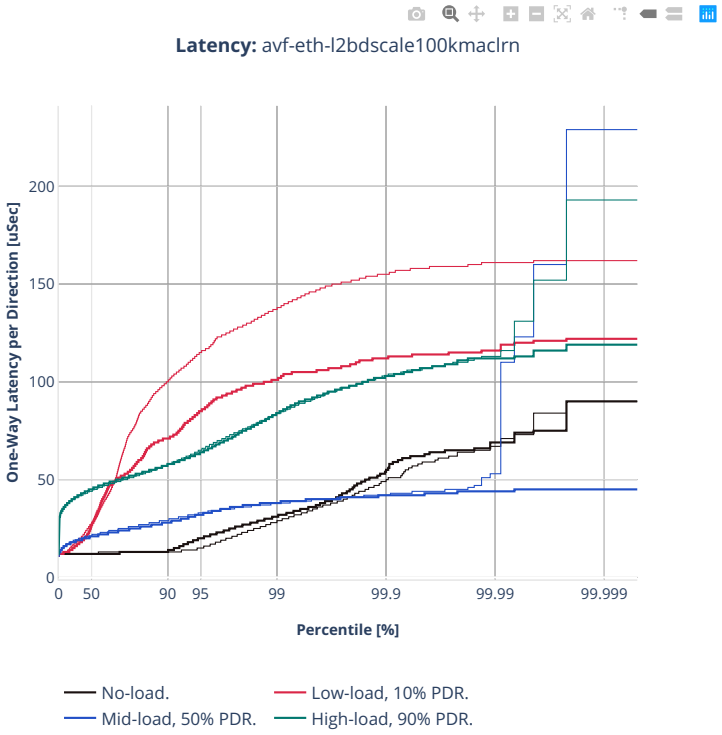



Latency: avf-eth-l2patch
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

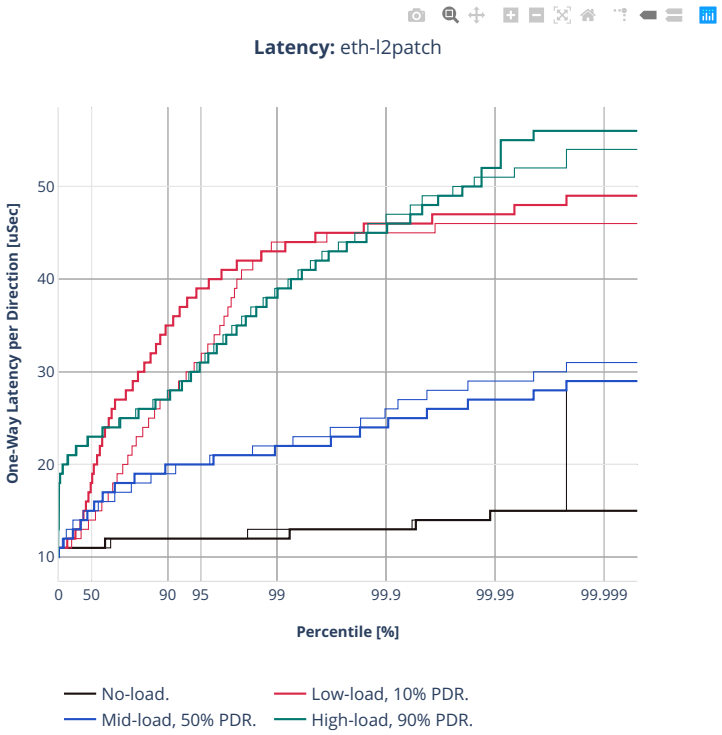

Latency: avf-eth-l2xcbase
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

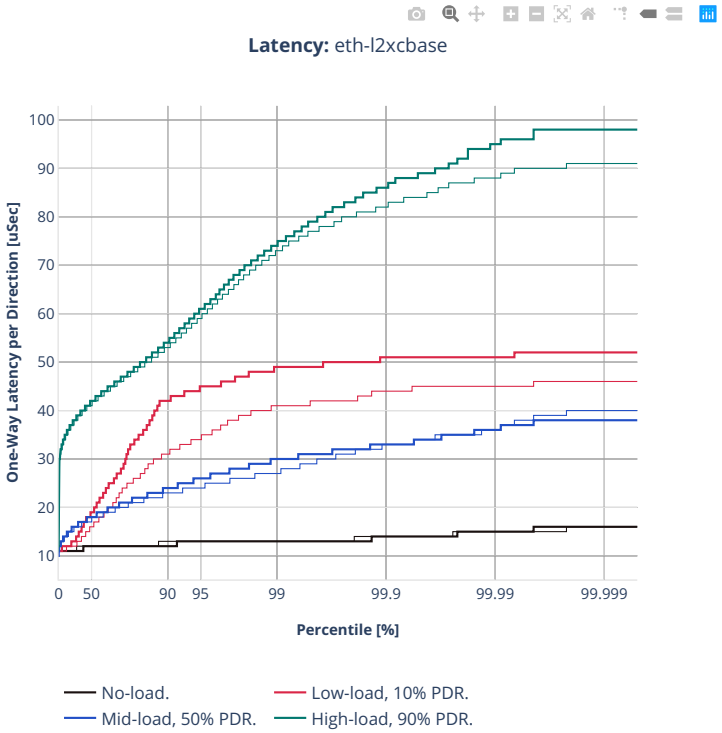

Latency: avf-eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

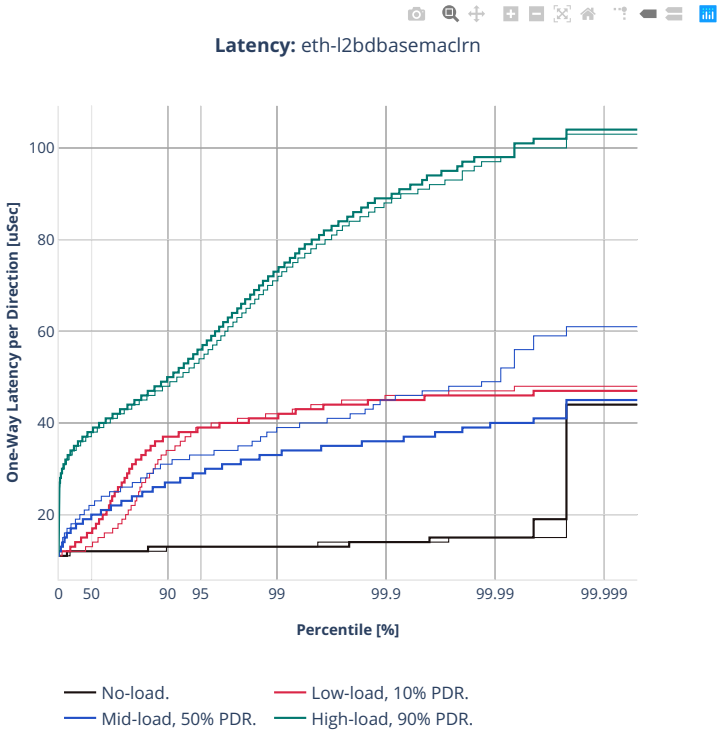

Latency: avf-eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

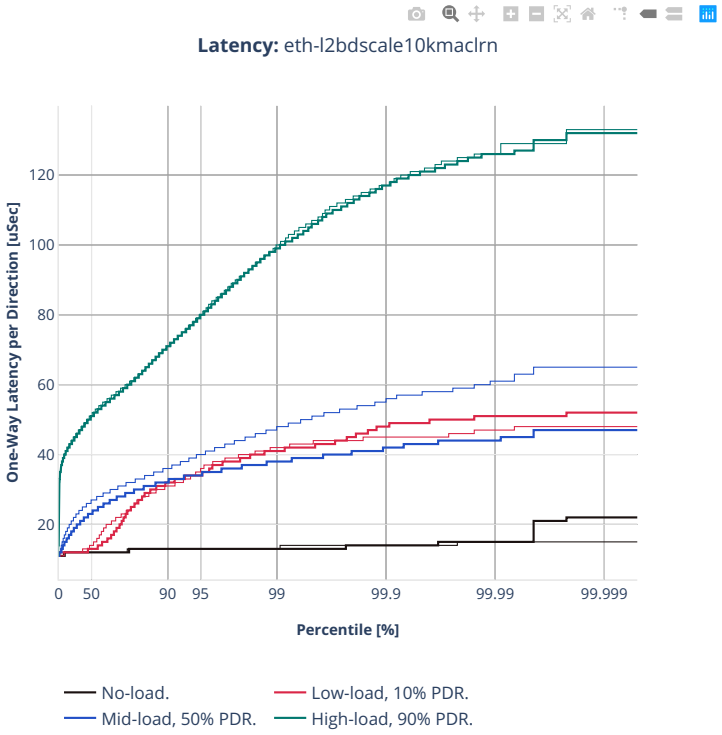

Latency: avf-eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

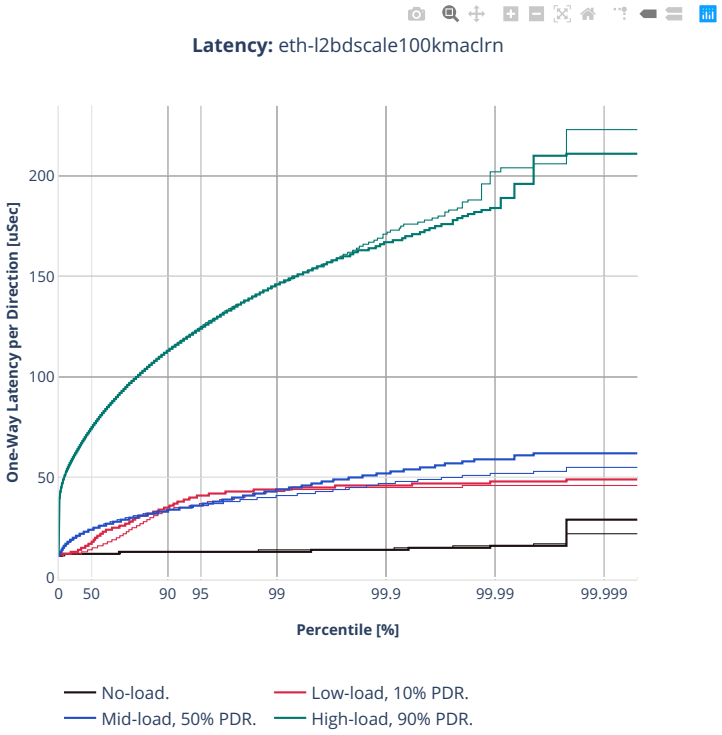
64b-2t1c-l2switching-base-scale-dpdk




Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
100
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

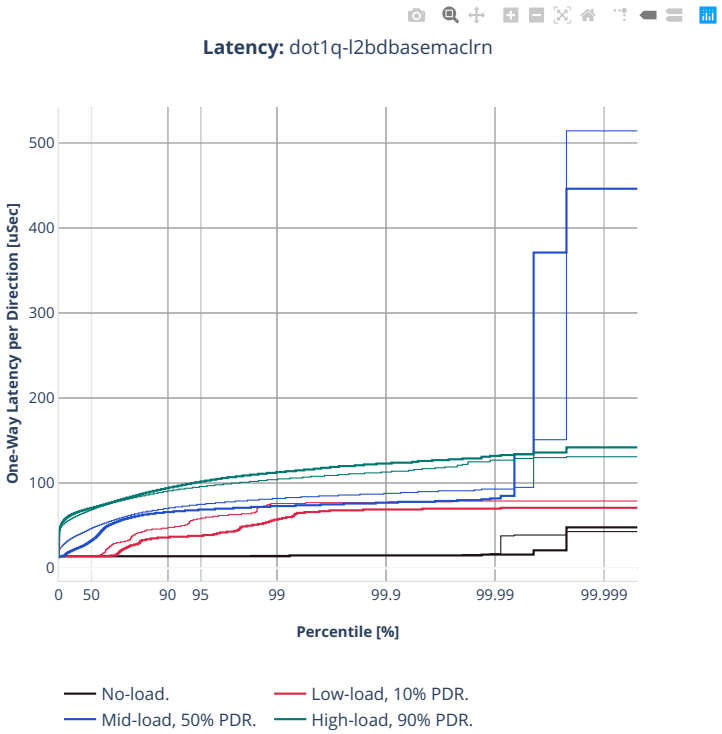

Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

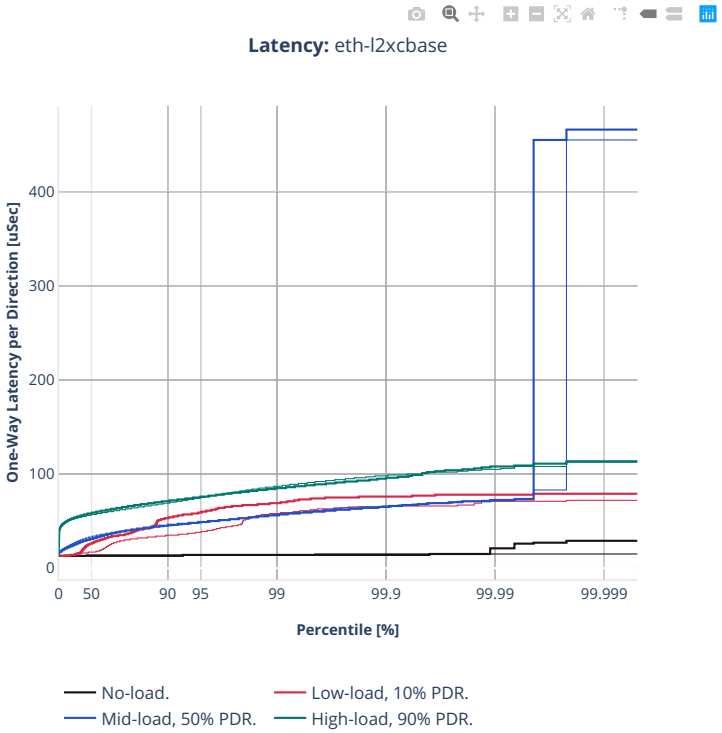

Latency: eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

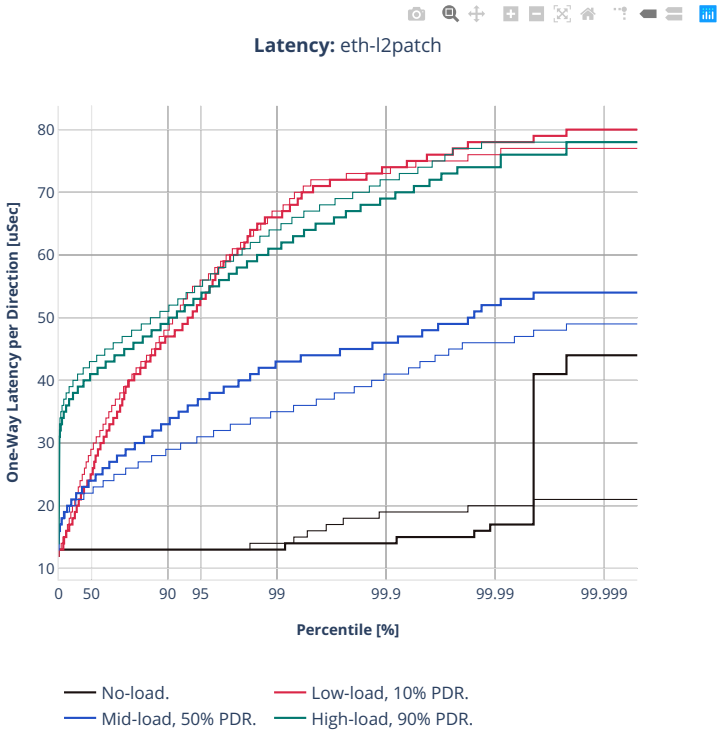

Latency: eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

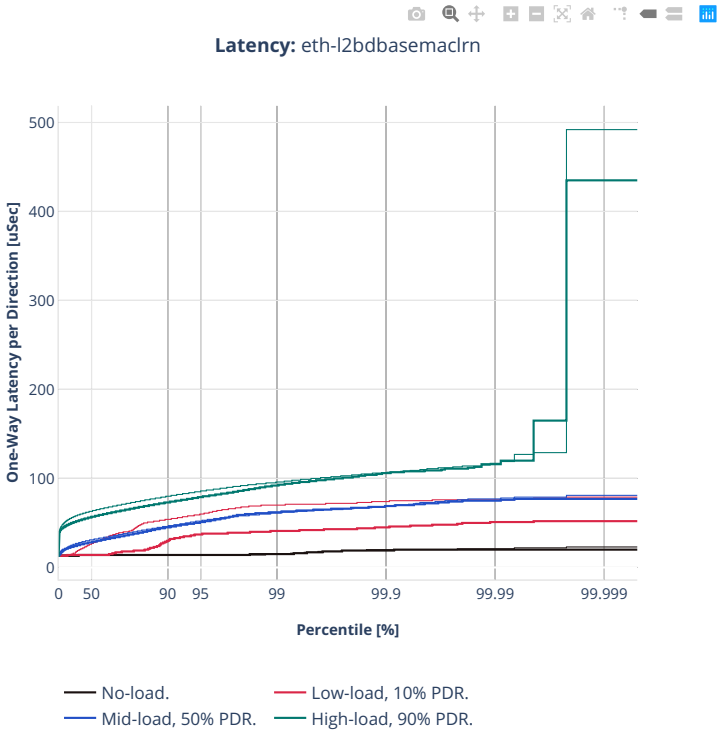
3n-alt-xl710

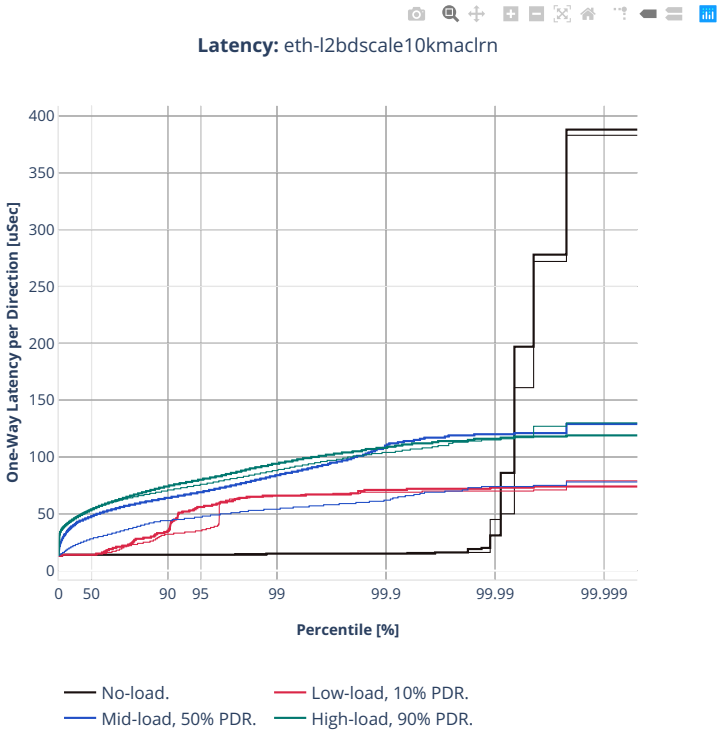
64b-1t1c-l2switching-base-scale

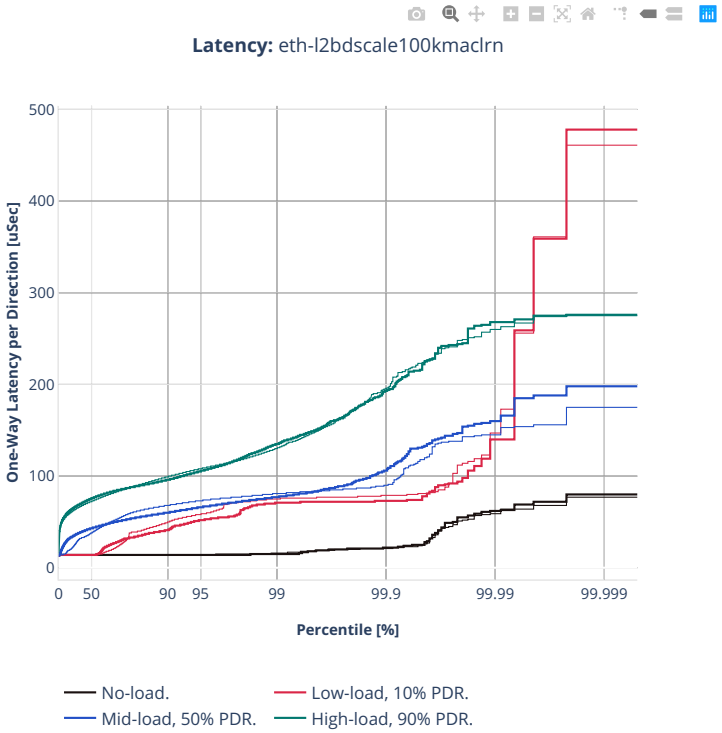



Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

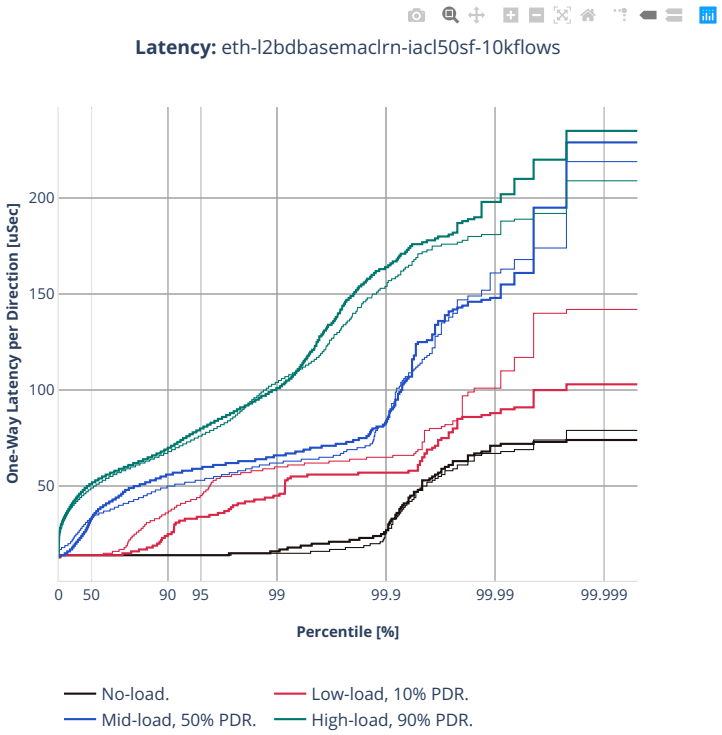

Latency: eth-l2patch
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

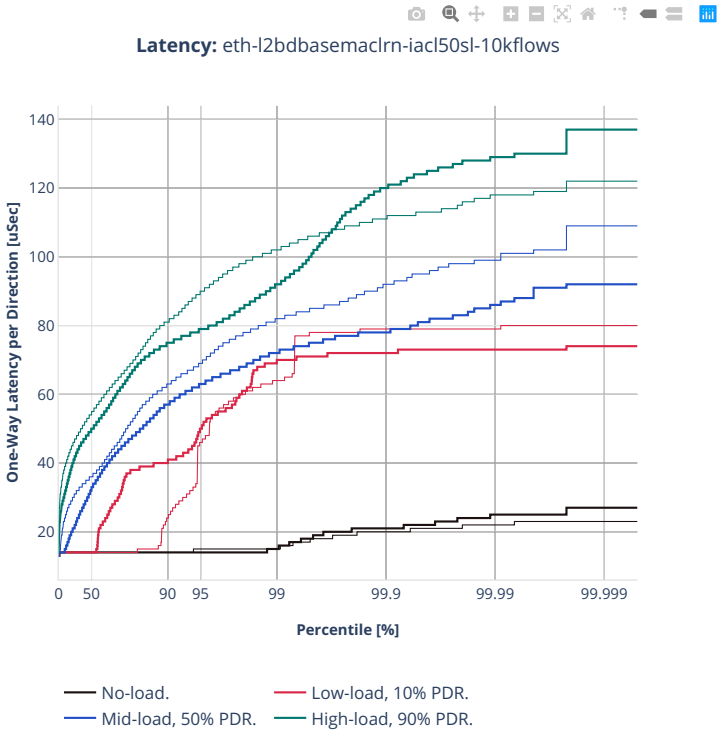

Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

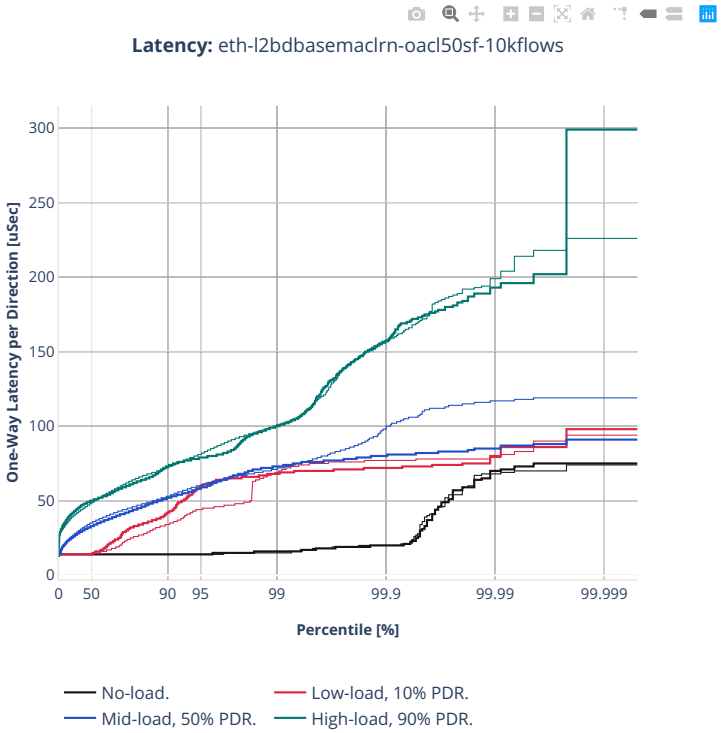

Latency: eth-l2bdscale10kmaclrn
One-Way Latency per Direction [uSec]
400
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

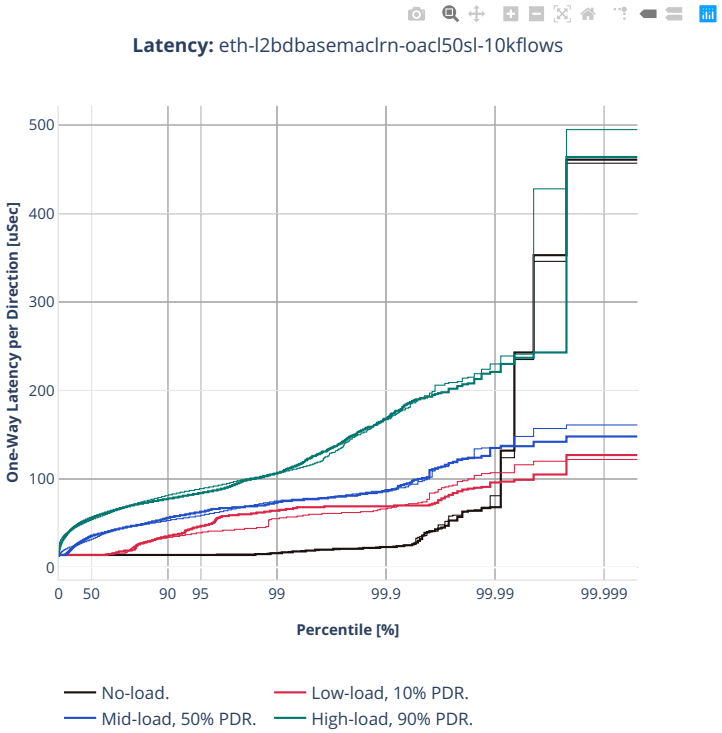

Latency: eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

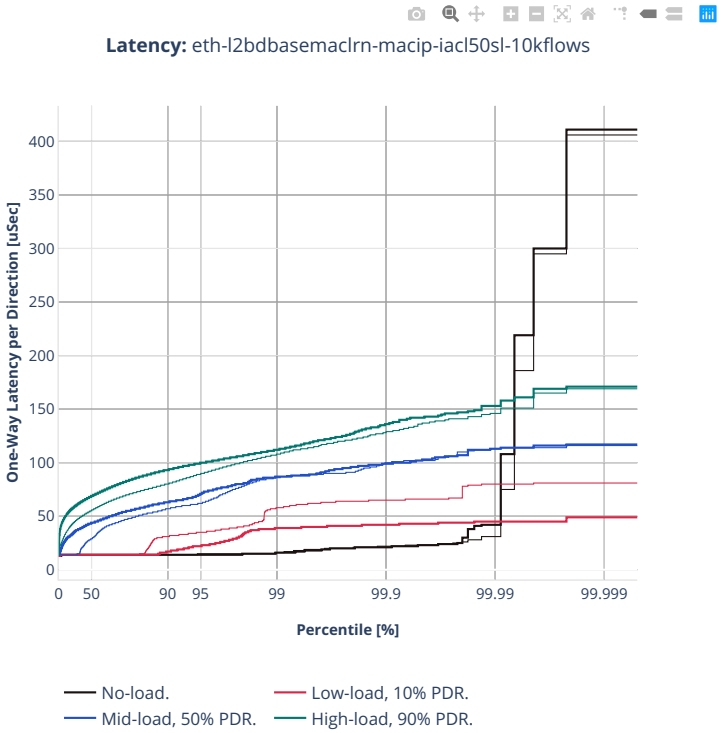
64b-1t1c-features-l2switching-base




Latency: eth-l2bdbasemaclrn-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

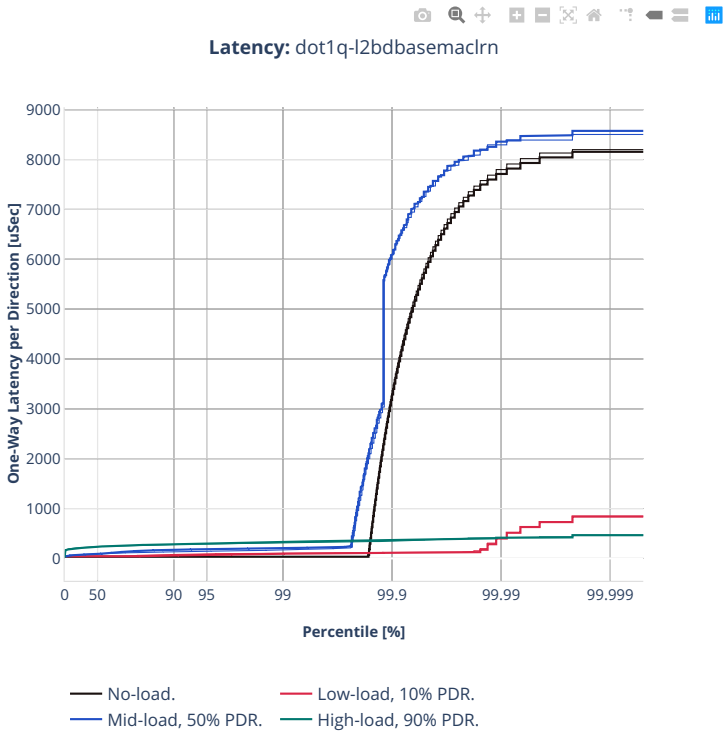

Latency: eth-l2bdbasemaclrn-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

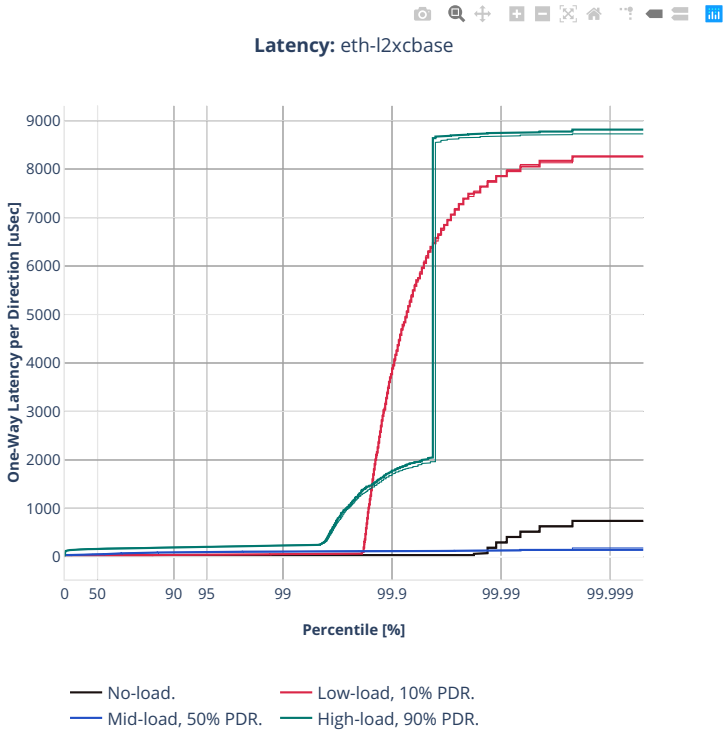

Latency: eth-l2bdbasemaclrn-macip-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
400
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

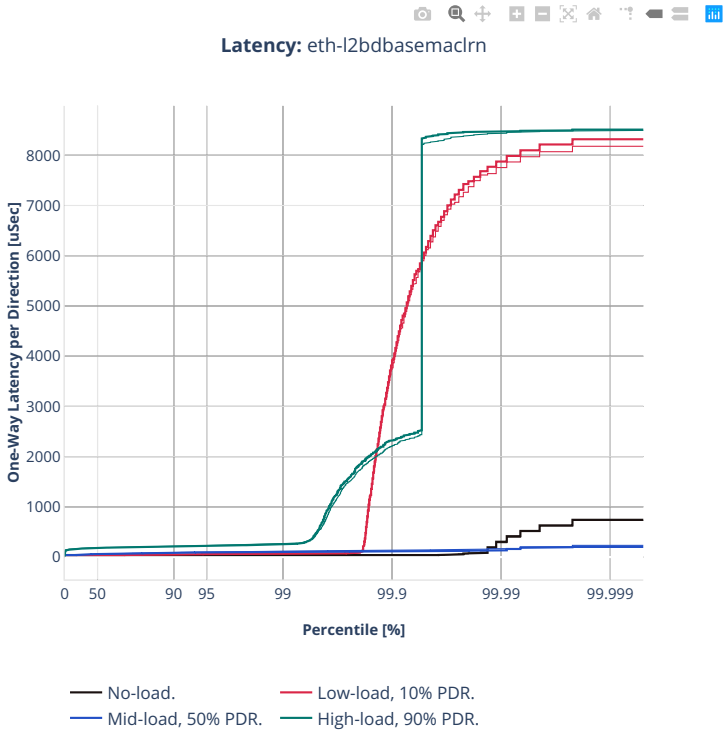
3n-tsh-x520

64b-1t1c-l2switching-base-scale-ixgbe

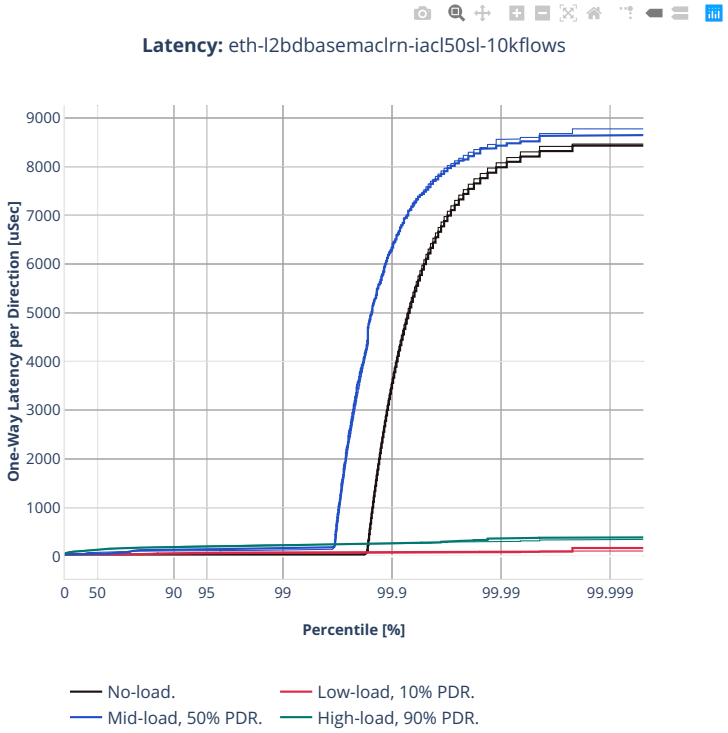


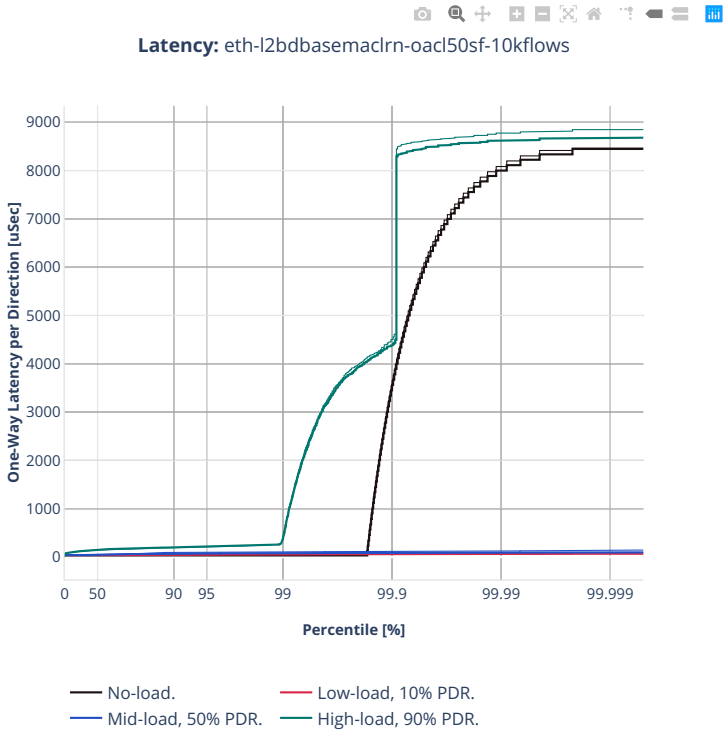

Latency: eth-l2patch
One-Way Latency per Direction [uSec]
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

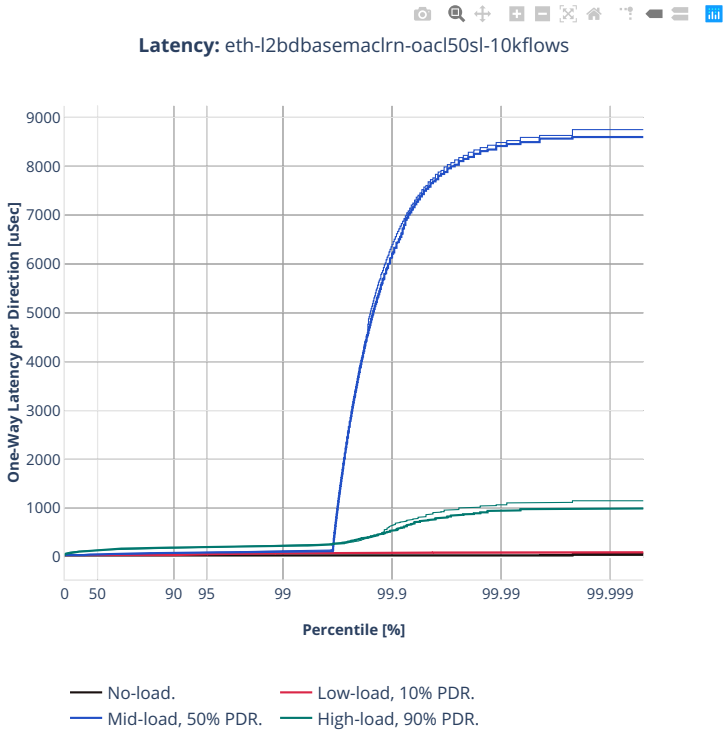

Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-1t1c-features-l2switching-base-ixgbe

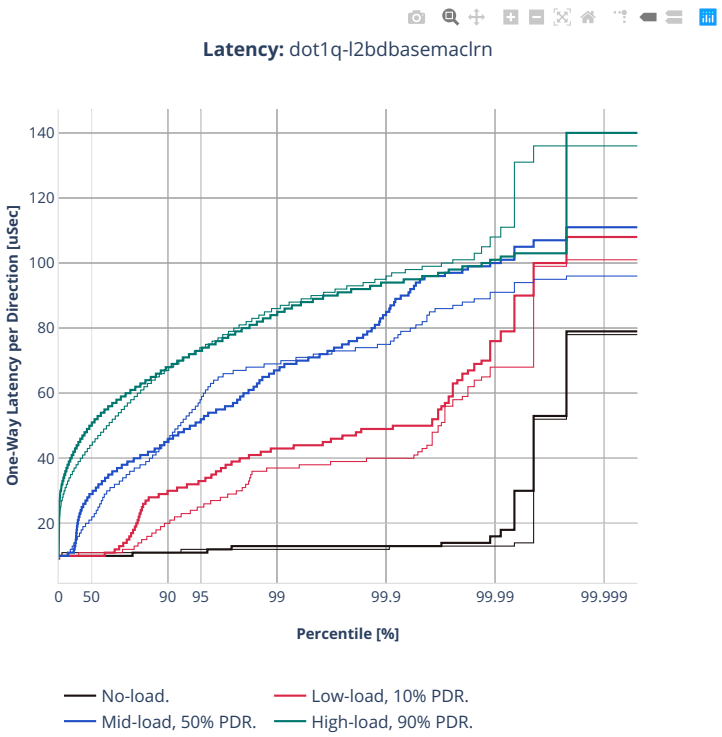


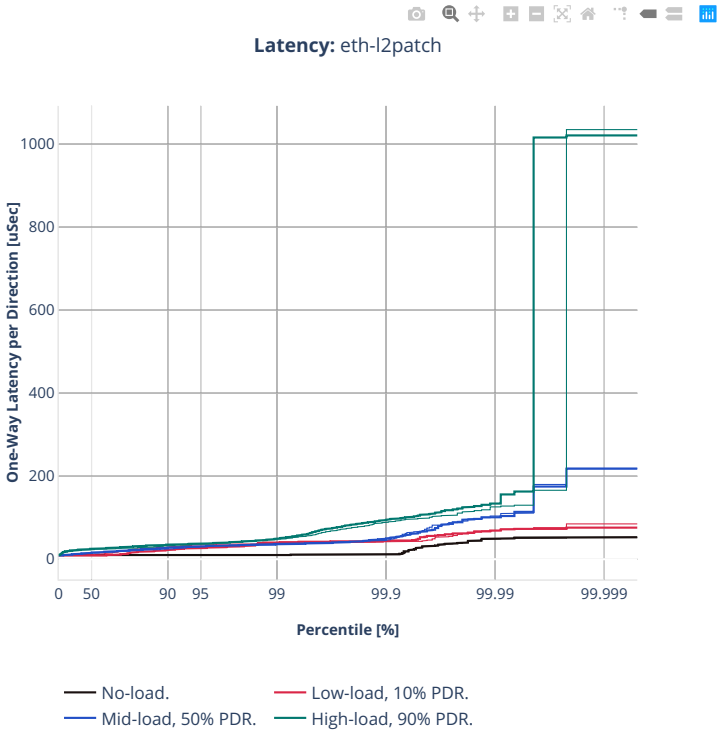

Latency: eth-l2bdbasemaclrn-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

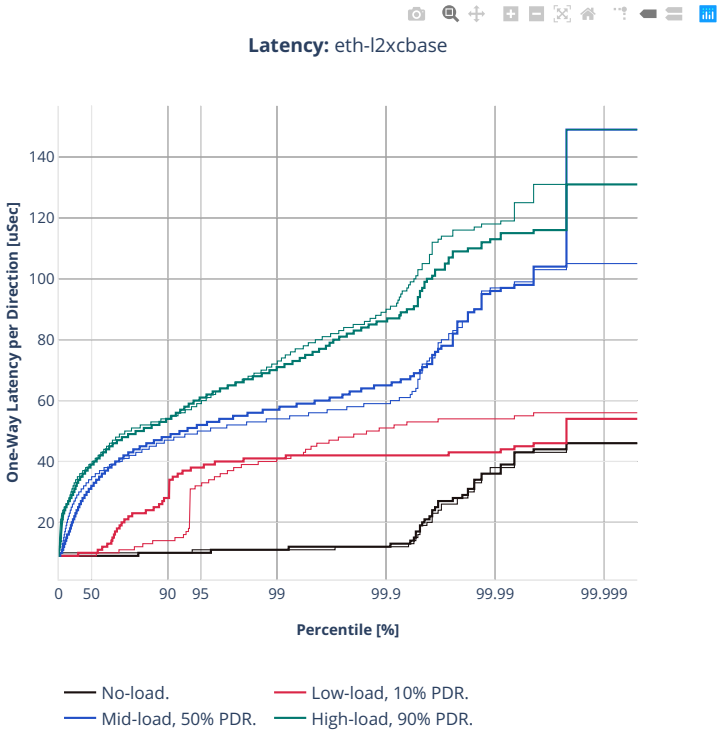

Latency: eth-l2bdbasemaclrn-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

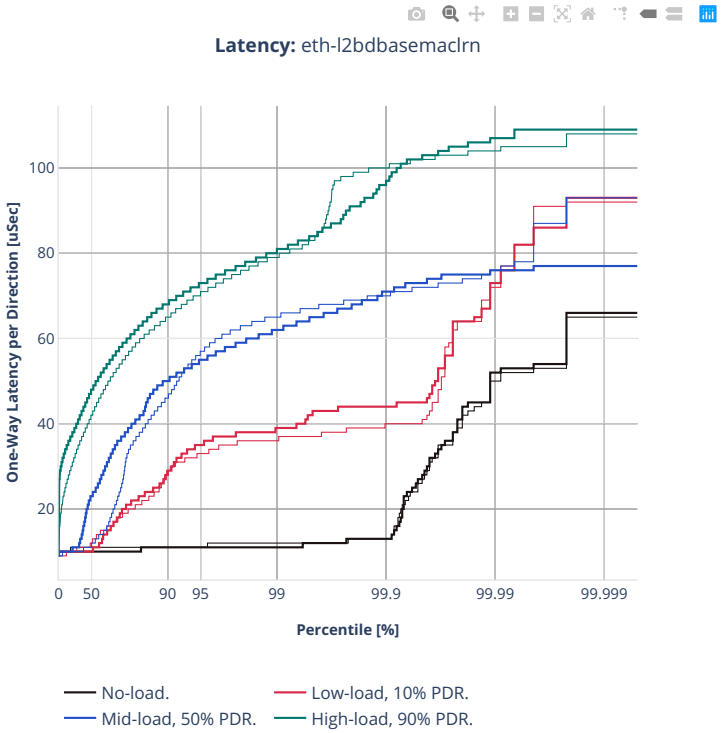
2n-tx2-xl710

64b-1t1c-l2switching-base-dpdk

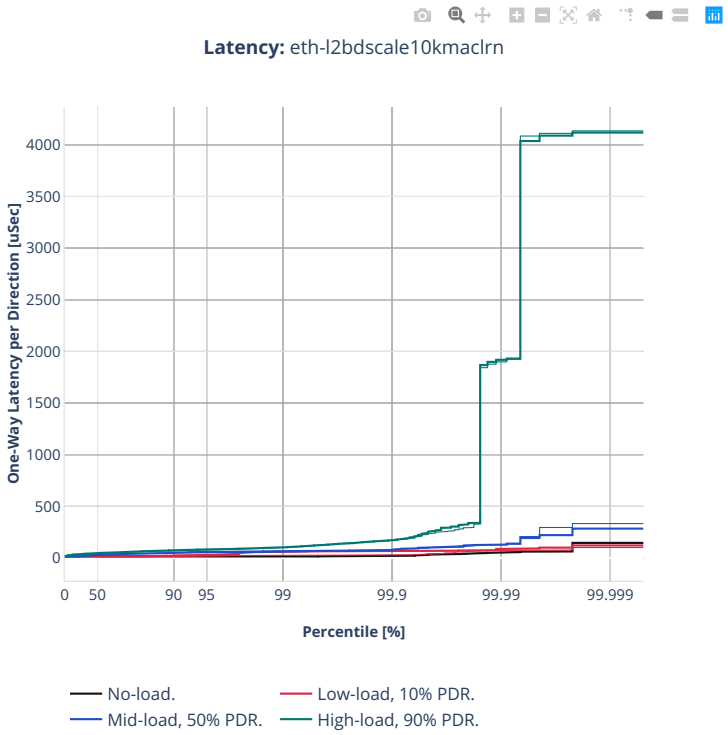


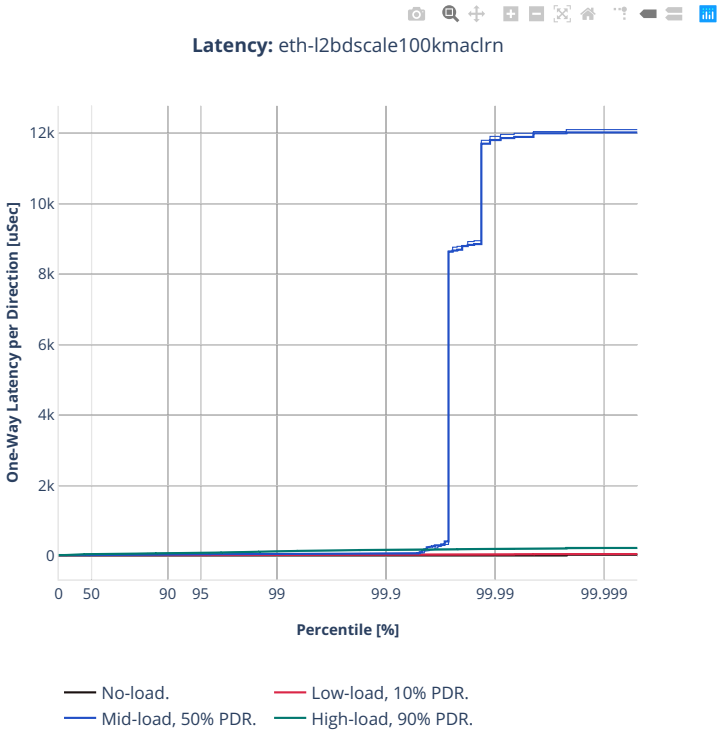

Latency: eth-l2patch
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2xcbase
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

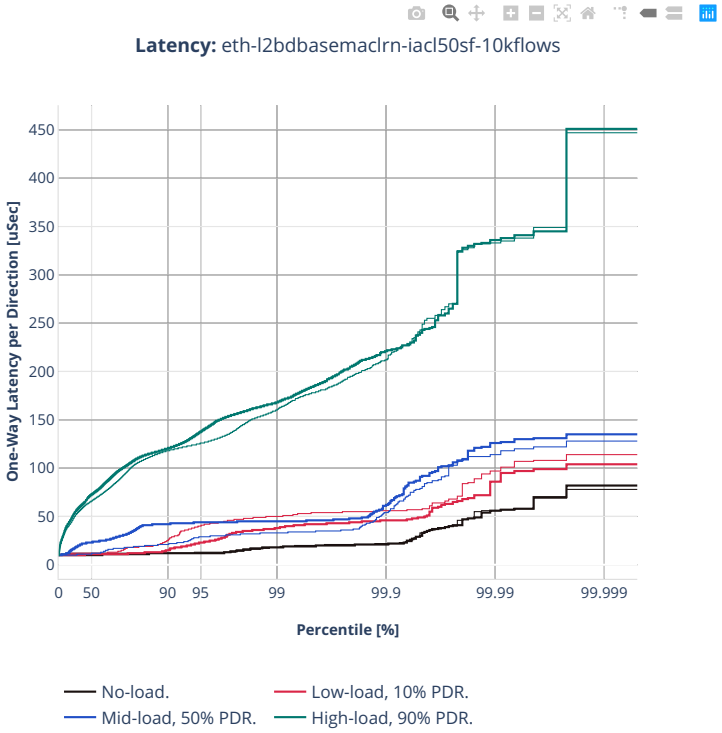

Latency: eth-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

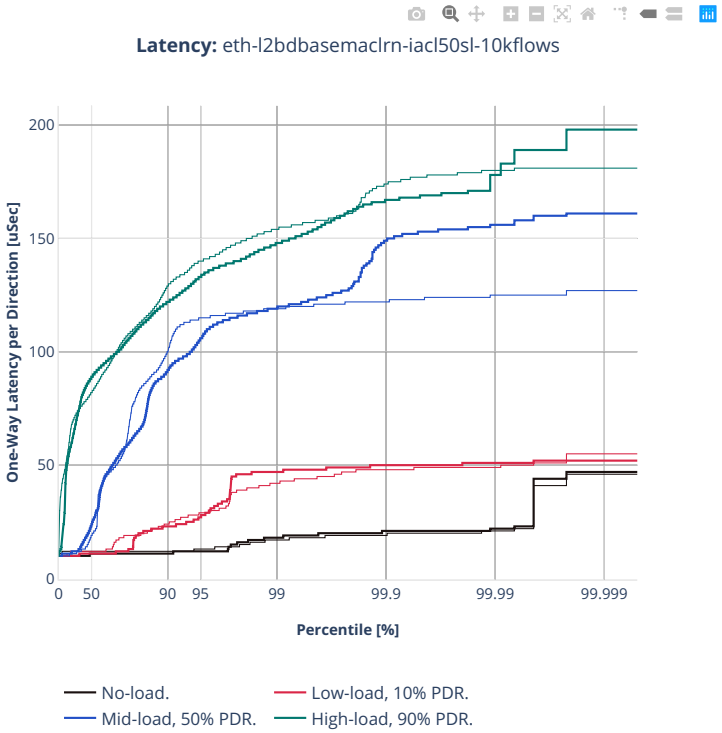
64b-1t1c-l2switching-scale-dpdk

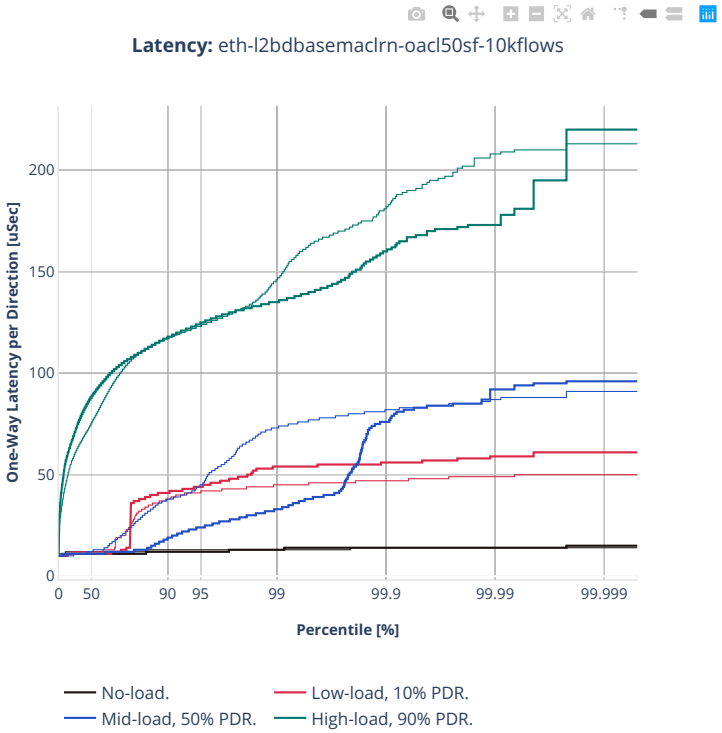


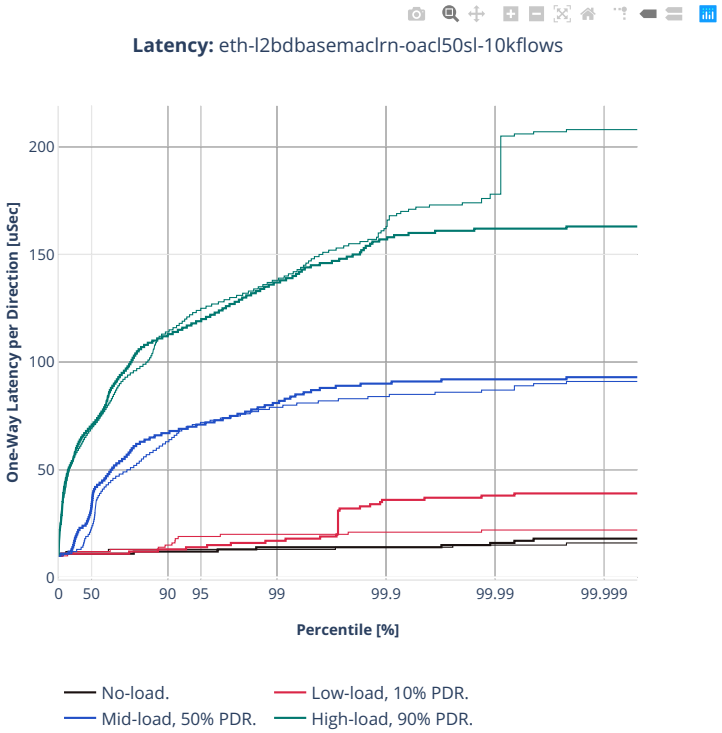

Latency: eth-l2bdscale100kmaclrn
One-Way Latency per Direction [uSec]
12k
10k
8k
6k
4k
2k
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

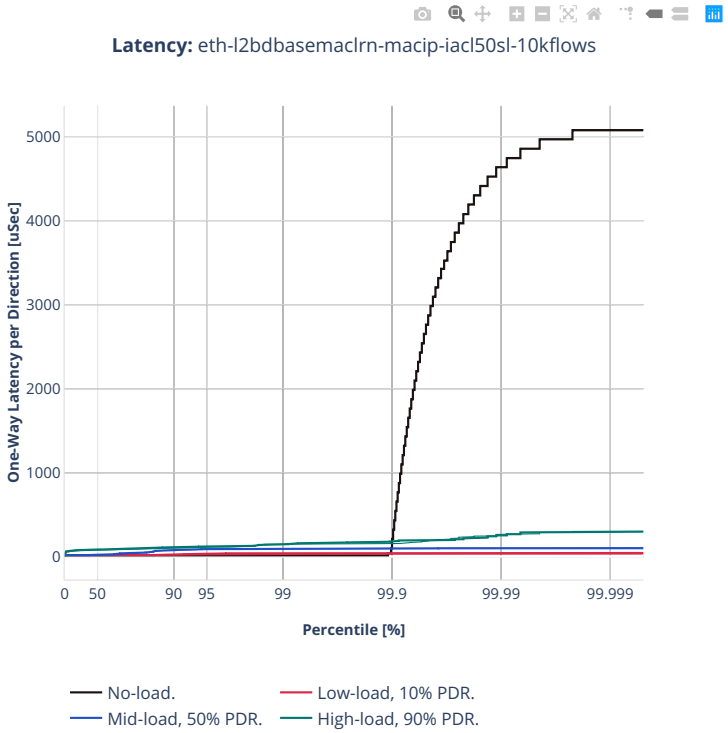
64b-1t1c-features-l2switching-base-dpdk




Latency: eth-l2bdbasemaclrn-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn-macip-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

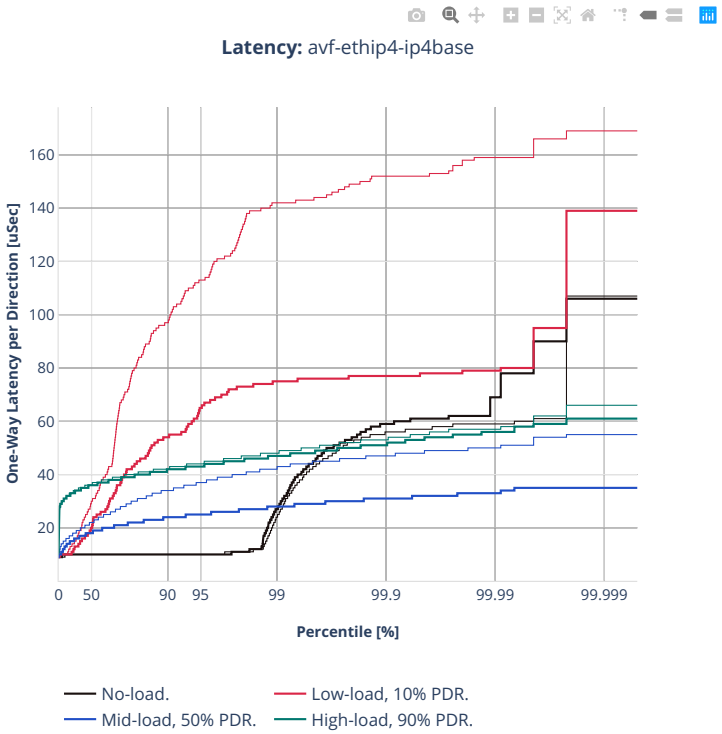
2.5.2 IPv4 Routing

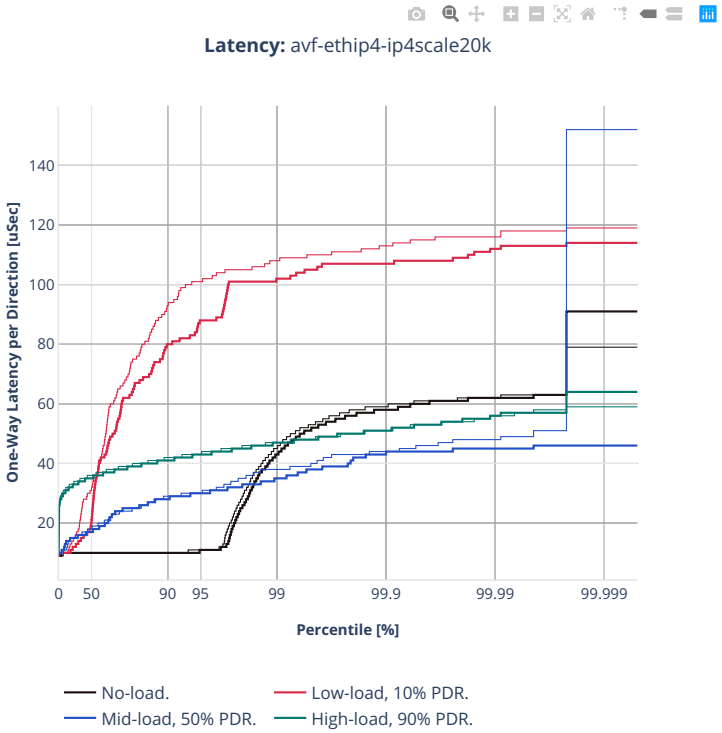
CSIT source code for the test cases used for plots can be found in CSIT git repository[151].

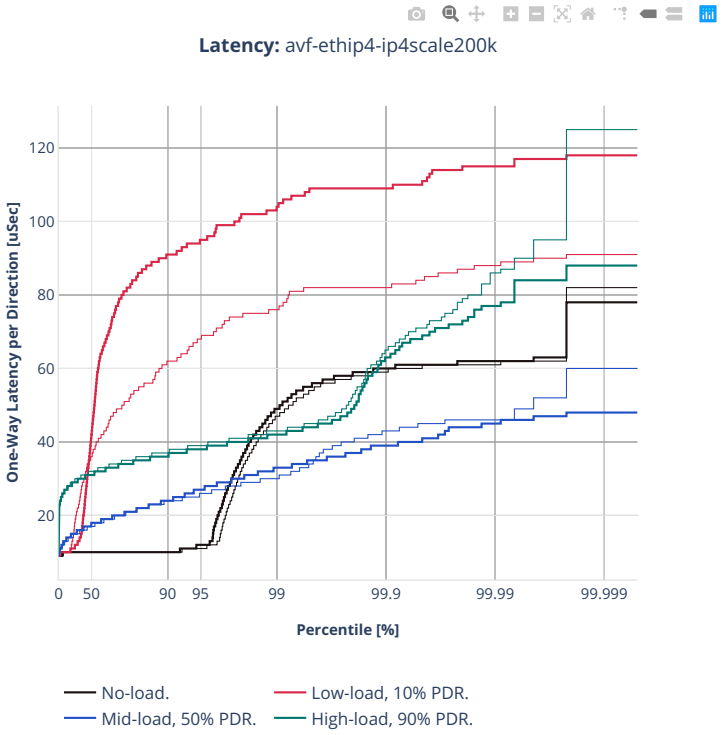
[151] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls2210

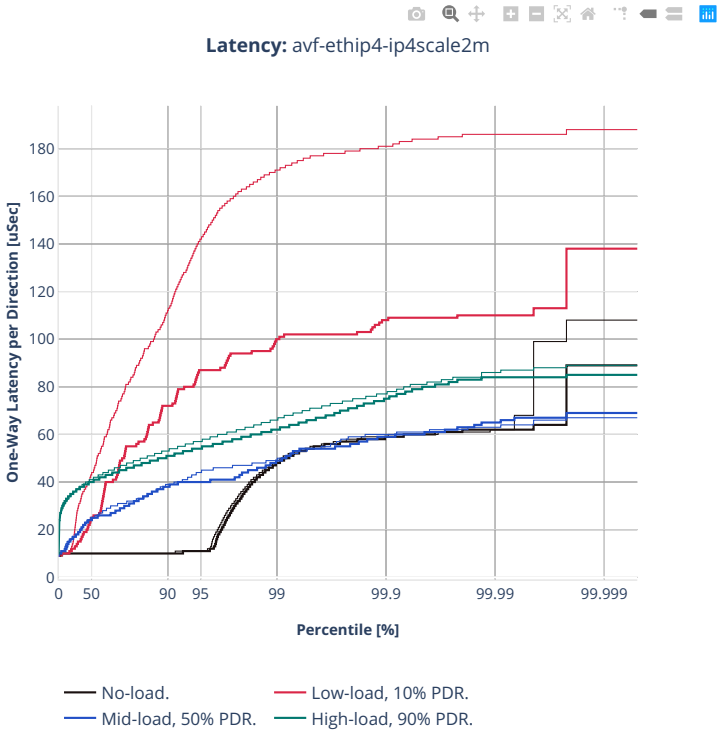
2n-icx-xxv710

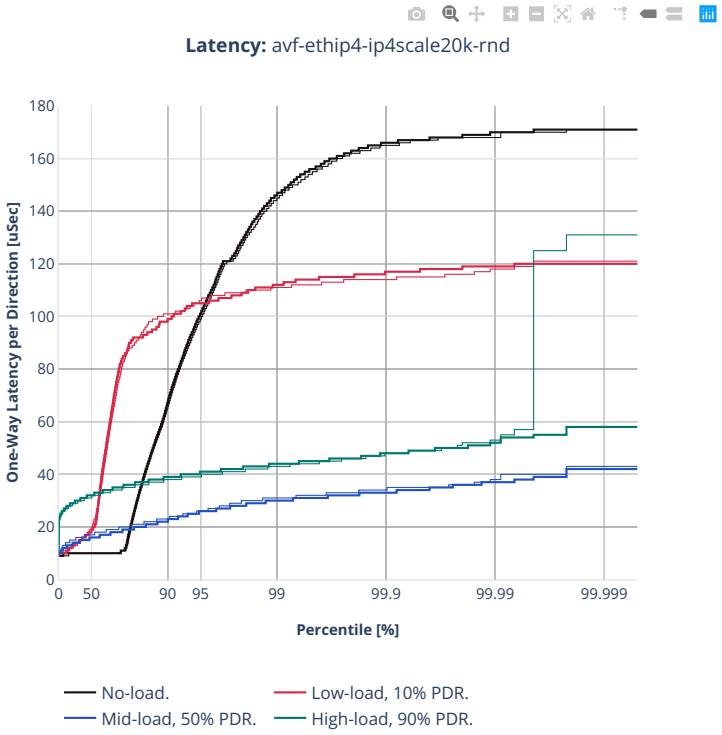
64b-2t1c-ip4routing-base-scale-avf

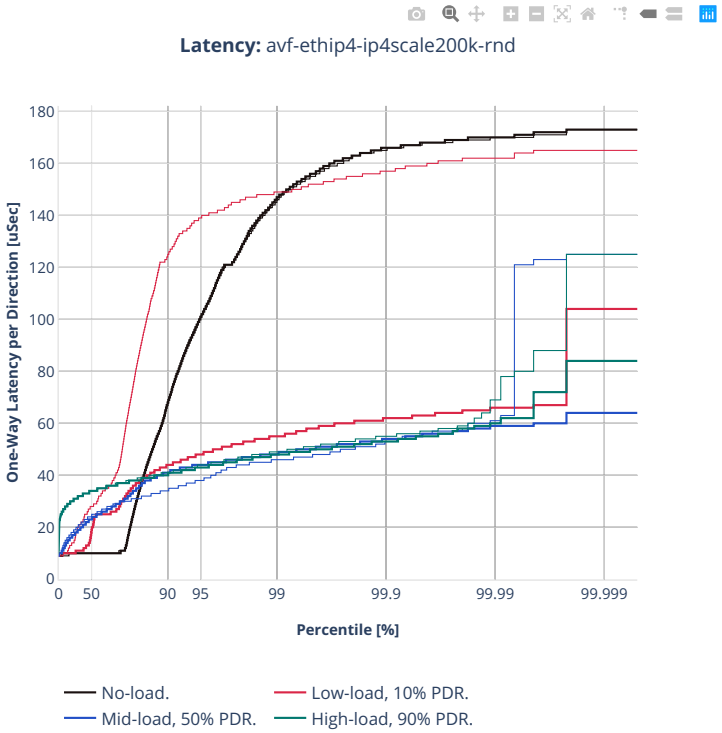


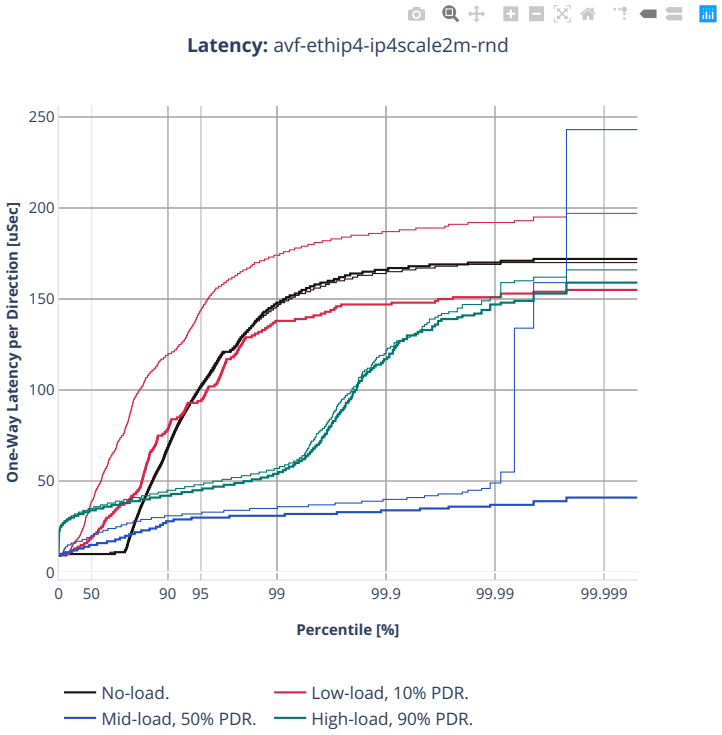

Latency: avf-ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4scale200k
One-Way Latency per Direction [uSec]
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

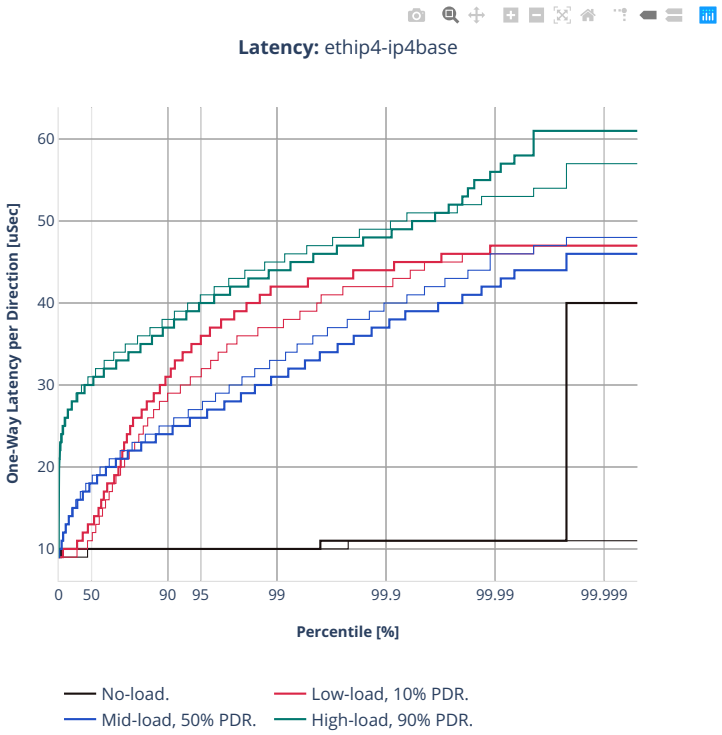

Latency: avf-ethip4-ip4scale2m
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

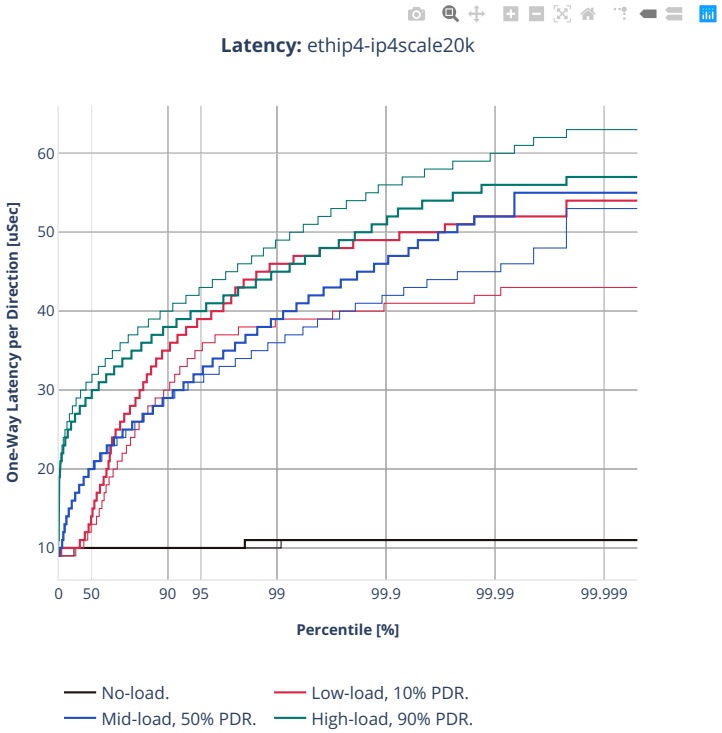

Latency: avf-ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

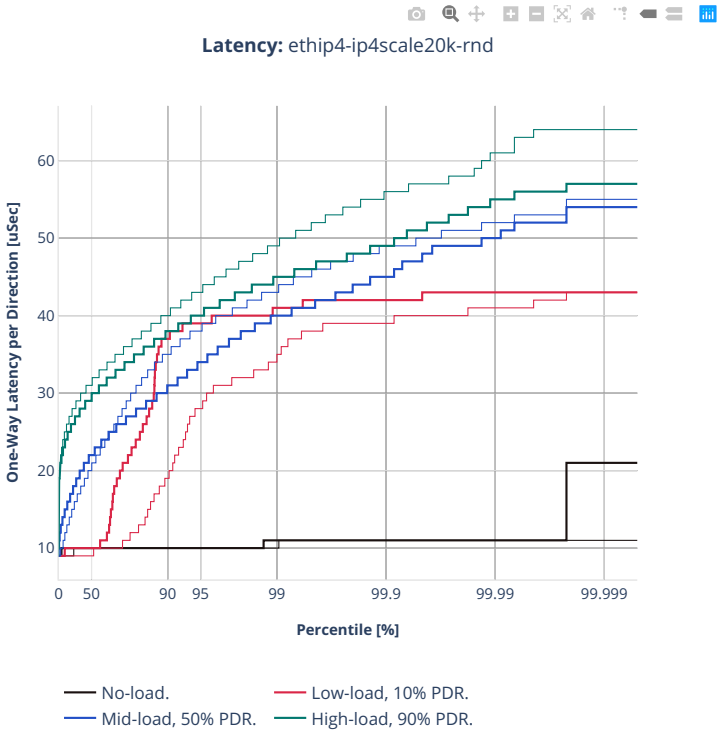

Latency: avf-ethip4-ip4scale200k-rnd
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4scale2m-rnd
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

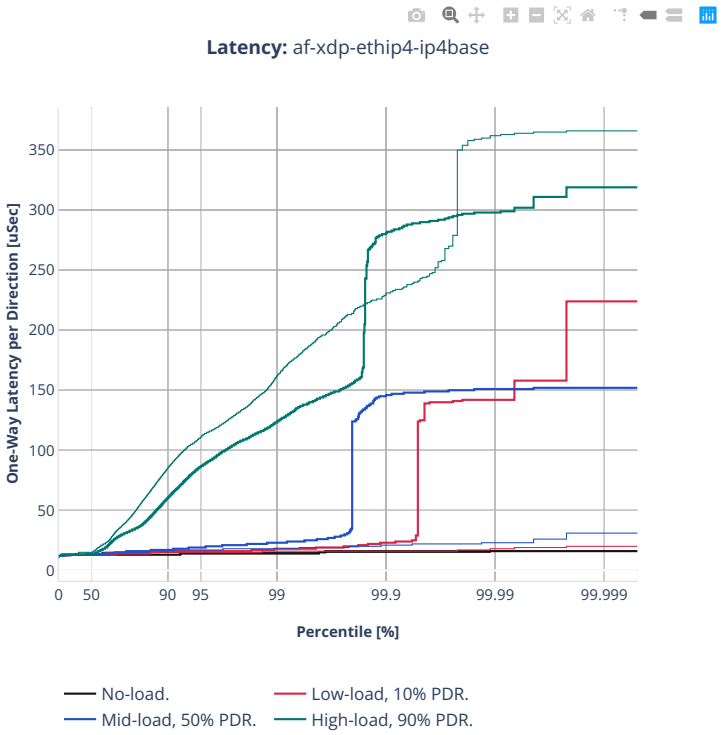
64b-2t1c-ip4routing-base-scale-dpdk

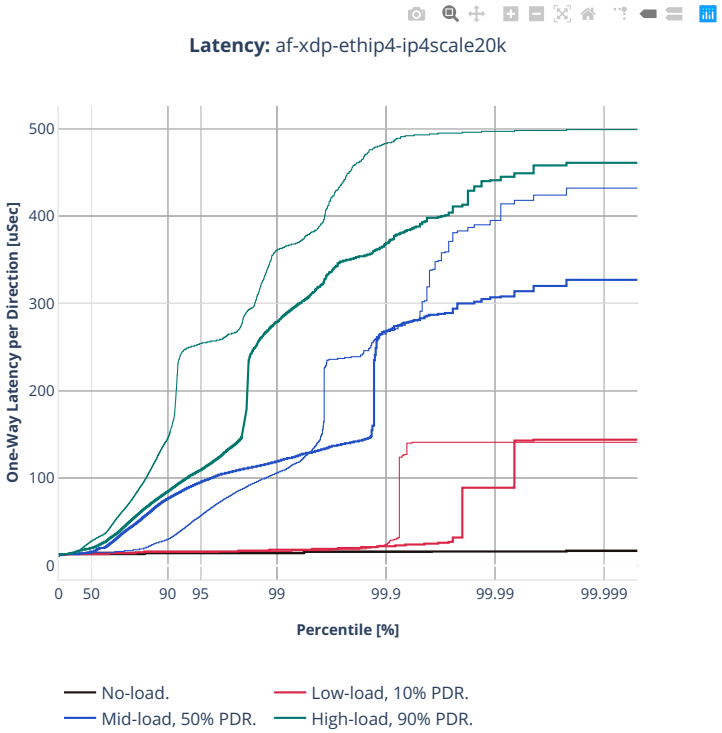


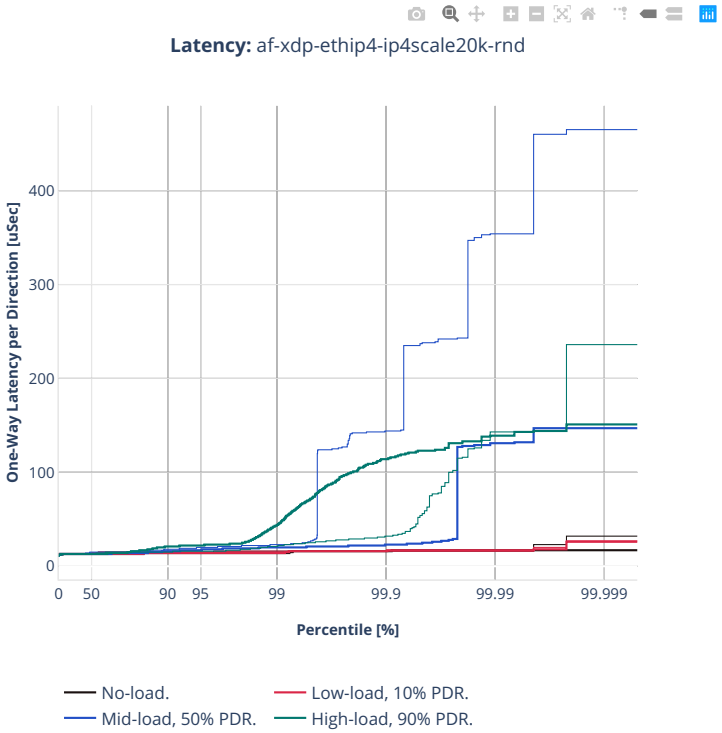

Latency: ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

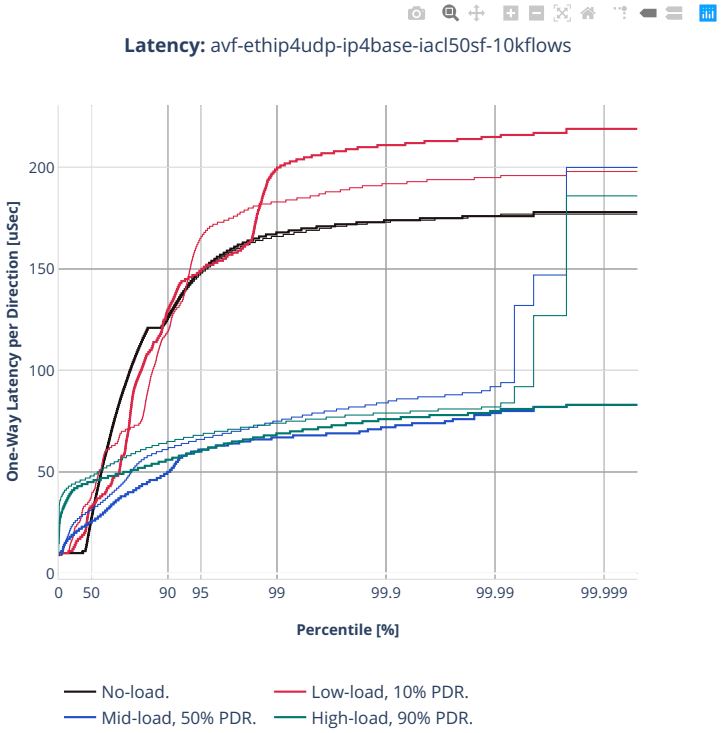
64b-2t1c-ip4routing-base-scale-af-xdp

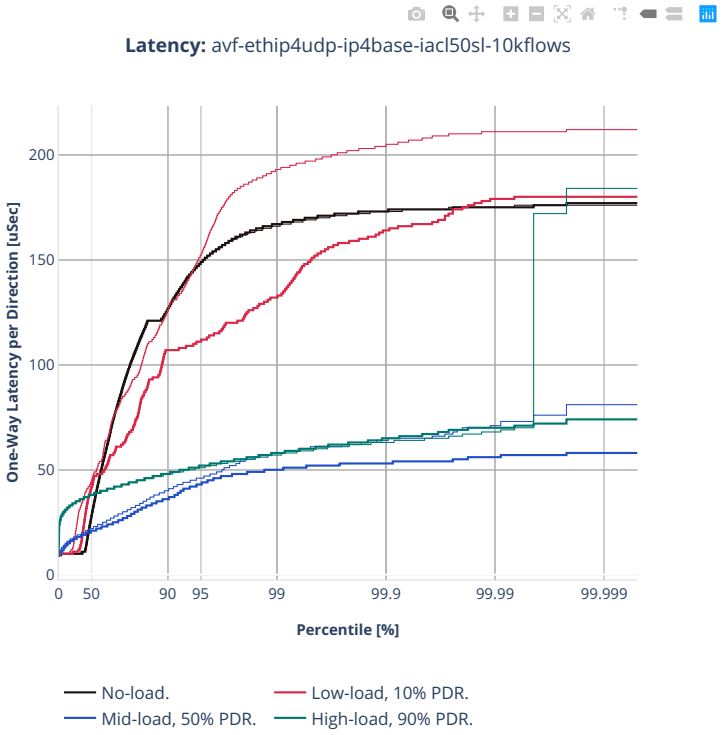


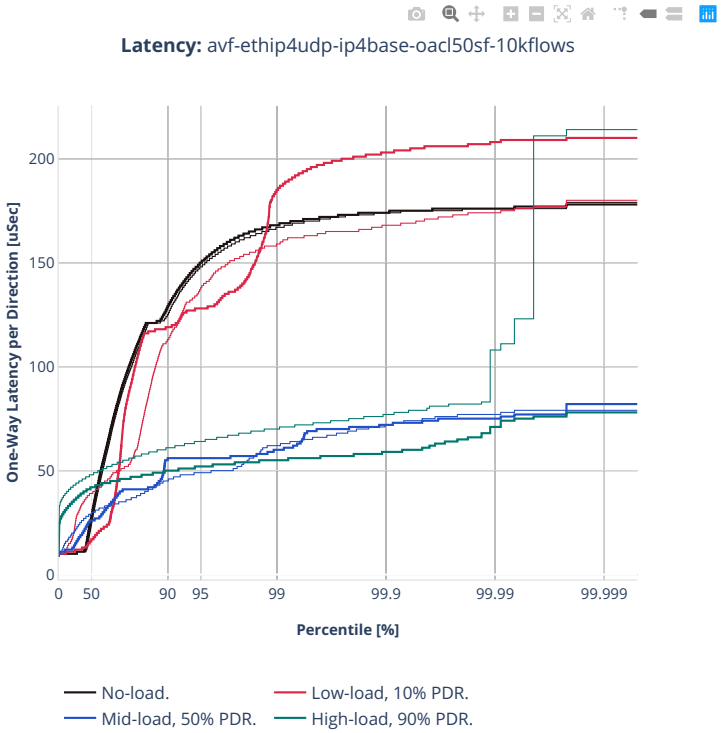

Latency: af-xdp-ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: af-xdp-ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

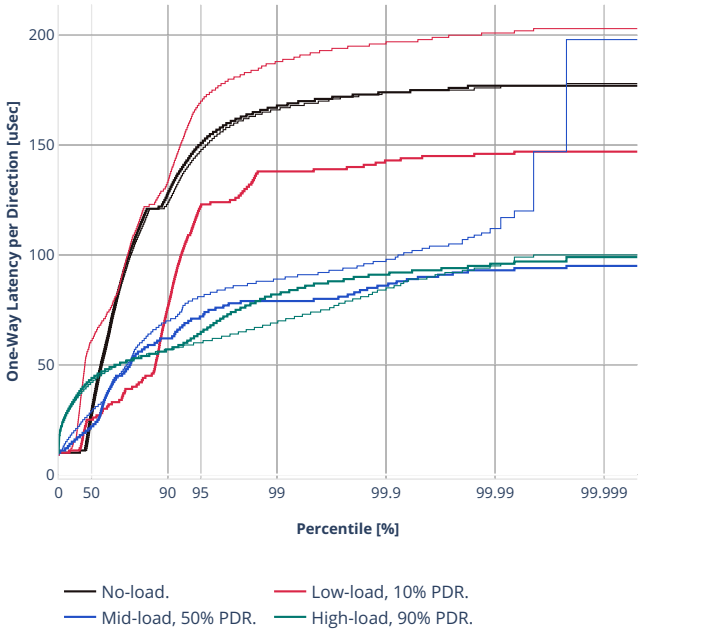
64b-2t1c-ip4routing-features-avf




Latency: avf-ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

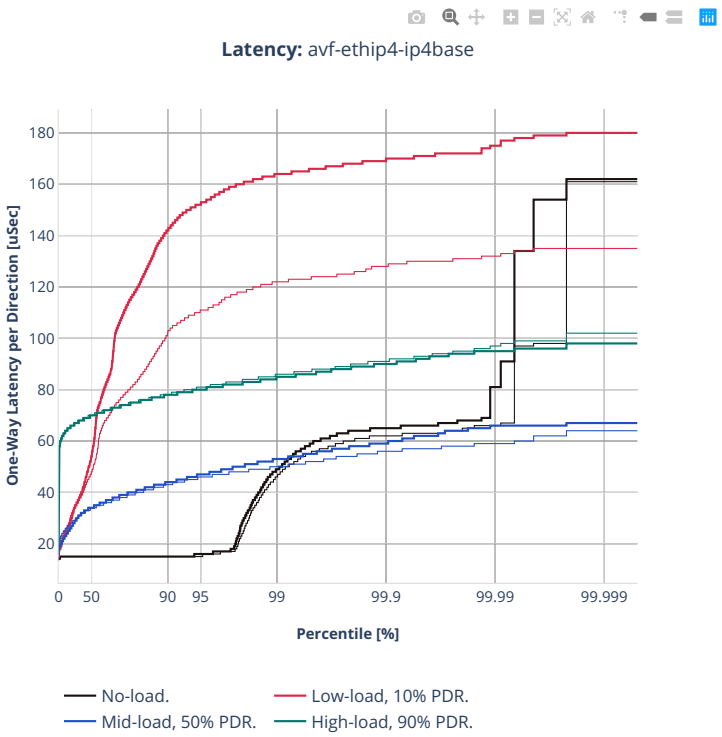

Latency: avf-ethip4udp-ip4base-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

Latency: avf-ethip4udp-ip4base-oacl50sl-10kflows

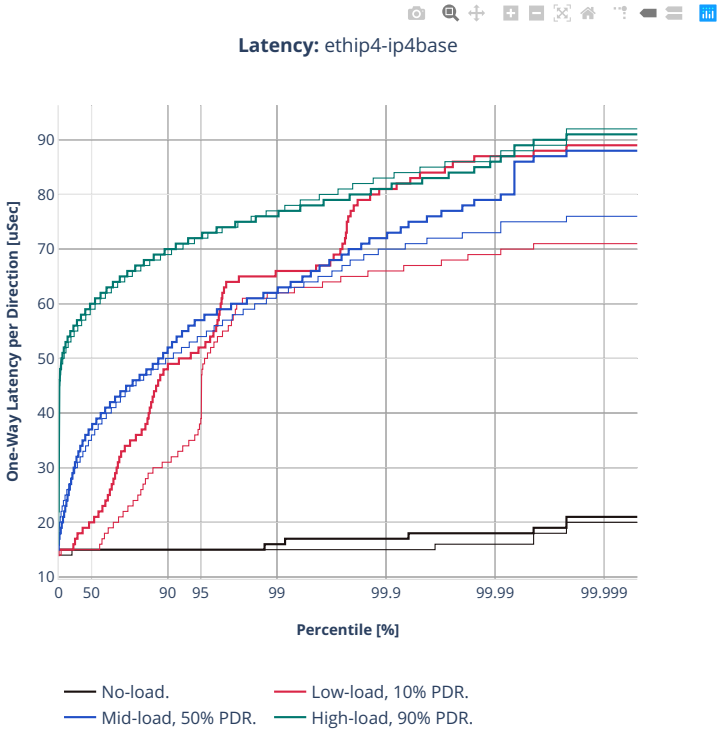


3n-icx-xxv710

64b-2t1c-ip4routing-base-avf

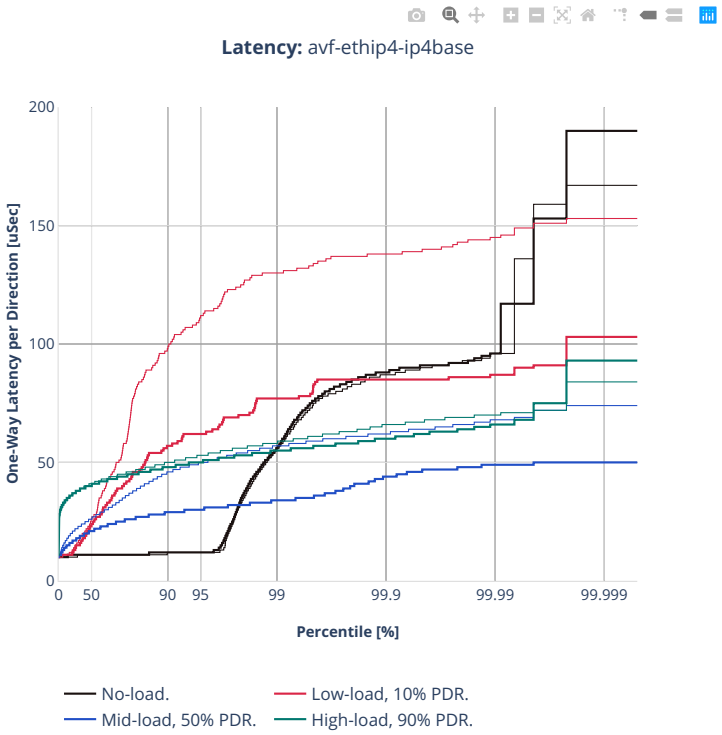


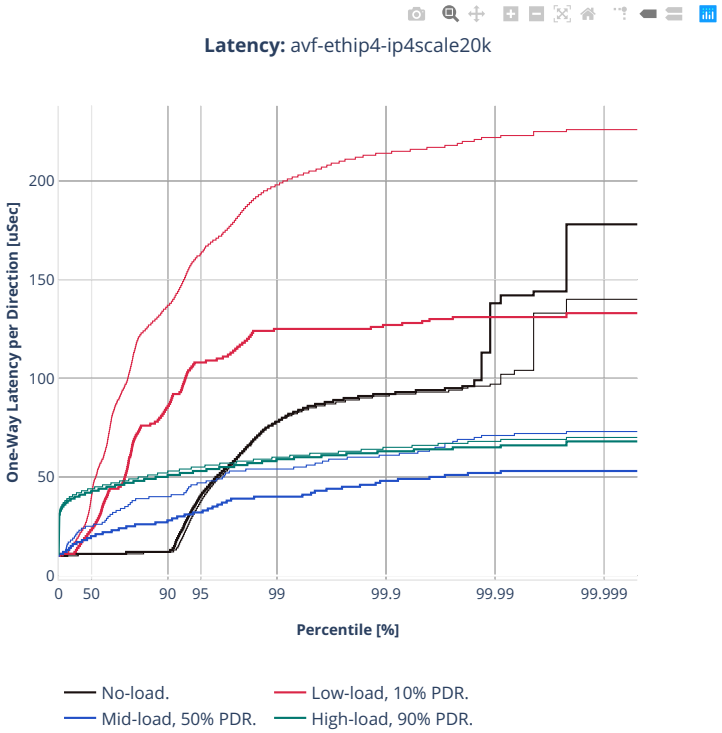
64b-2t1c-ip4routing-base-dpdk

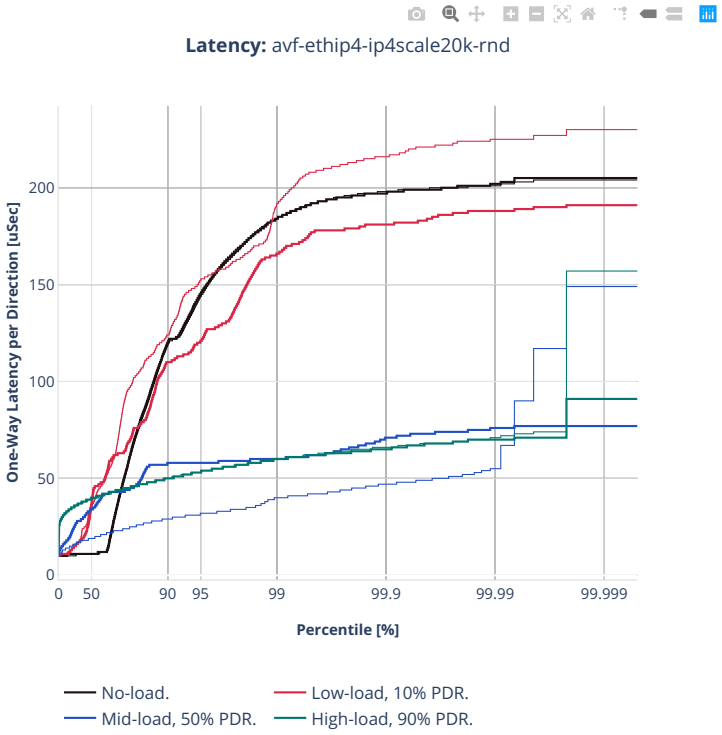


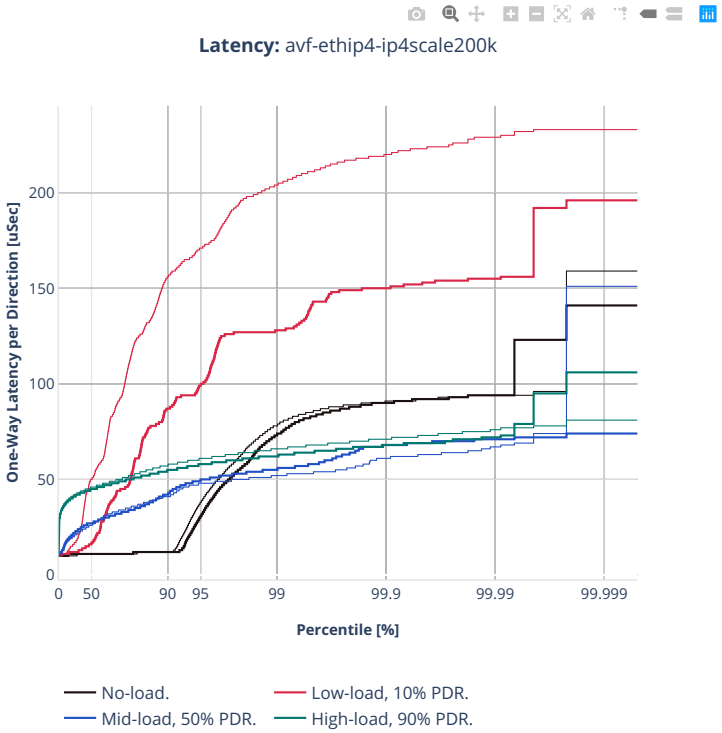
2n-clx-xxv710

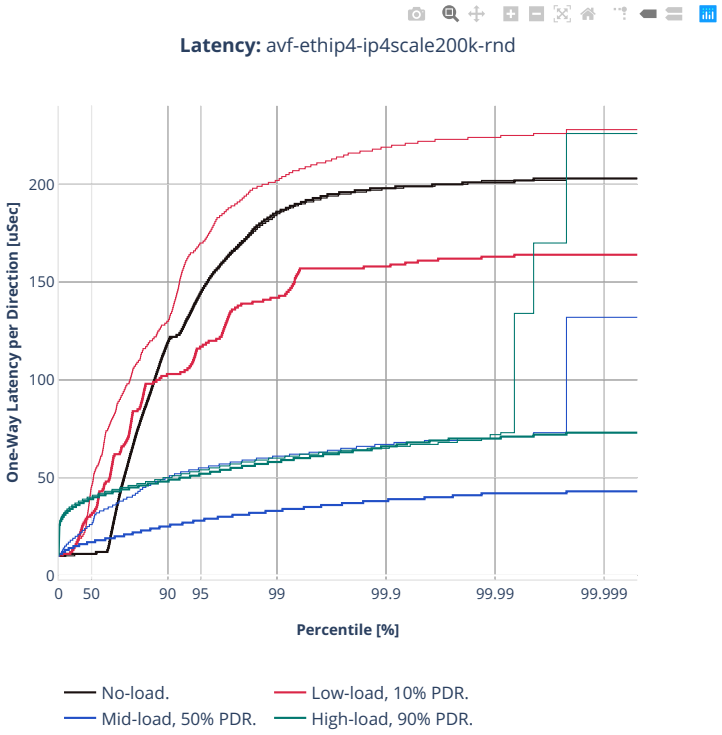
64b-2t1c-ip4routing-base-scale-avf

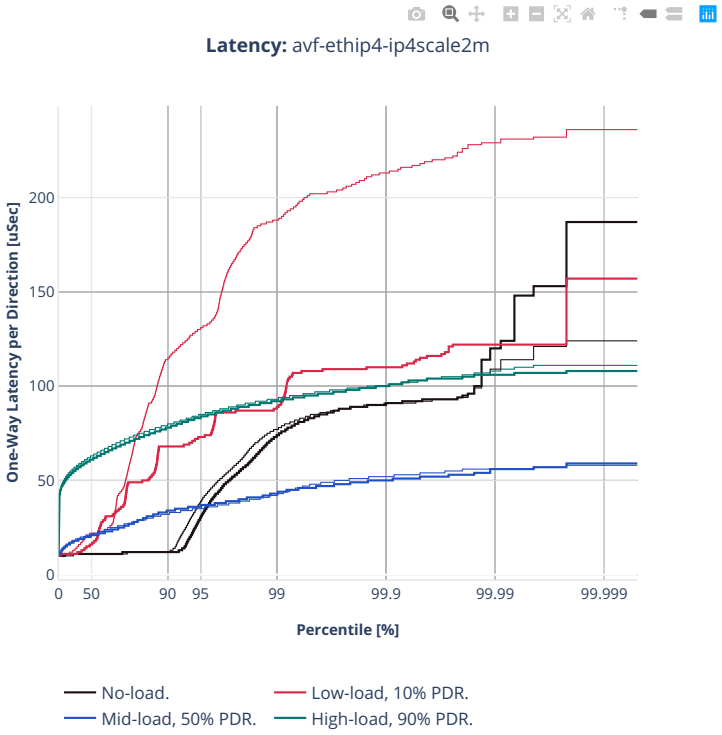


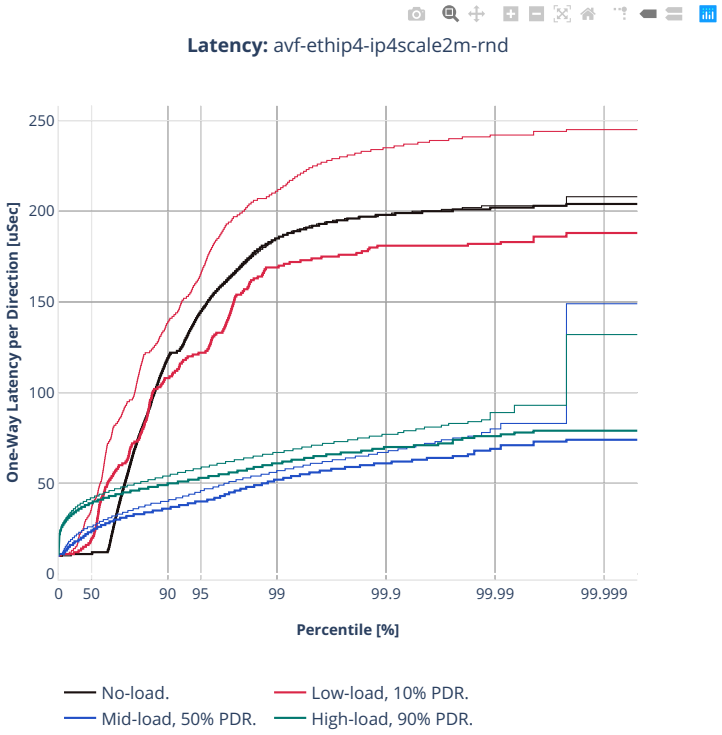

Latency: avf-ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

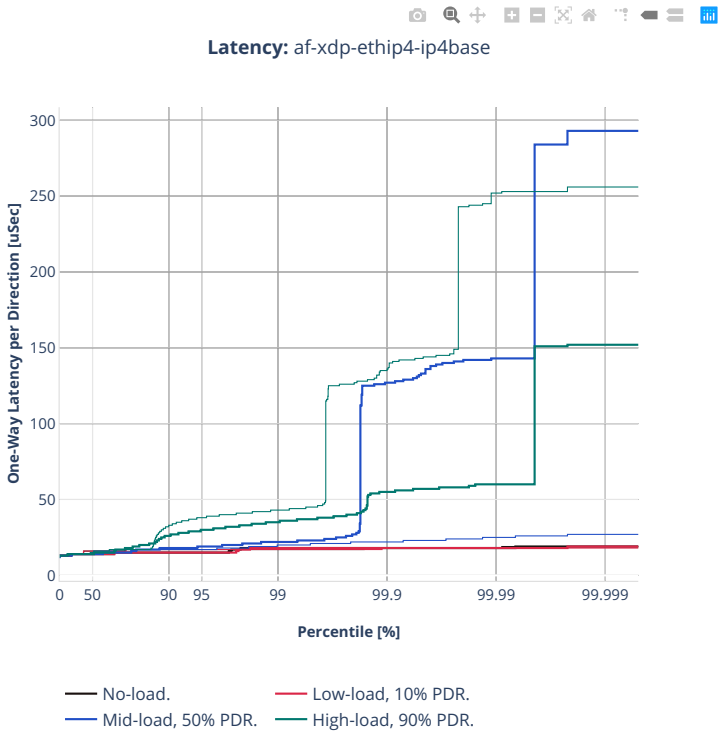

Latency: avf-ethip4-ip4scale200k
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

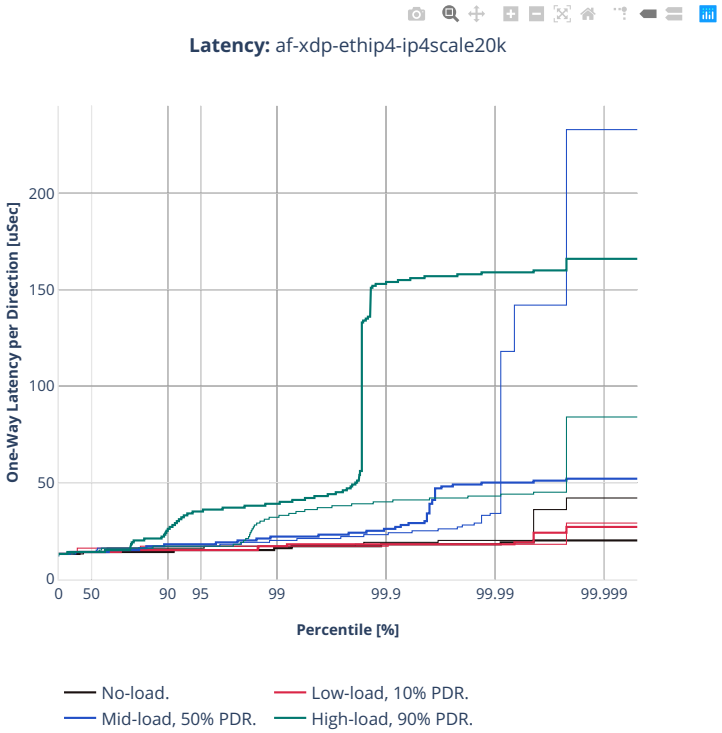

Latency: avf-ethip4-ip4scale200k-rnd
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

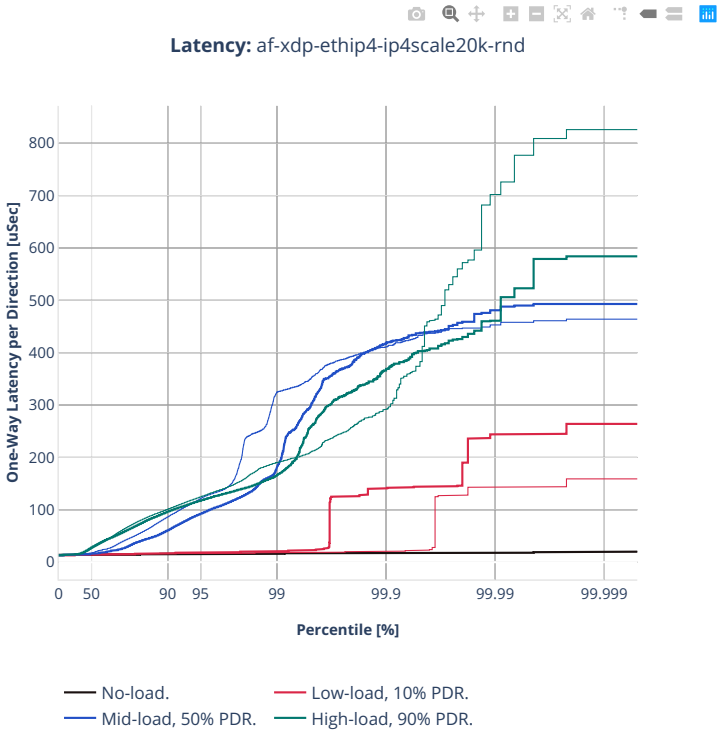

Latency: avf-ethip4-ip4scale2m
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4scale2m-rnd
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

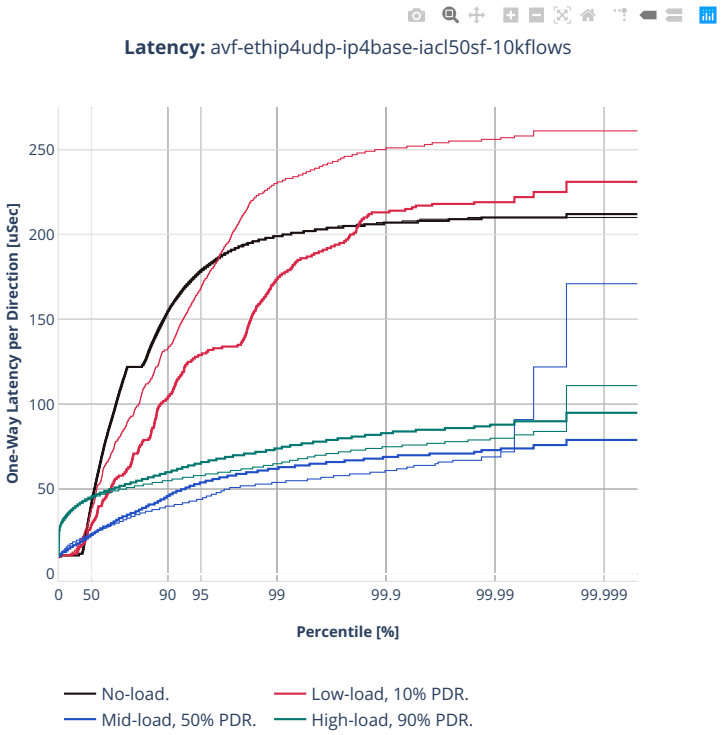
64b-2t1c-ip4routing-base-scale-af-xdp

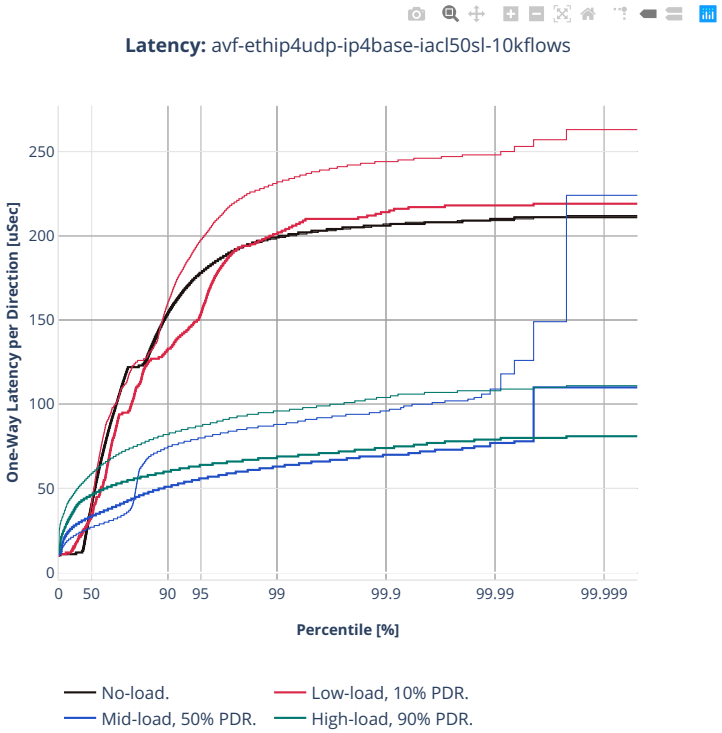


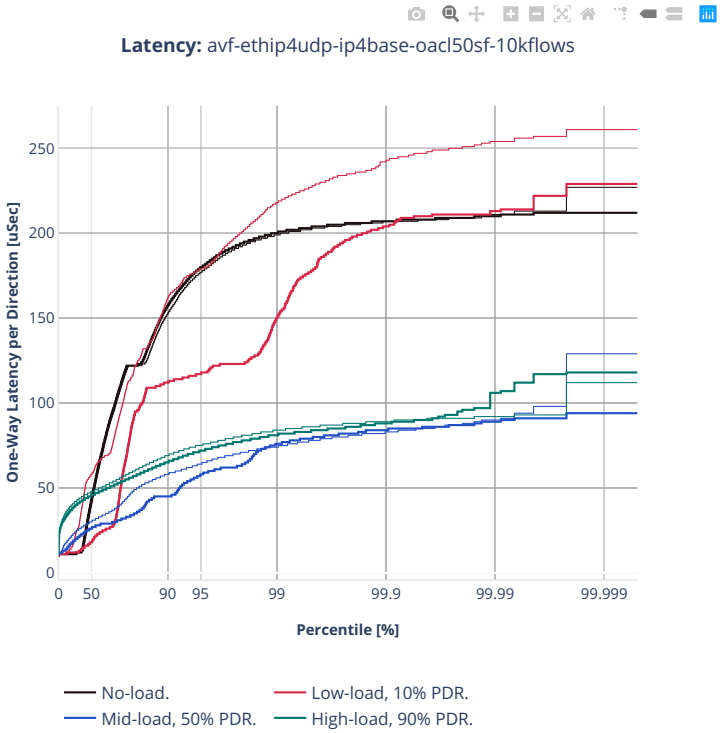

Latency: af-xdp-ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: af-xdp-ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
800
700
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

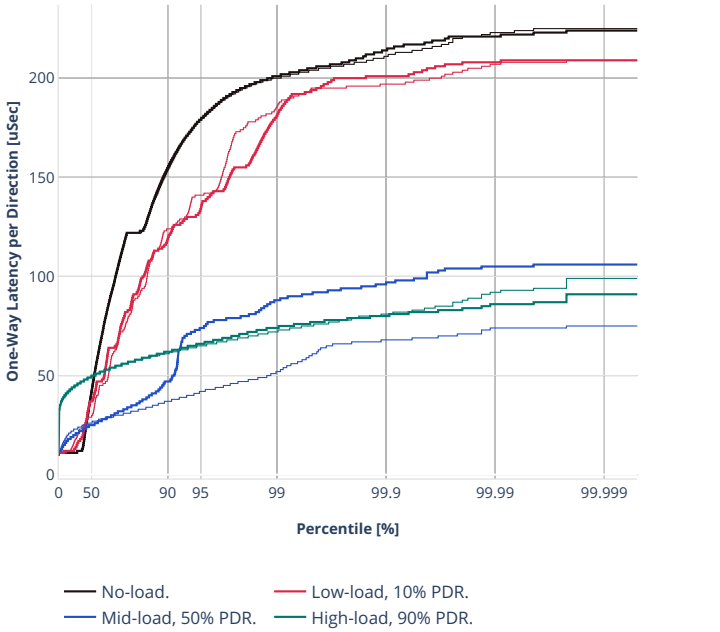
64b-2t1c-ip4routing-features-avf



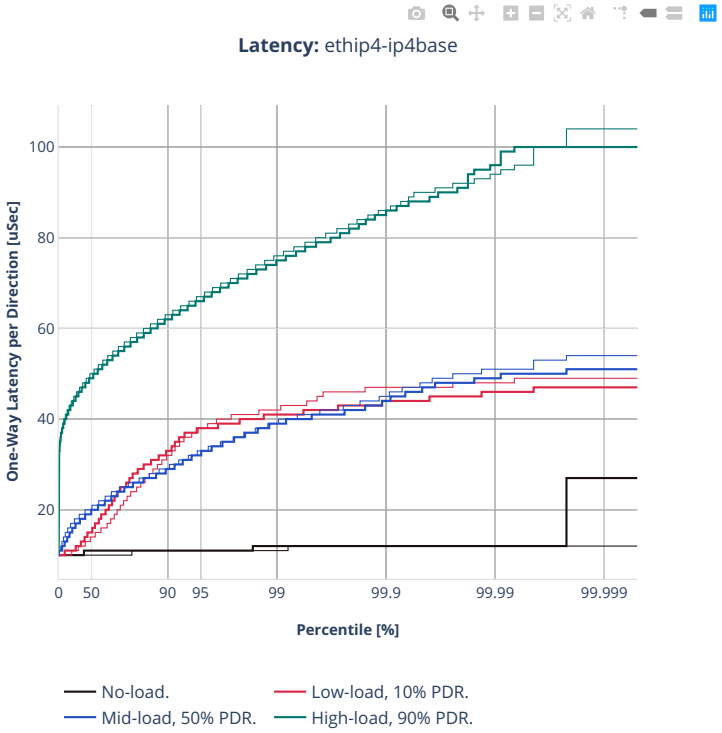

Latency: avf-ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

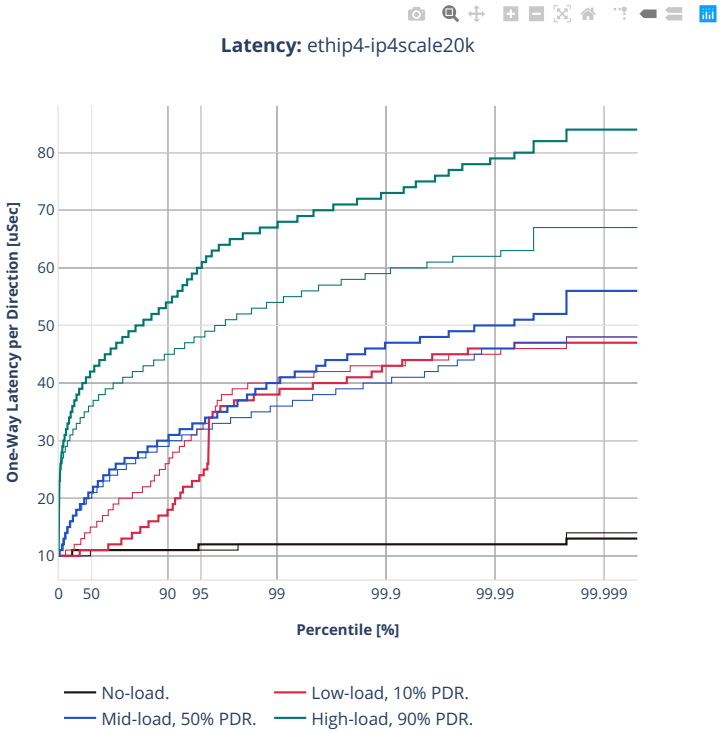

Latency: avf-ethip4udp-ip4base-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

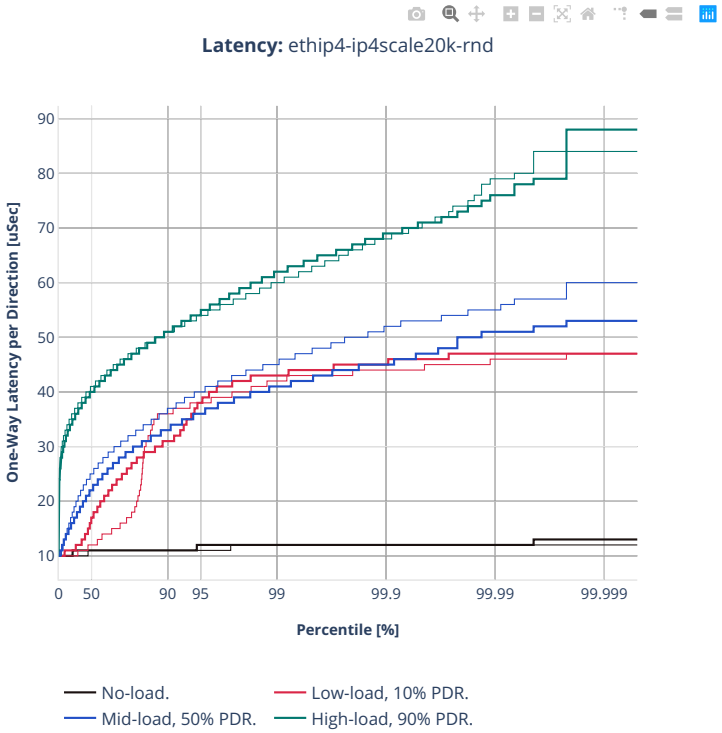
Latency: avf-ethip4udp-ip4base-oacl50sl-10kflows



64b-2t1c-ip4routing-base-scale-dpdk

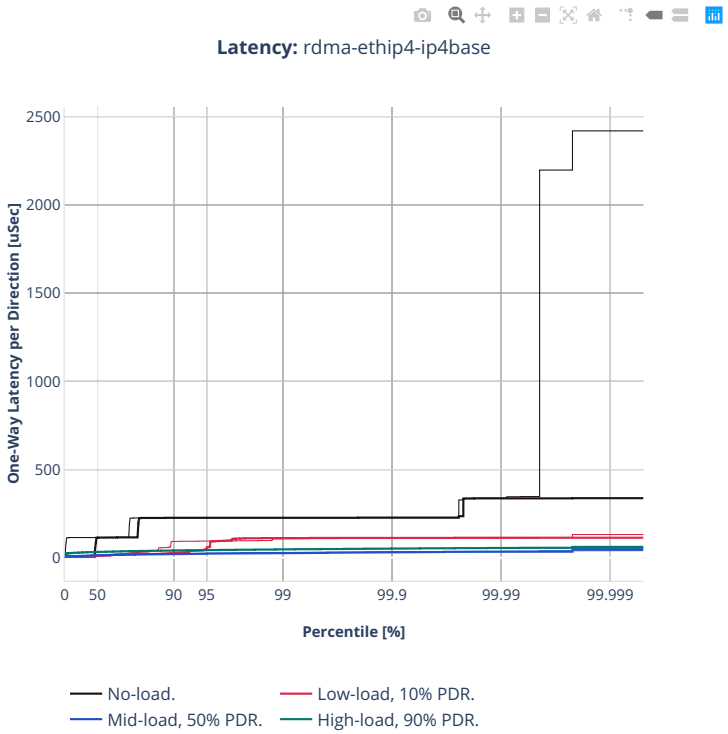


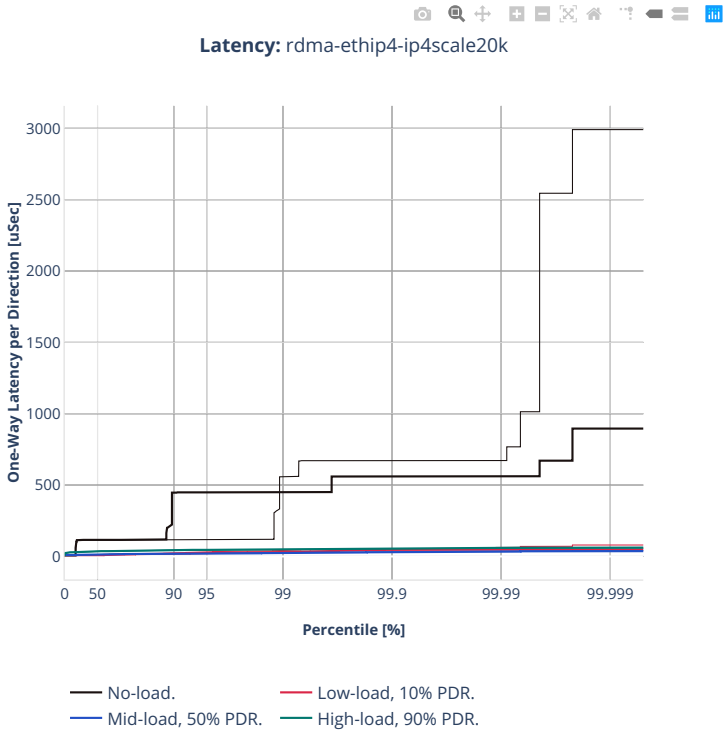

Latency: ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

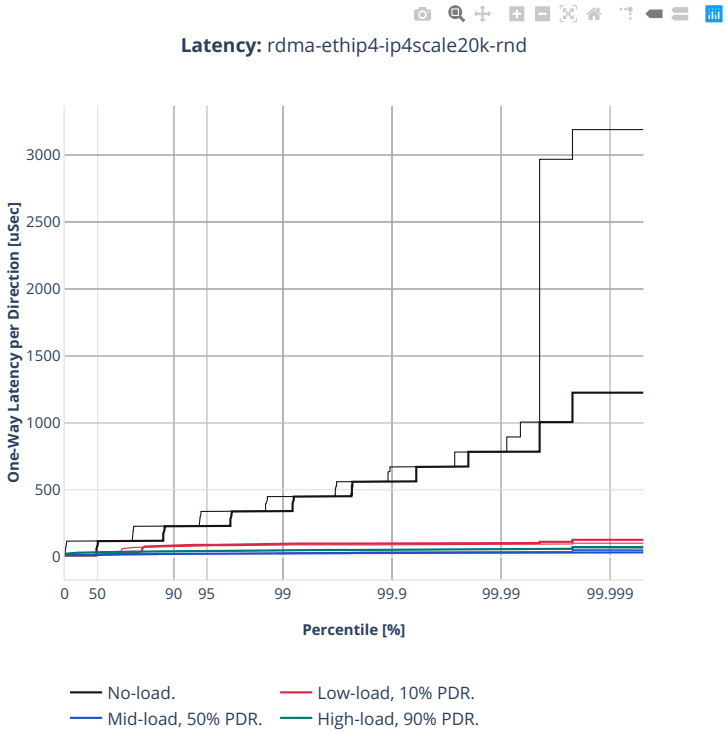

Latency: ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-clx-cx556a

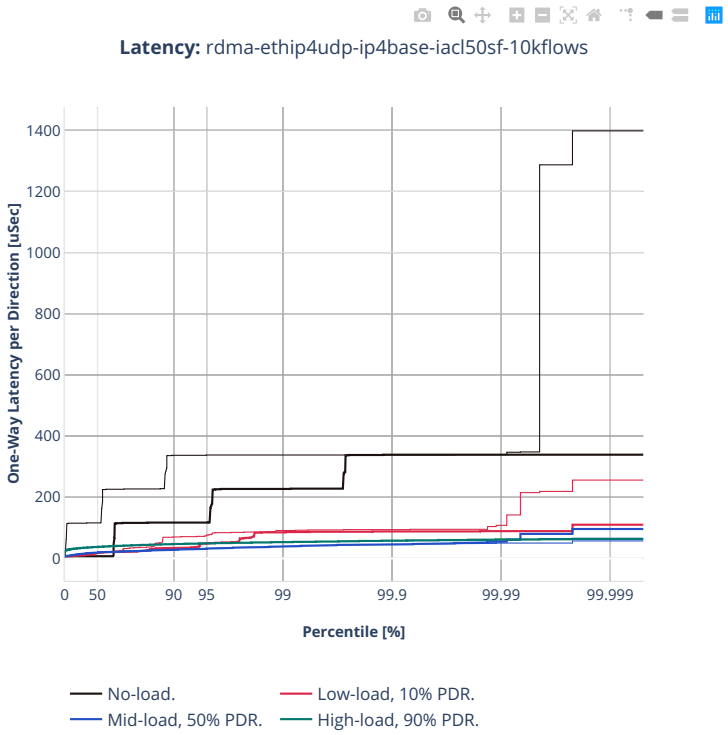
64b-2t1c-ip4routing-base-scale-rdma

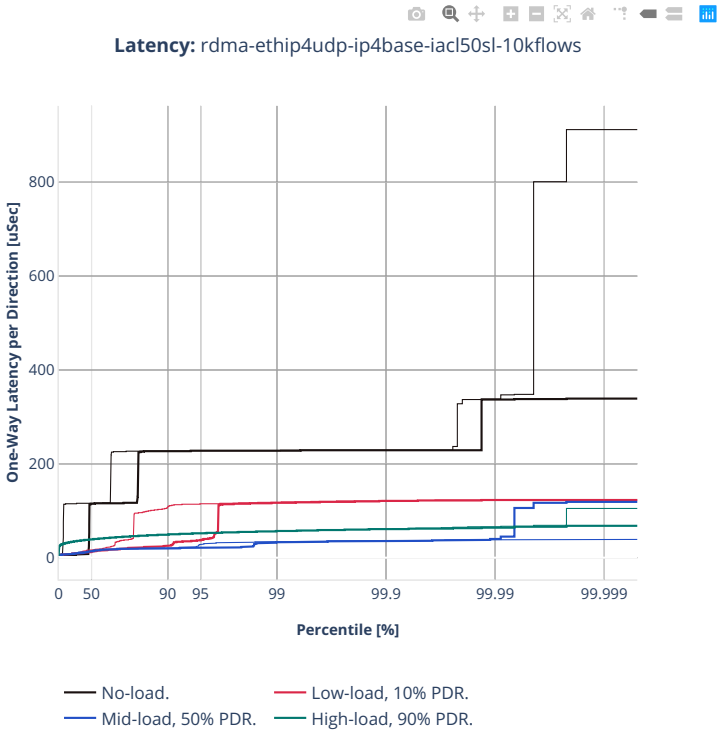


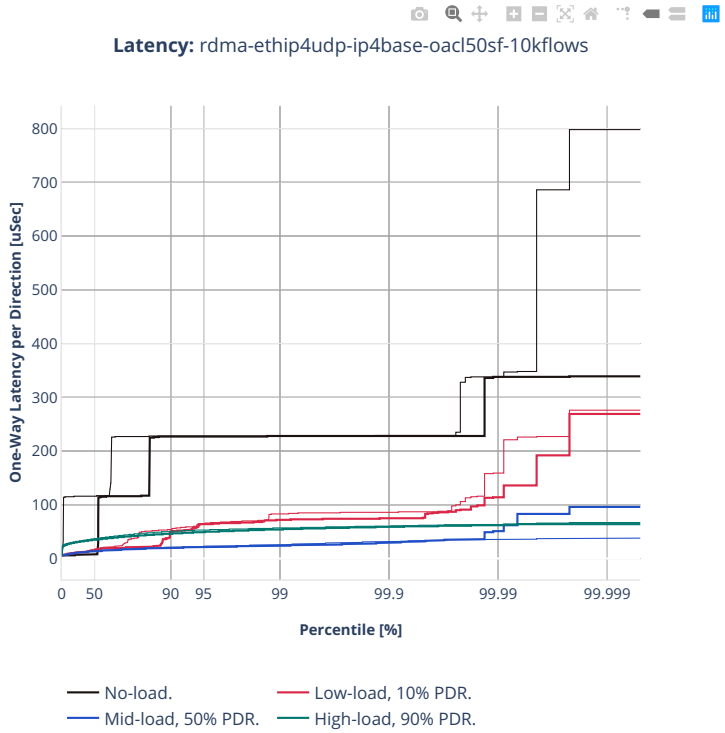

Latency: rdma-ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

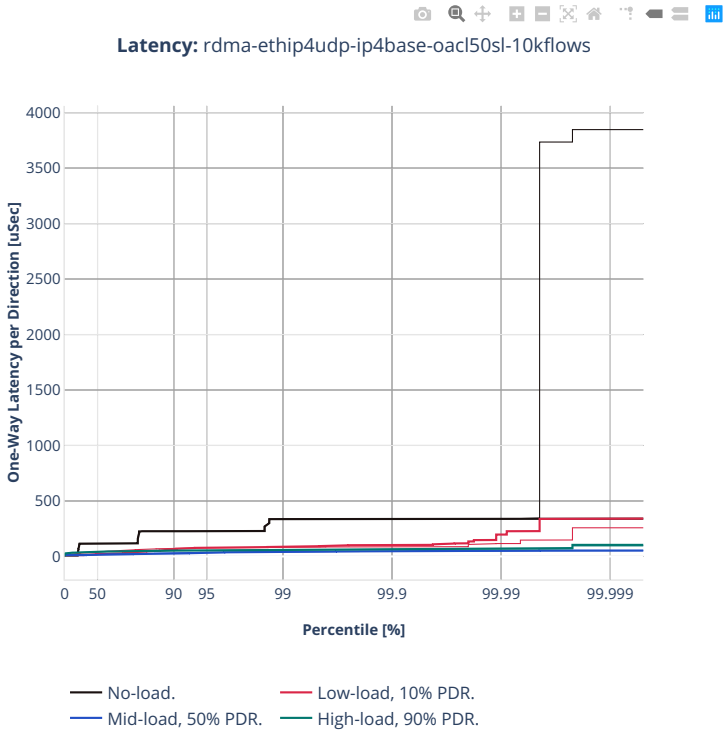

Latency: rdma-ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-ip4routing-features-rdma



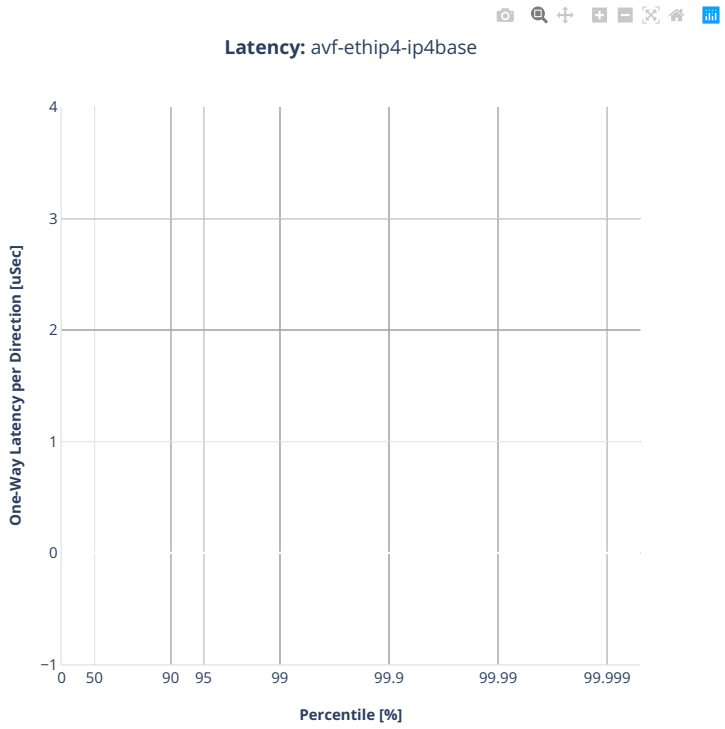

Latency: rdma-ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

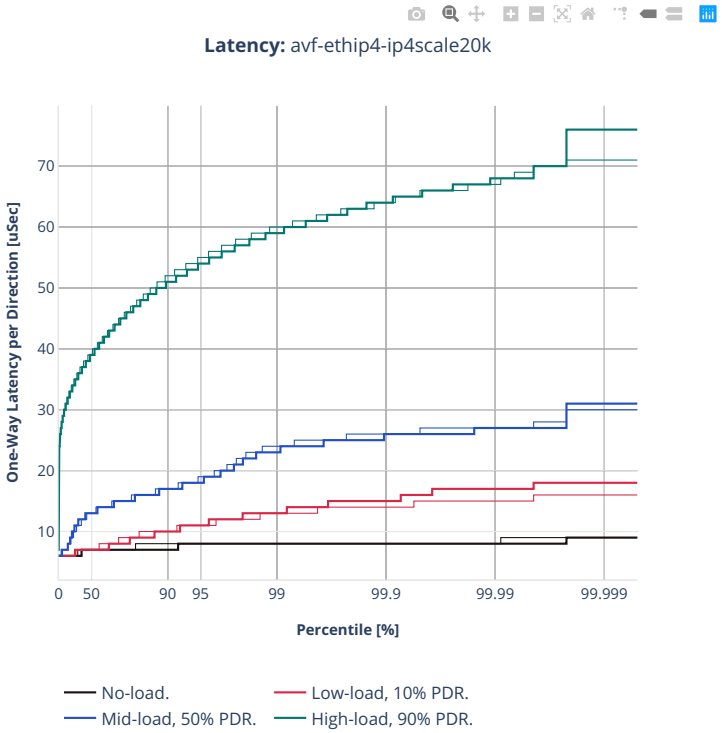

Latency: rdma-ethip4udp-ip4base-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
800
700
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

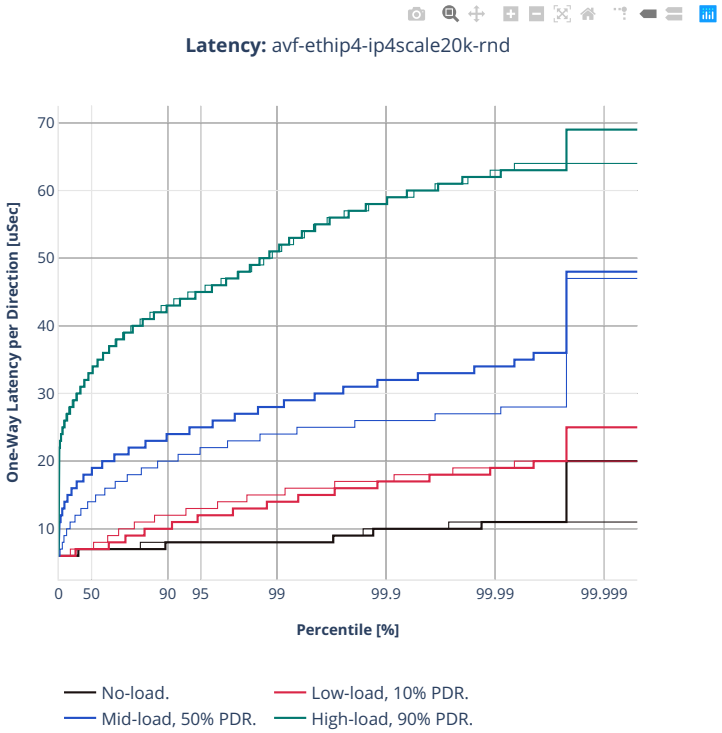

Latency: rdma-ethip4udp-ip4base-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
4000
3500
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-clx-e810cq

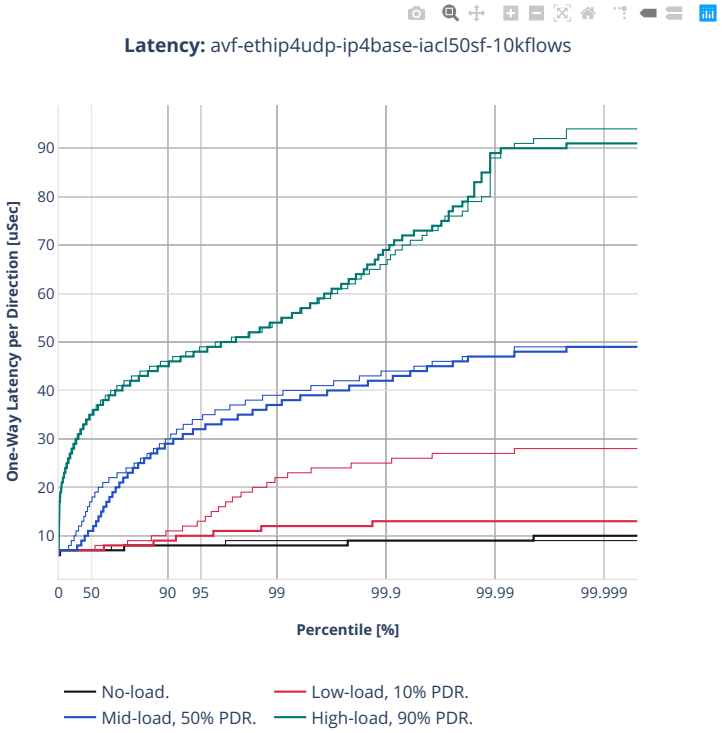
64b-2t1c-ip4routing-base-scale-avf

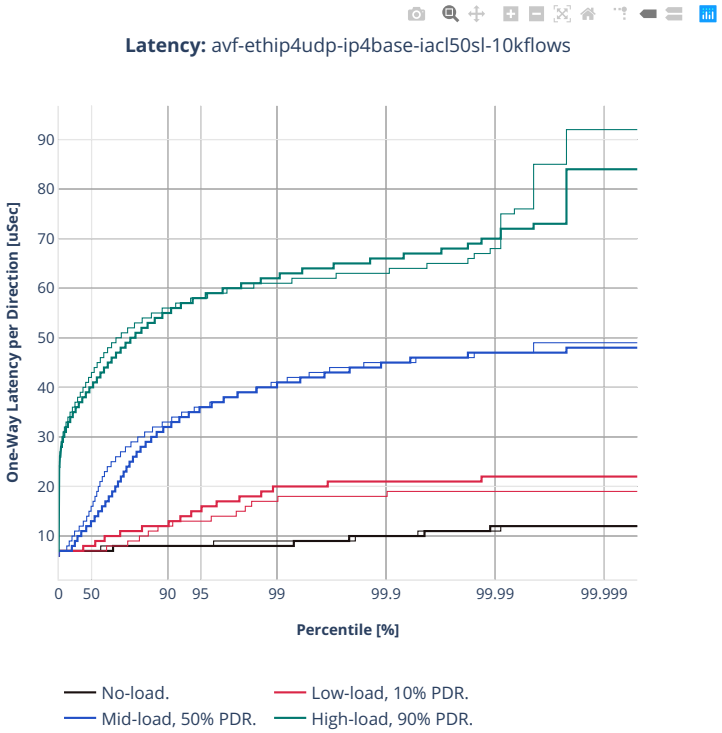


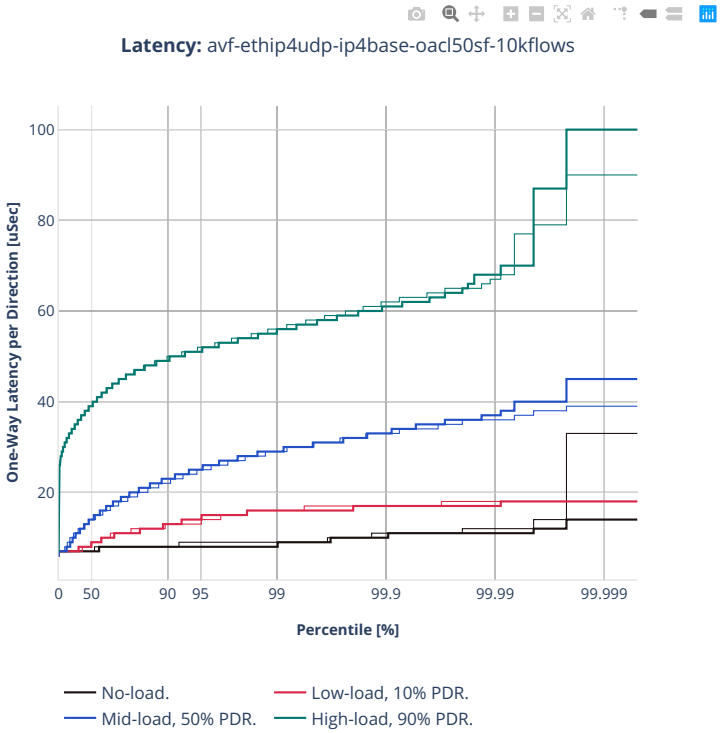

Latency: avf-ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

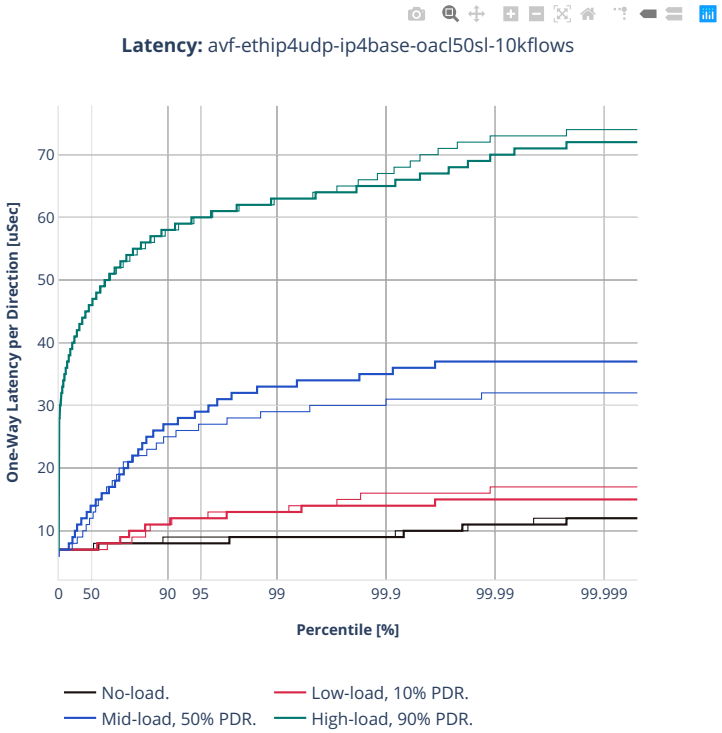

Latency: avf-ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-ip4routing-features-avf

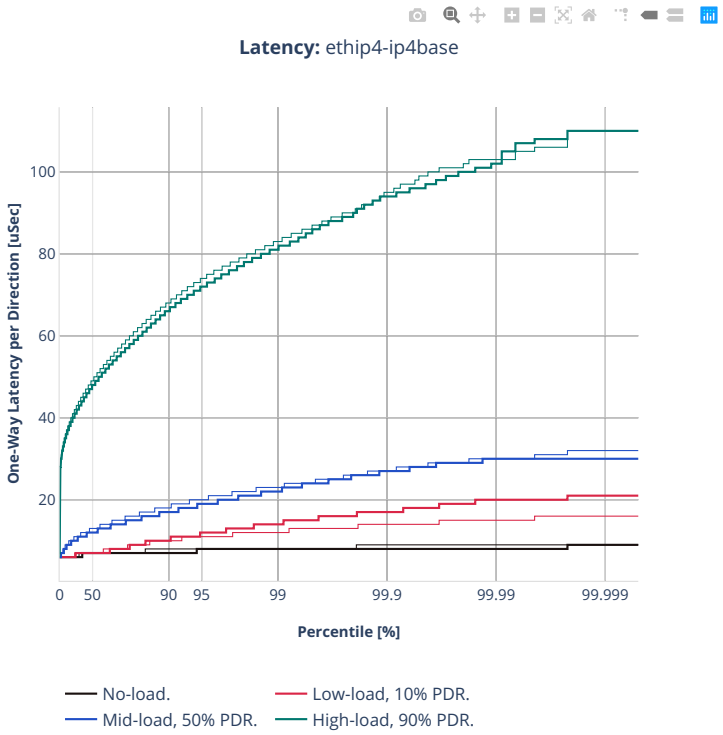


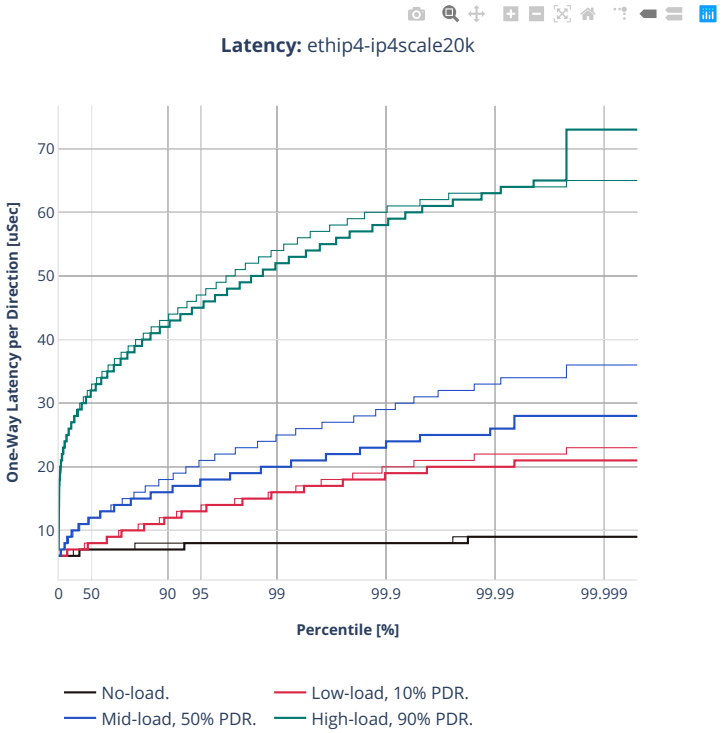

Latency: avf-ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

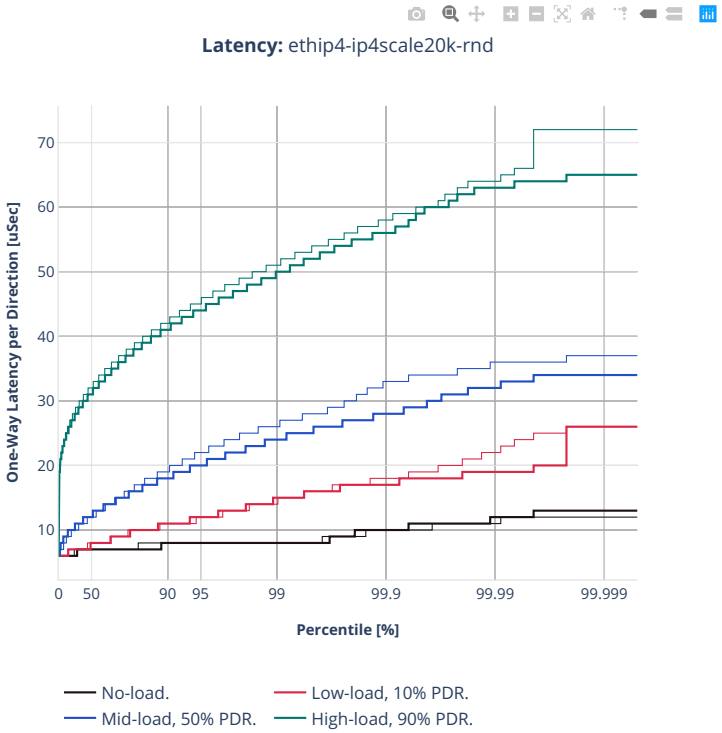

Latency: avf-ethip4udp-ip4base-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4udp-ip4base-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-ip4routing-base-scale-dpdk

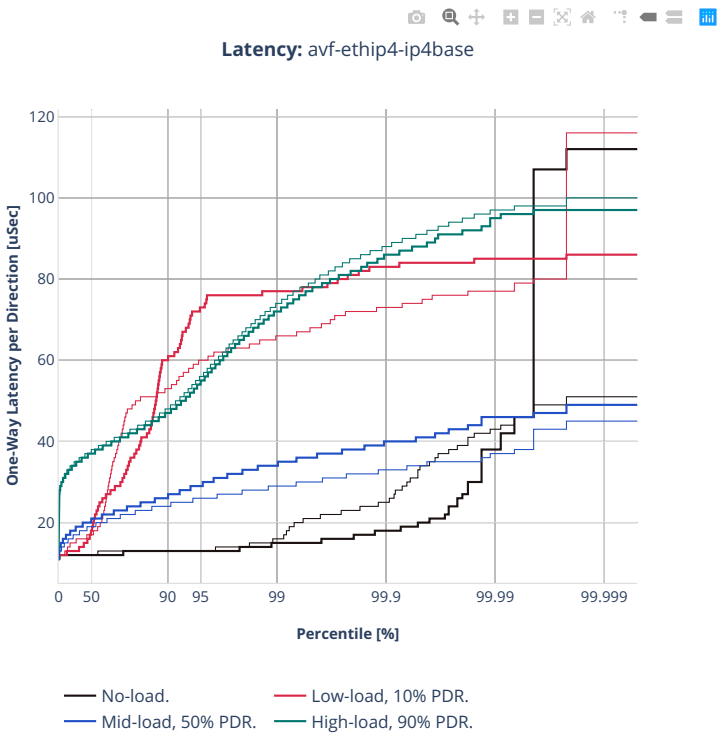


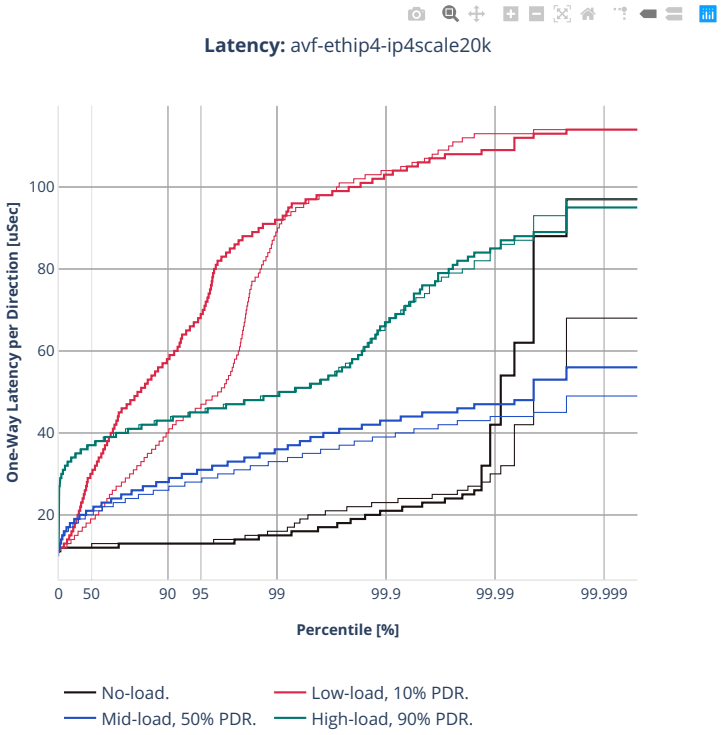

Latency: ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

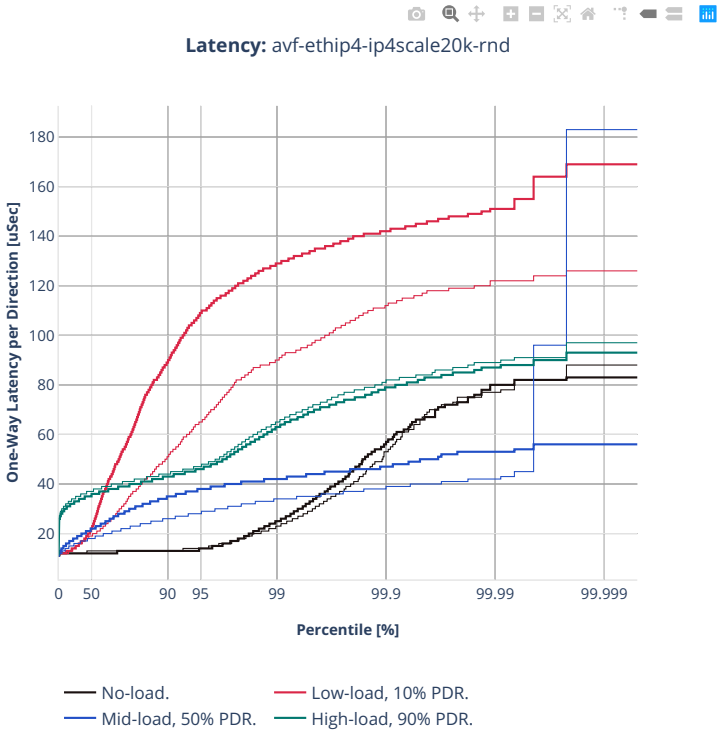

Latency: ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-zn2-xxv710

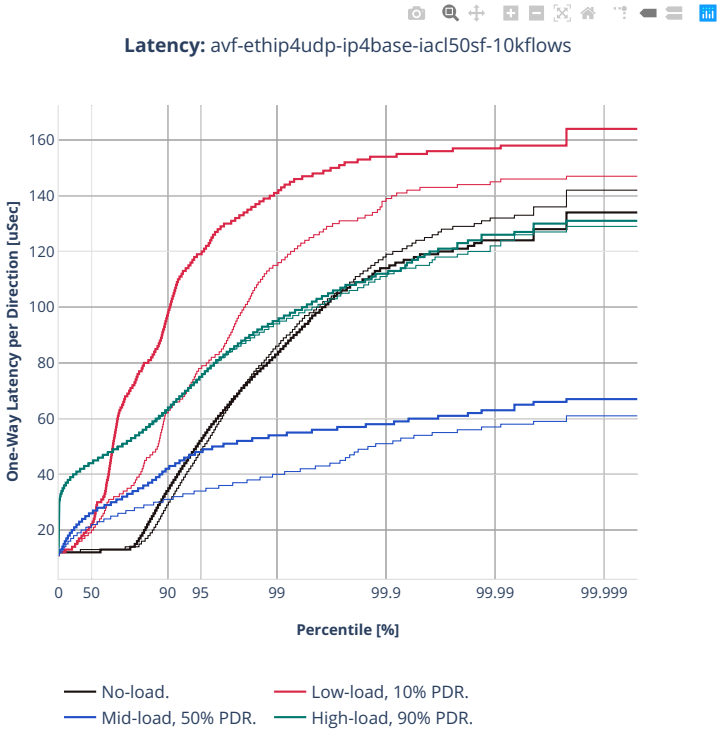
64b-2t1c-ip4routing-base-scale-avf

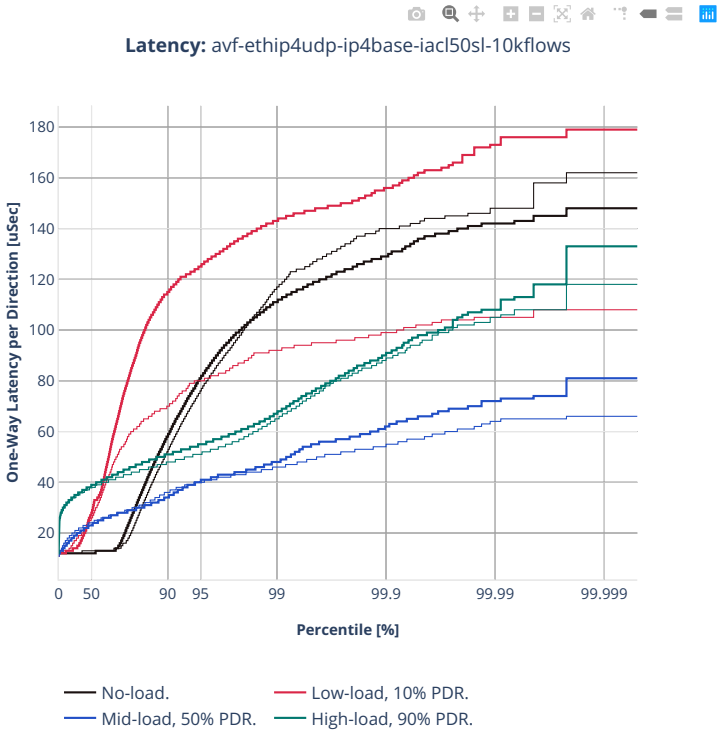



Latency: avf-ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

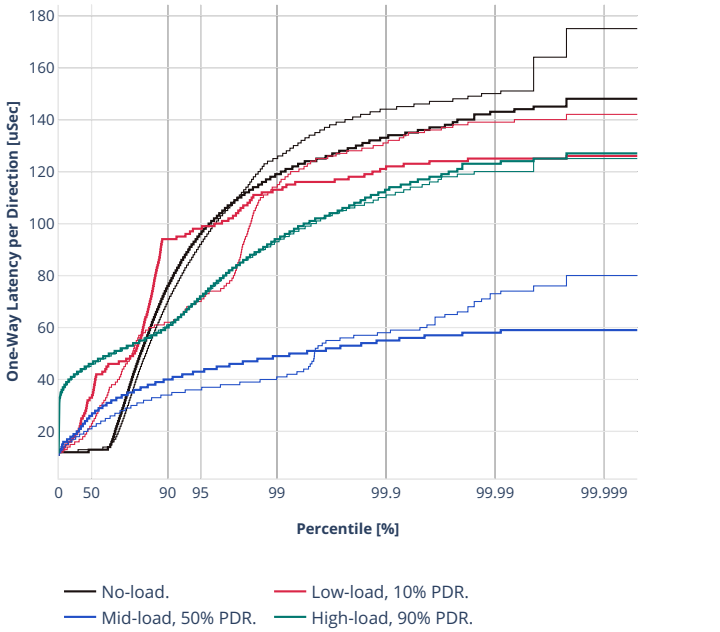

Latency: avf-ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

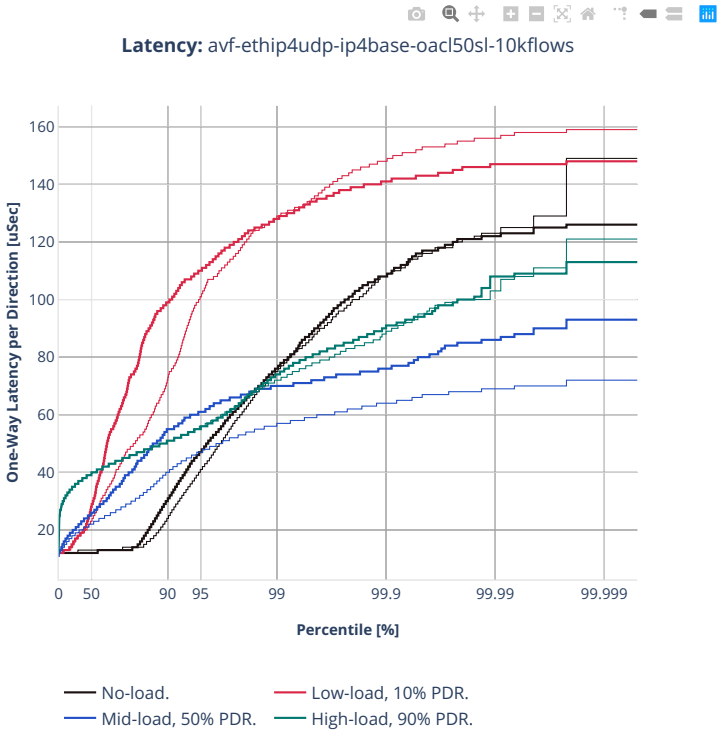
64b-2t1c-ip4routing-features-avf



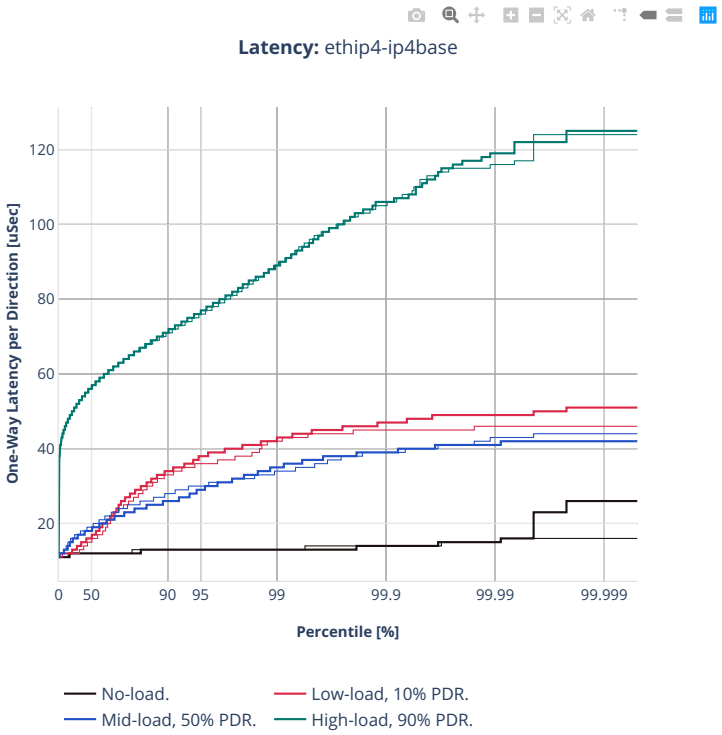

Latency: avf-ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

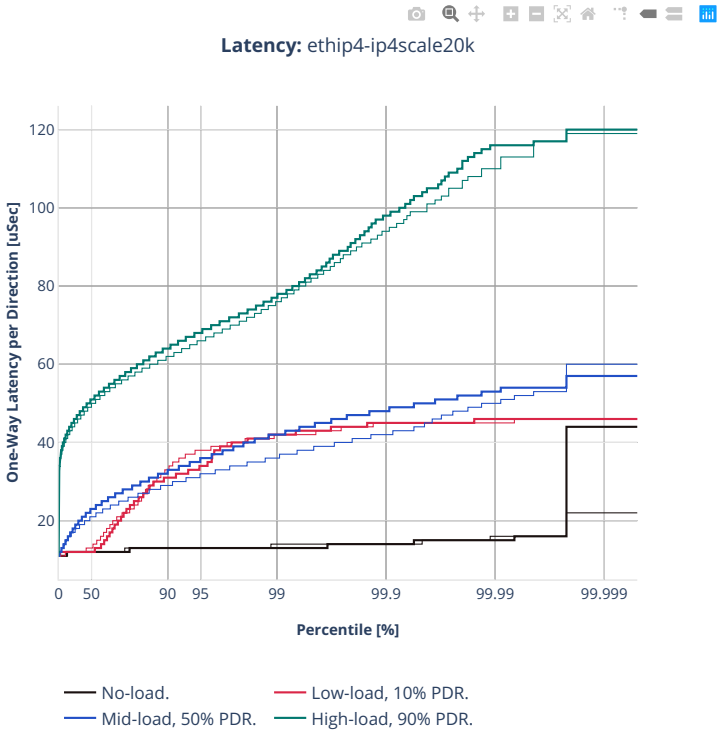
Latency: avf-ethip4udp-ip4base-oacl50sf-10kflows

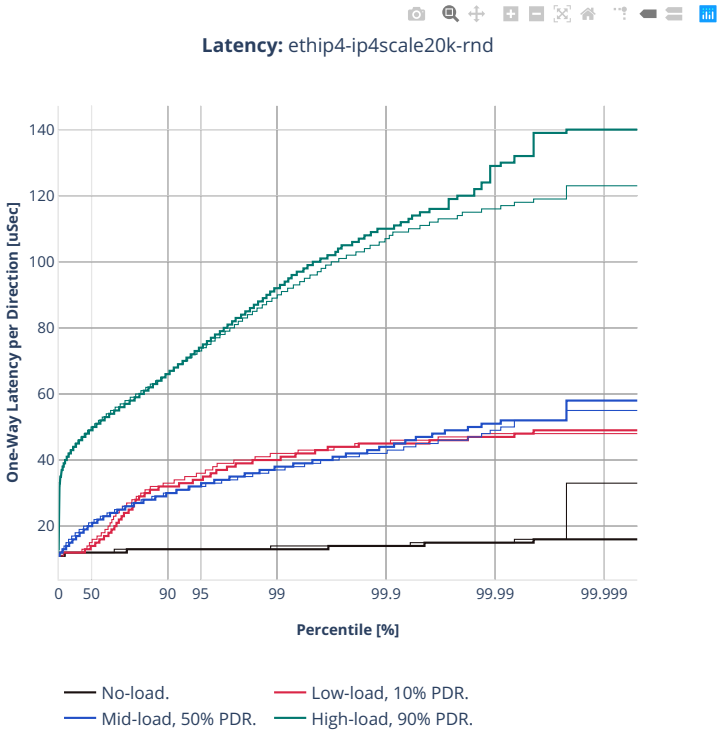



Latency: avf-ethip4udp-ip4base-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-ip4routing-base-scale-dpdk

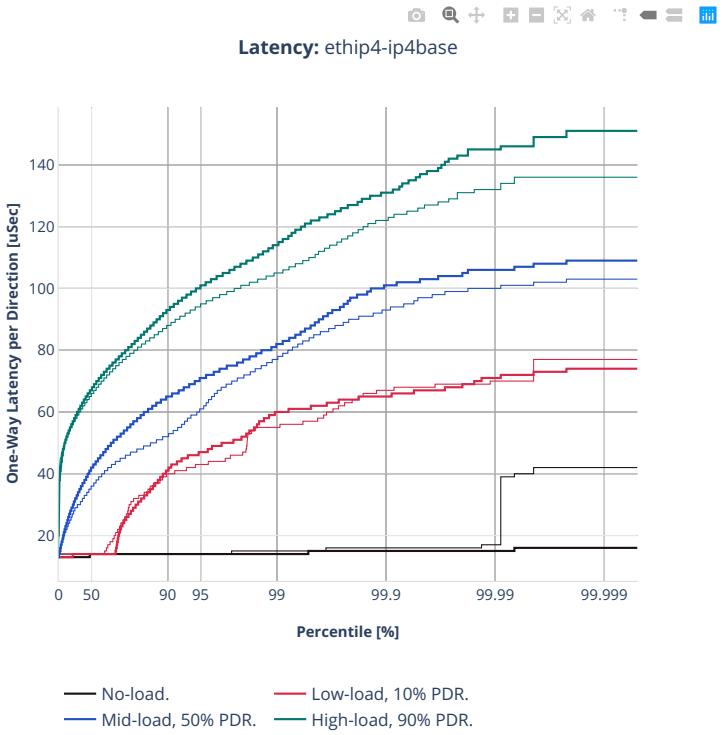


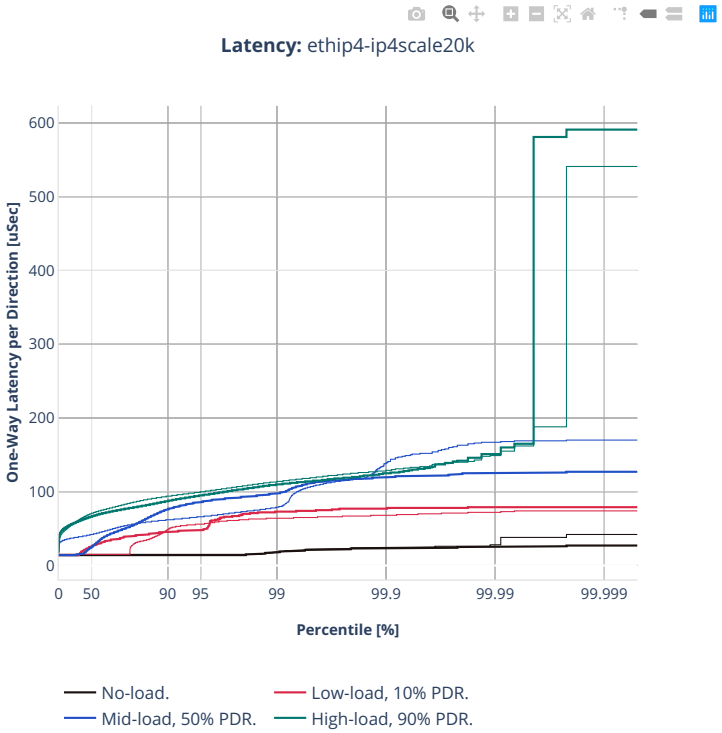

Latency: ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

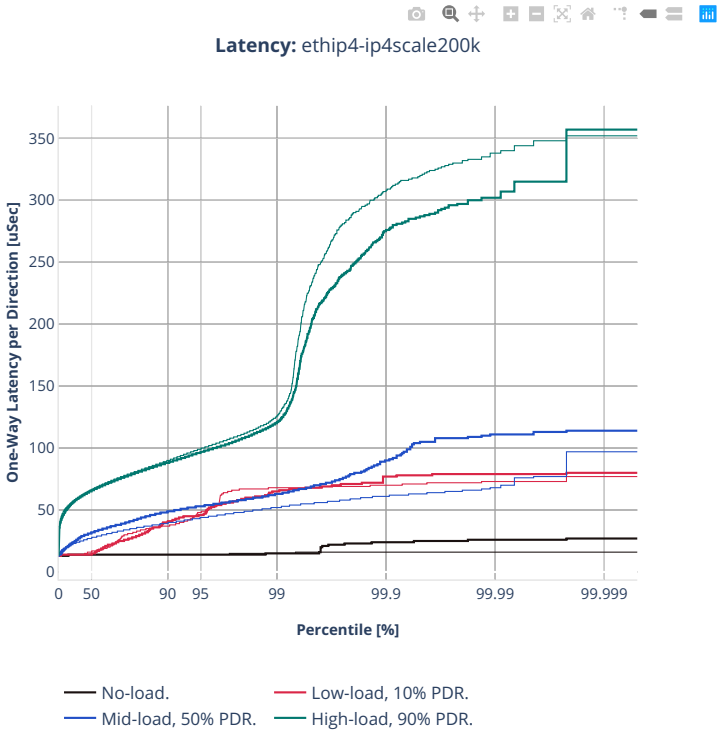

Latency: ethip4-ip4scale20k-rnd
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3n-alt-xl710

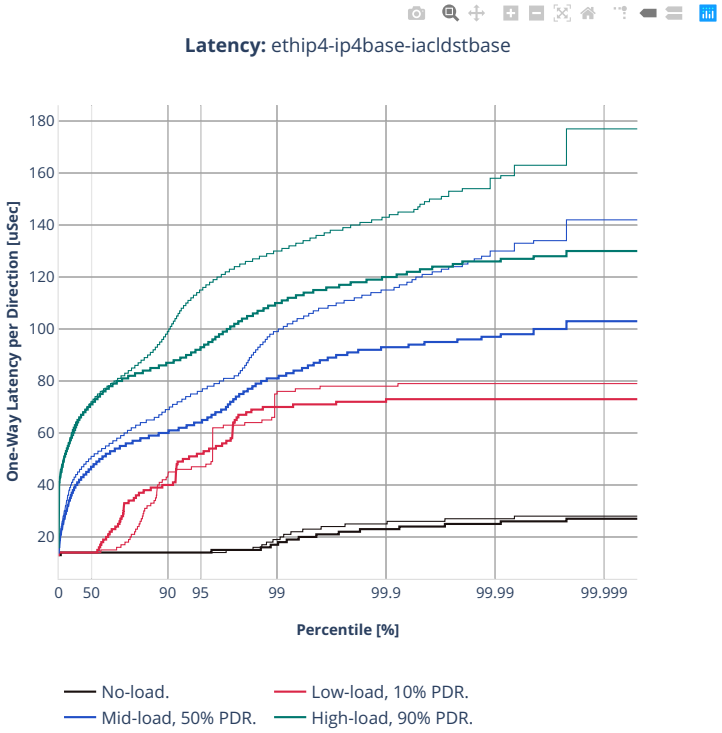
64b-1t1c-ip4routing-base-scale

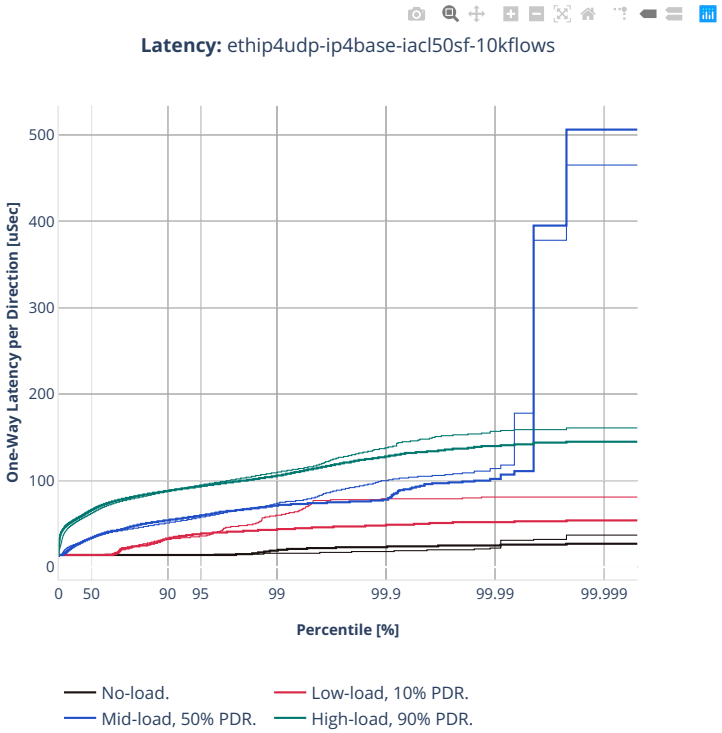


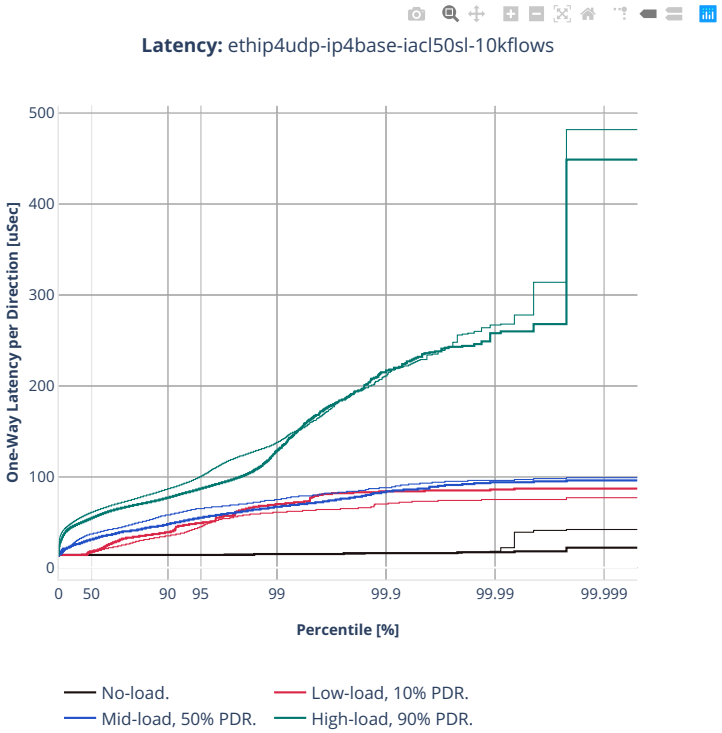

Latency: ethip4-ip4scale20k
One-Way Latency per Direction [uSec]
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

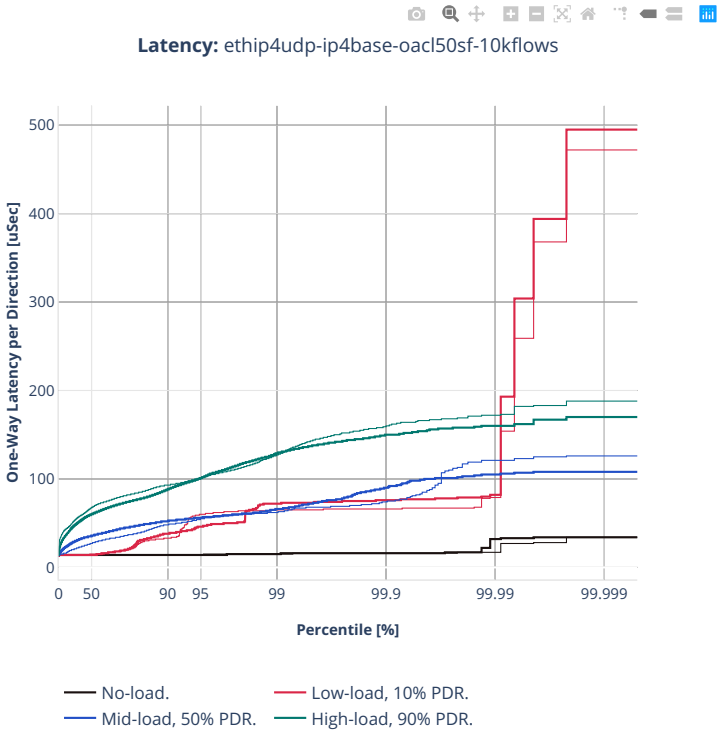

Latency: ethip4-ip4scale200k
One-Way Latency per Direction [uSec]
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

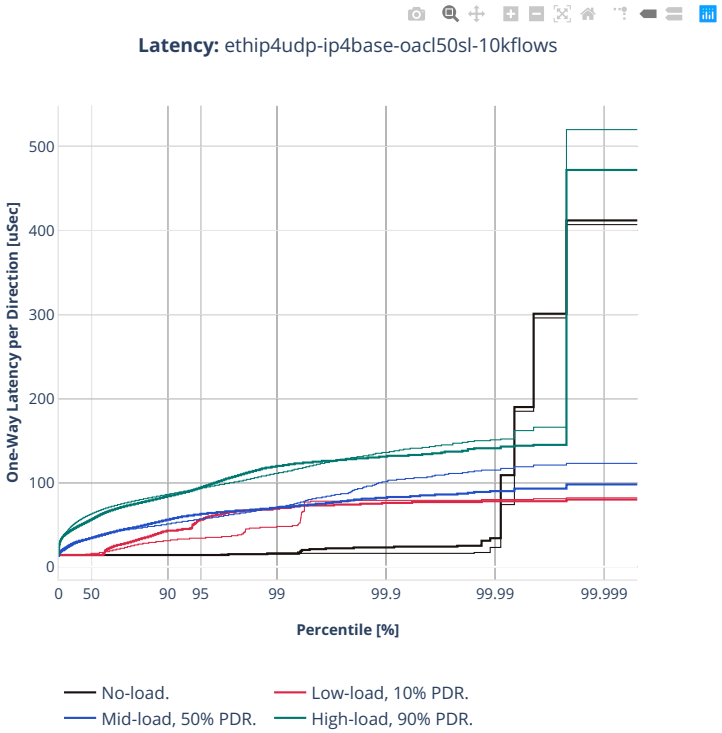
64b-1t1c-ip4routing-features




Latency: ethip4udp-ip4base-iacl50sf-10kflows
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

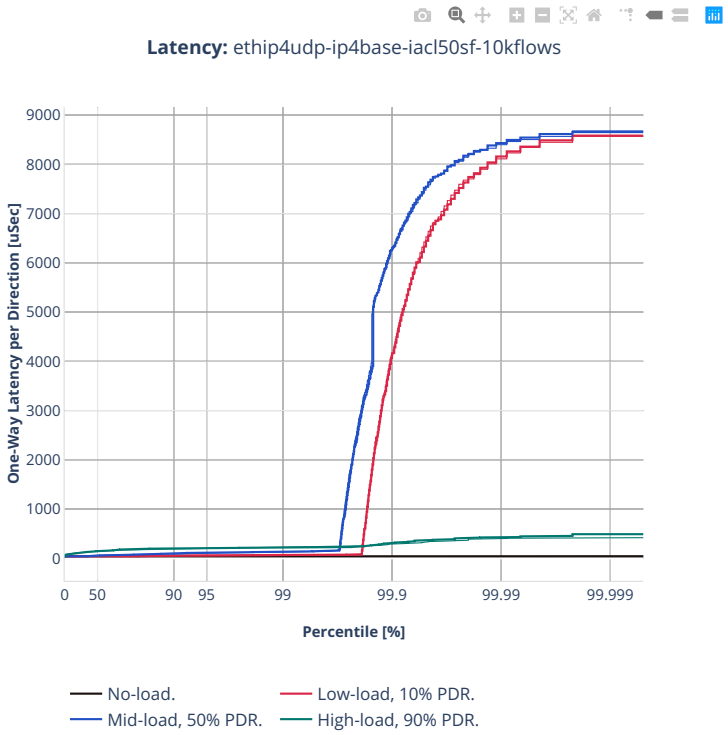

Latency: ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

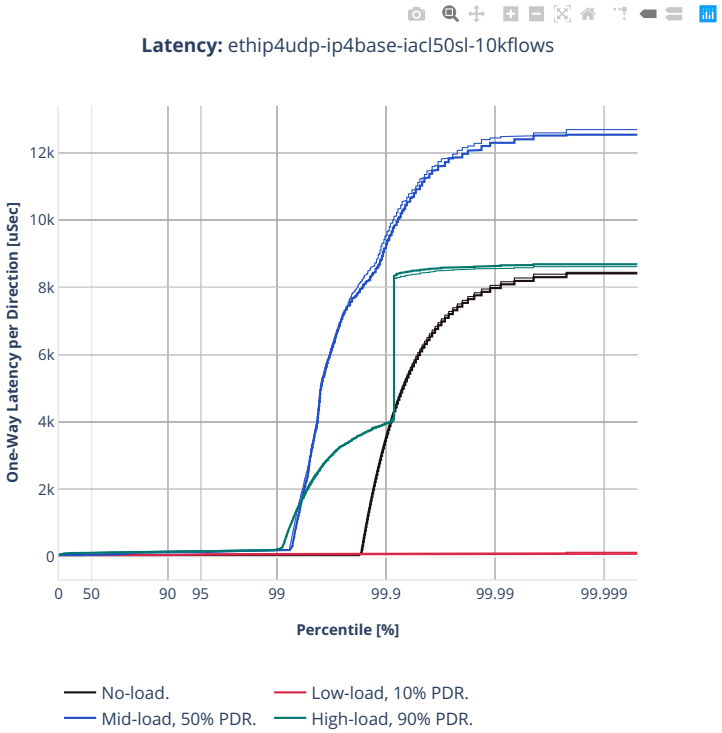

Latency: ethip4udp-ip4base-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

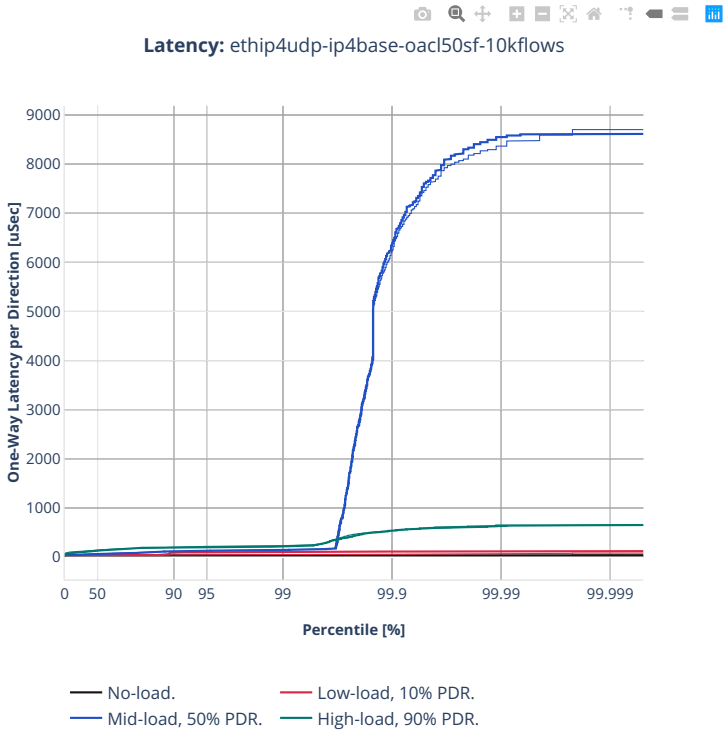

Latency: ethip4udp-ip4base-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3n-tsh-x520

64b-1t1c-ip4routing-features-ixgbe

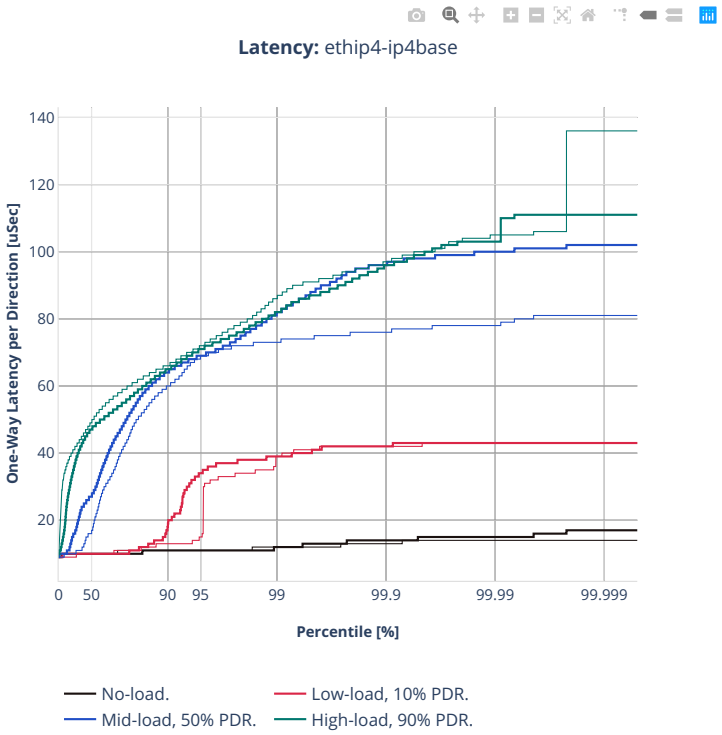



Latency: ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
12k
10k
8k
6k
4k
2k
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

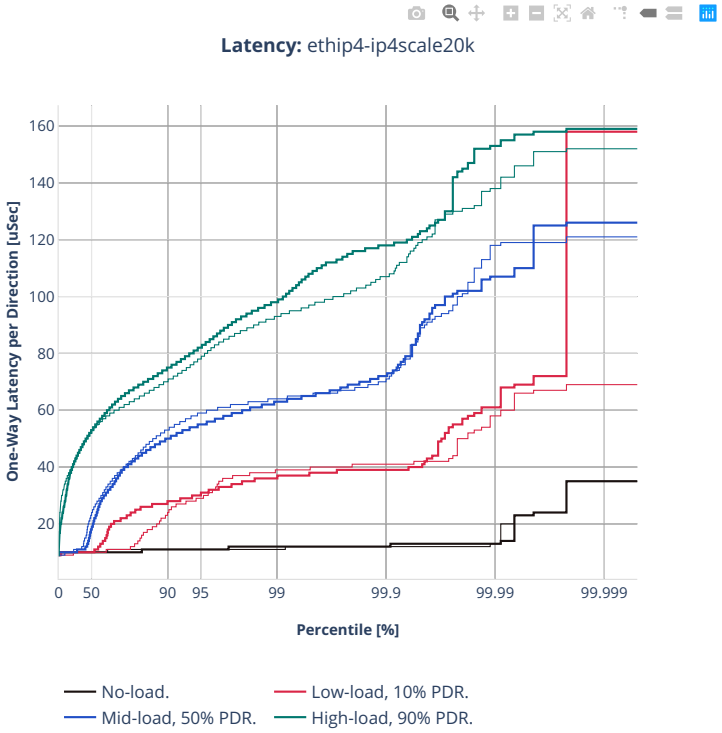

Latency: ethip4udp-ip4base-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

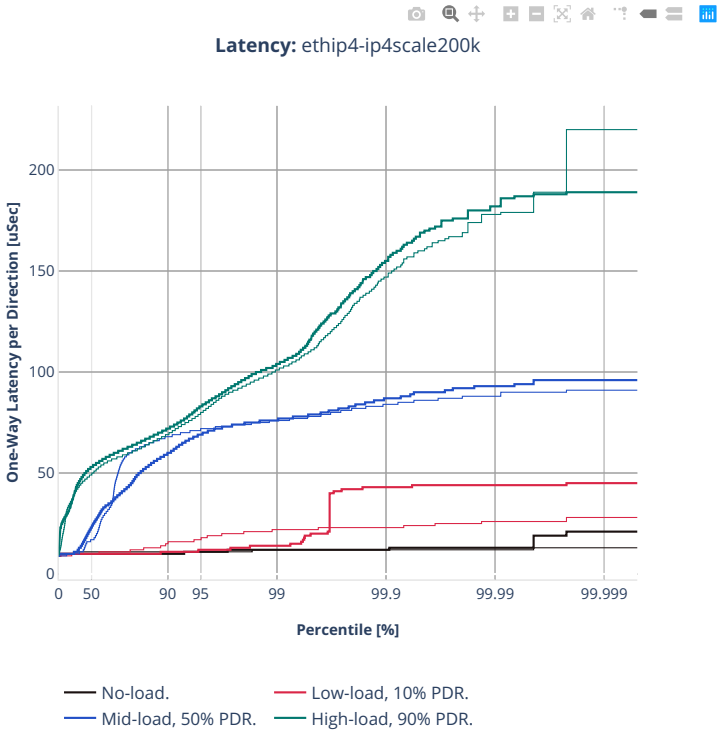
2n-tx2-xl710

64b-1t1c-ip4routing-base-dpdk

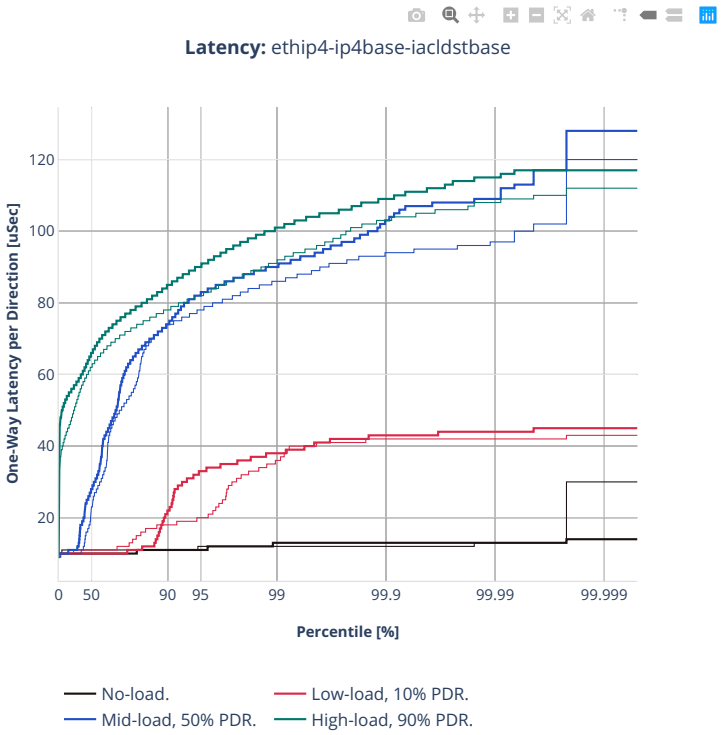


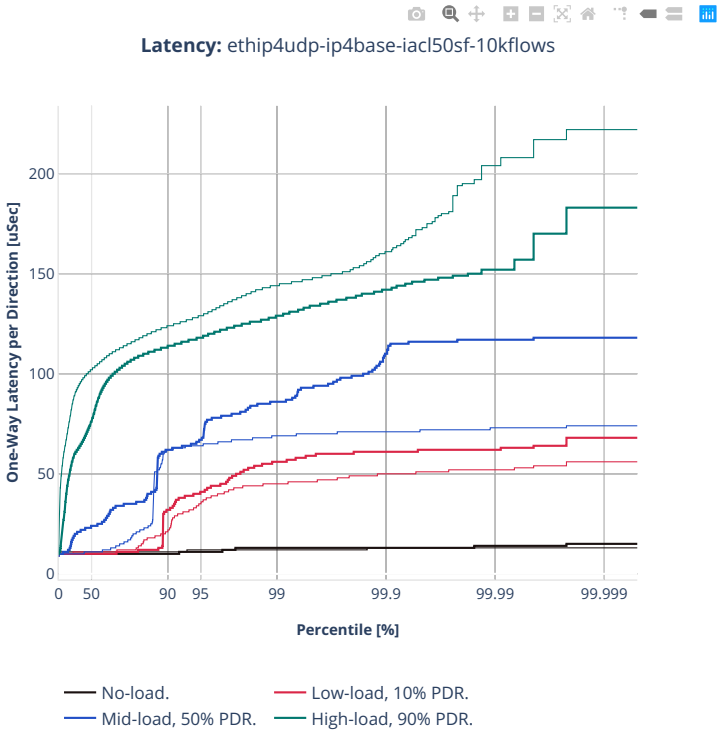
64b-1t1c-ip4routing-scale-dpdk

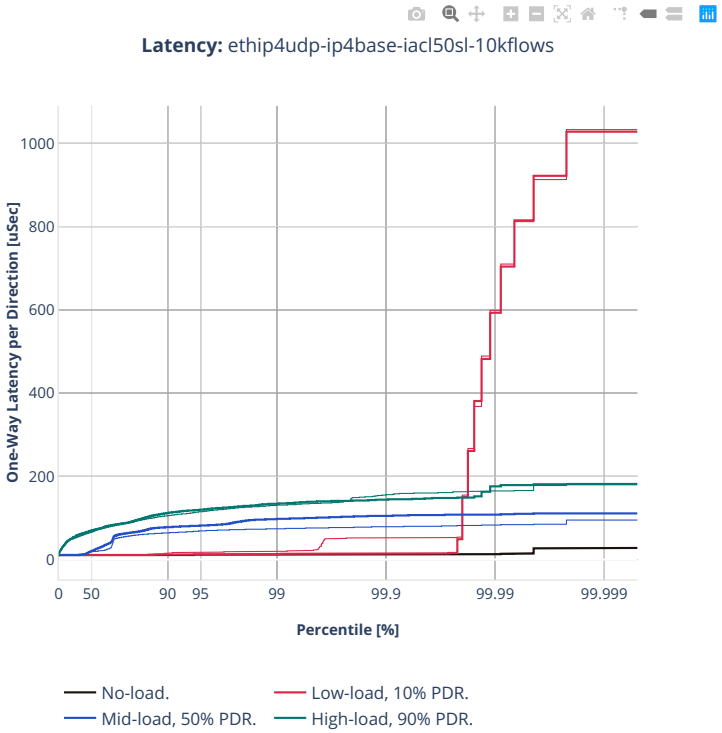


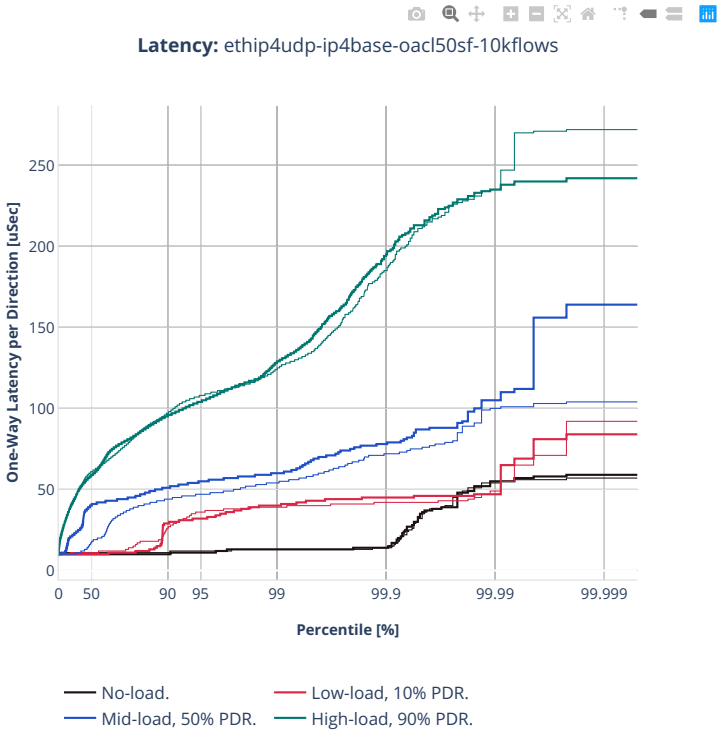

Latency: ethip4-ip4scale200k
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

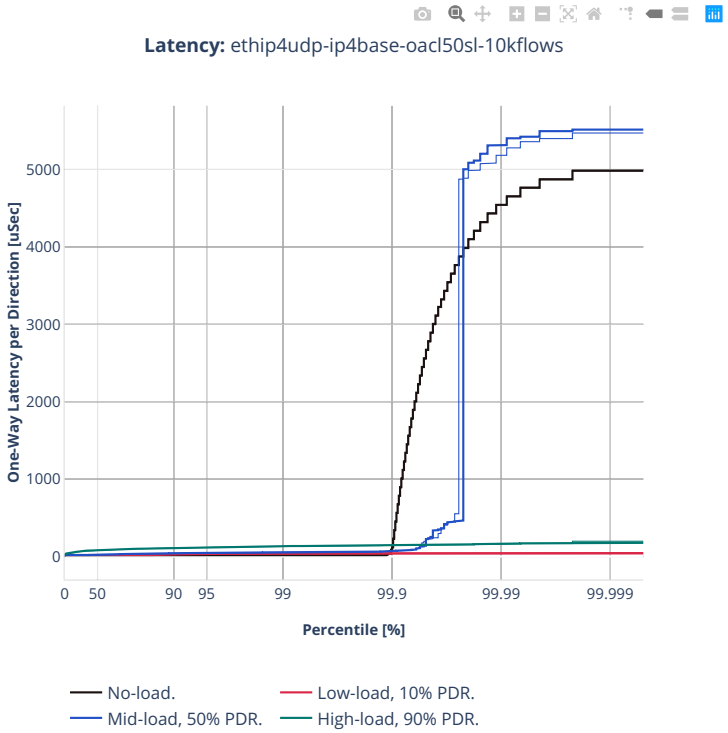
64b-1t1c-features-ip4routing-base-dpdk




Latency: ethip4udp-ip4base-iacl50sf-10kflows
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4udp-ip4base-iacl50sl-10kflows
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4udp-ip4base-oacl50sf-10kflows
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4udp-ip4base-oacl50sl-10kflows
One-Way Latency per Direction [uSec]
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

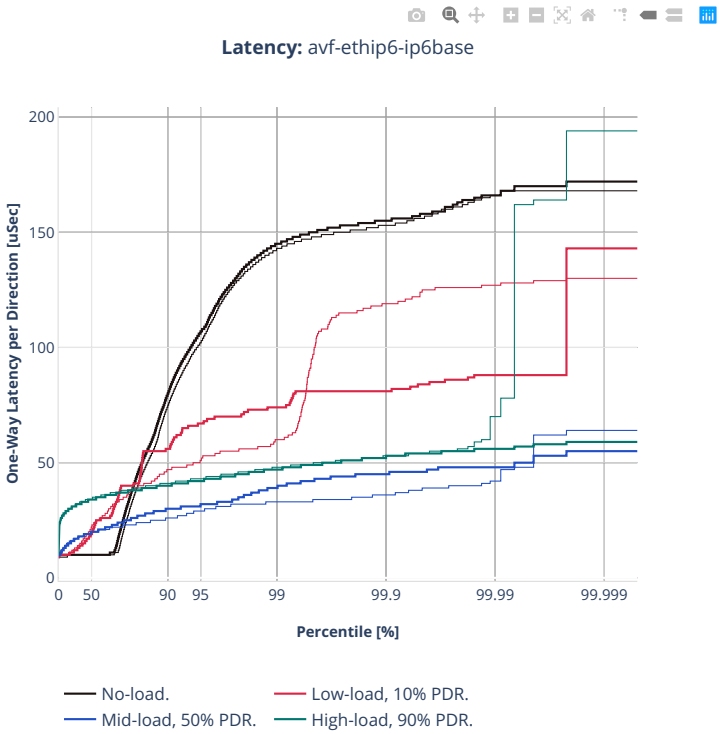
2.5.3 IPv6 Routing

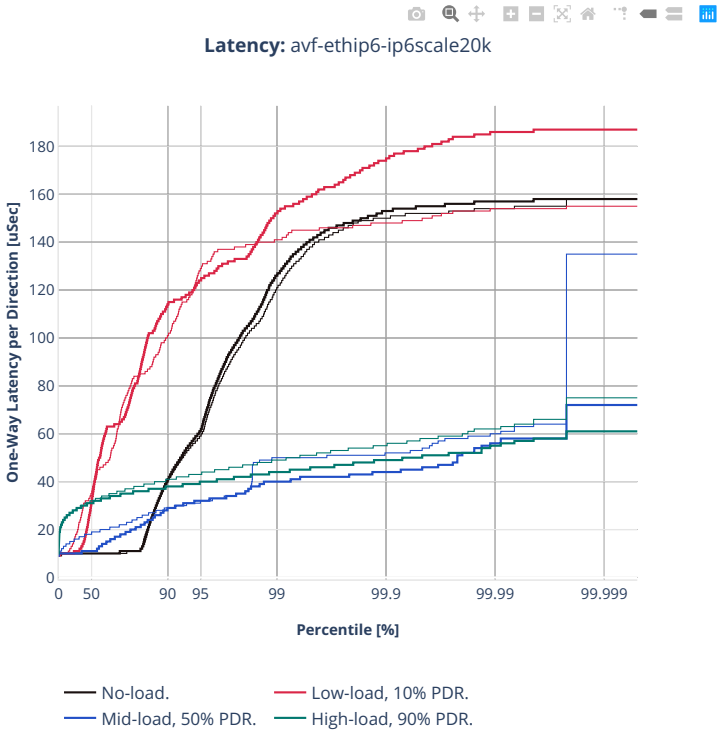
CSIT source code for the test cases used for plots can be found in CSIT git repository[152].

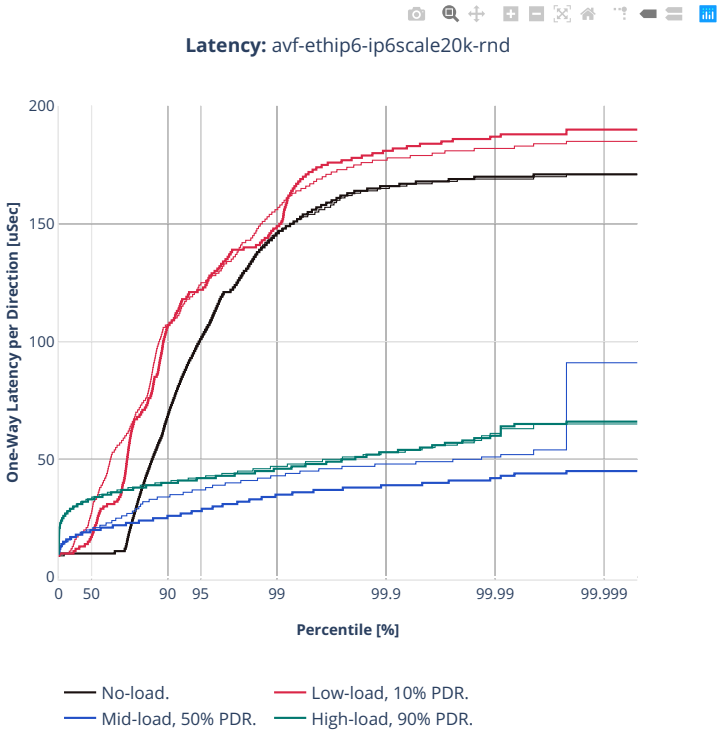
[152] https://git.fd.io/csit/tree/tests/vpp/perf/ip6?h=rls2210

2n-icx-xxv710

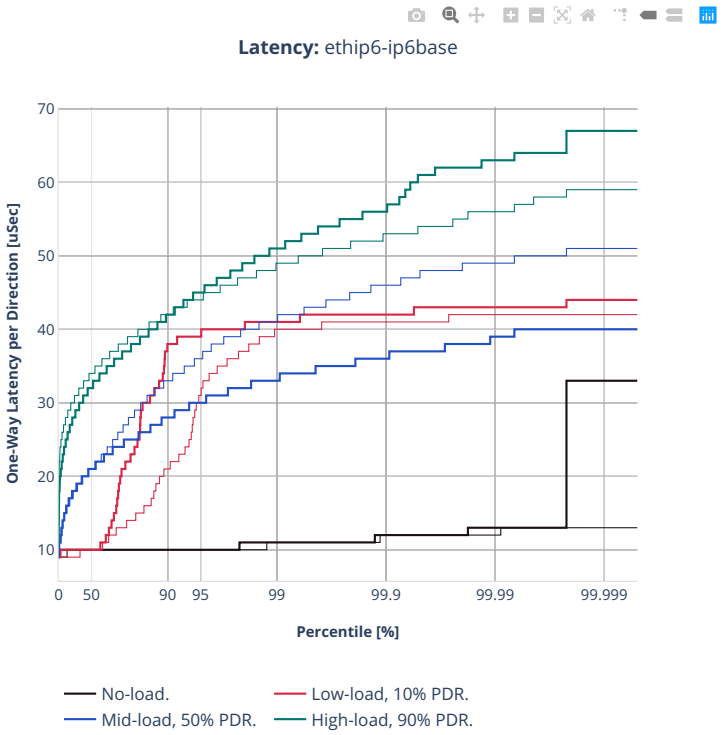
78b-2t1c-ip6routing-base-scale-avf

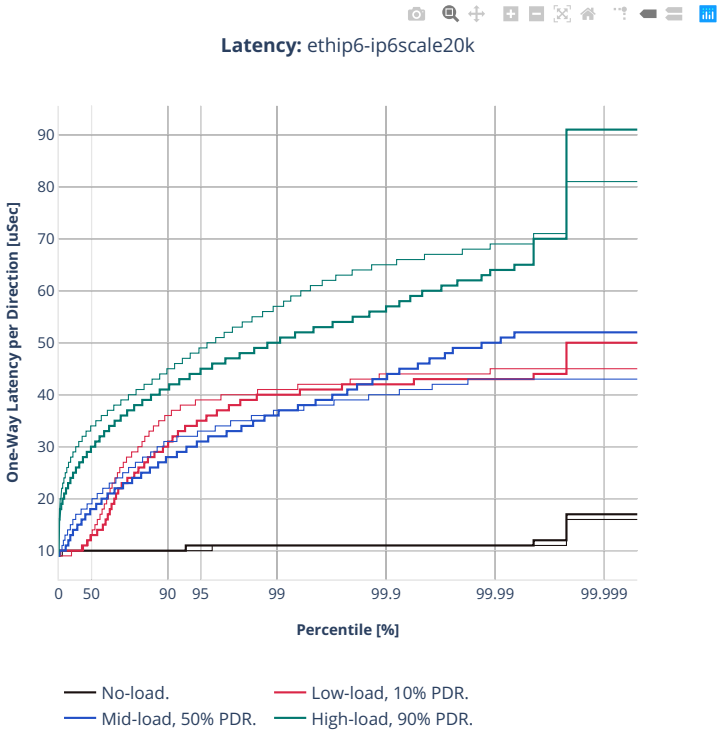


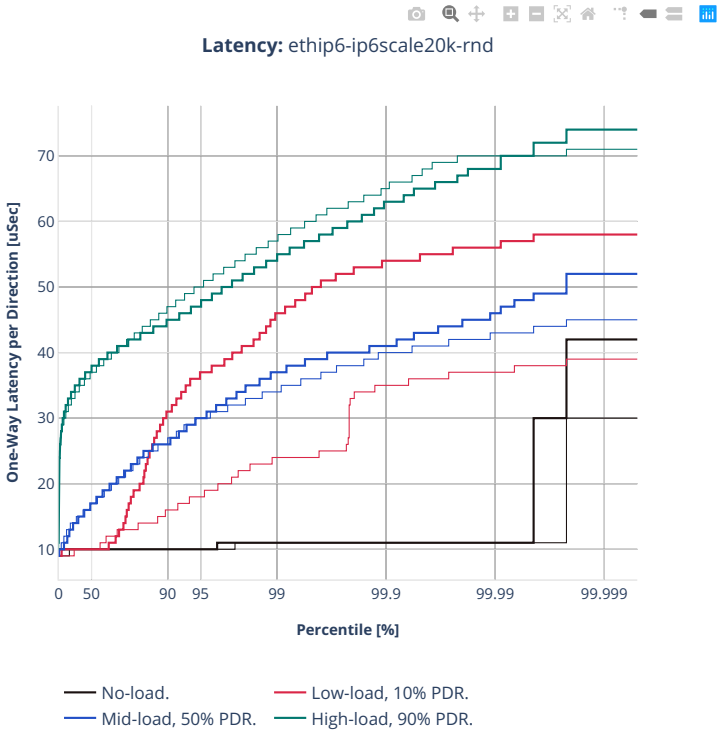

Latency: avf-ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

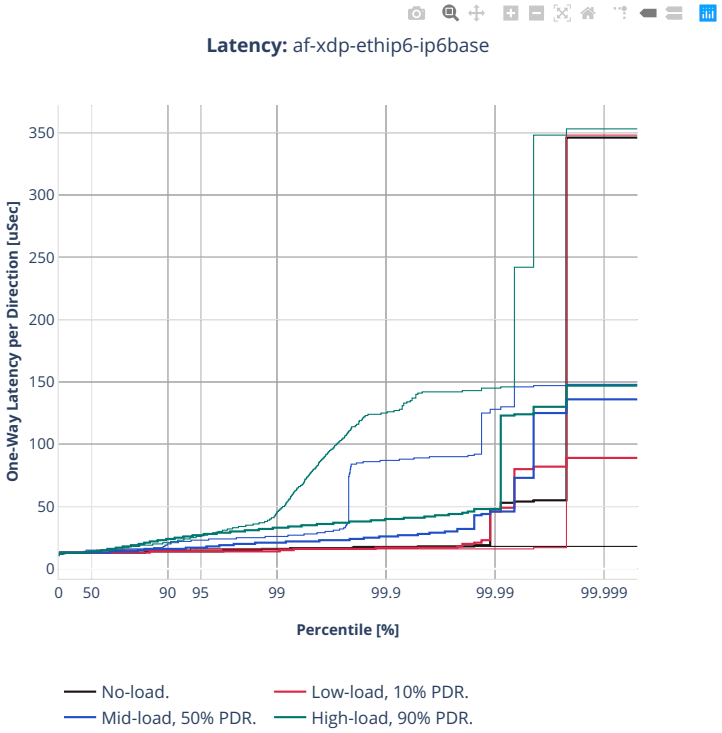
78b-2t1c-ip6routing-base-scale-dpdk

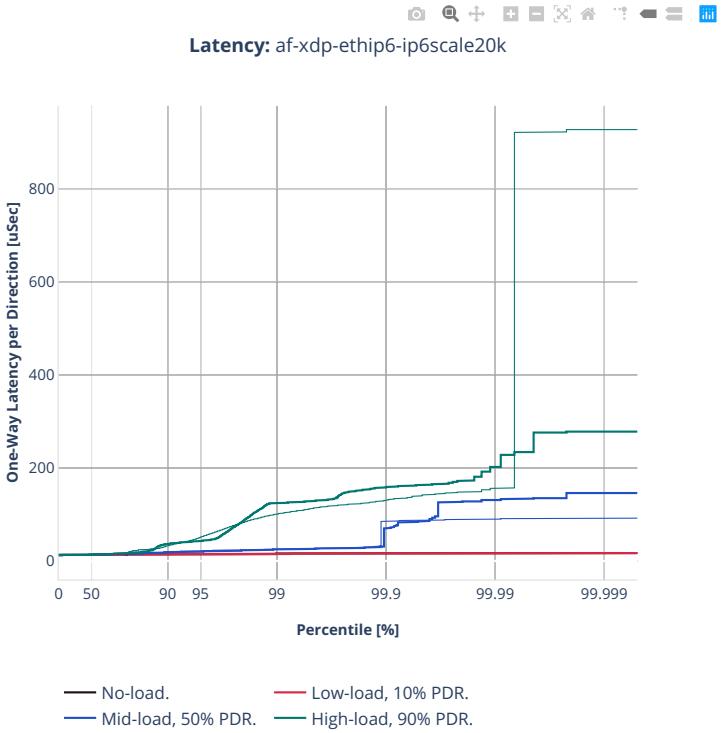


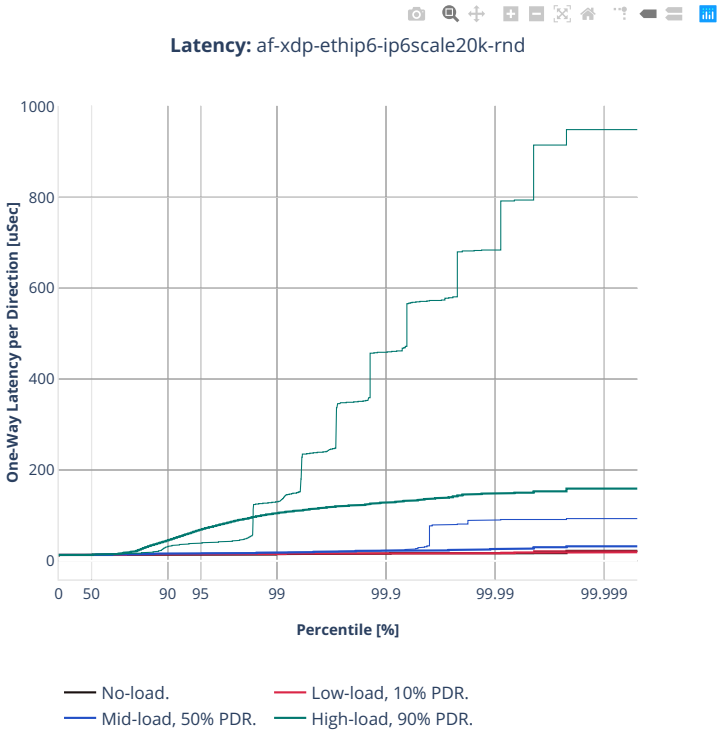

Latency: ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
90
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

78b-2t1c-ip6routing-base-scale-af-xdp

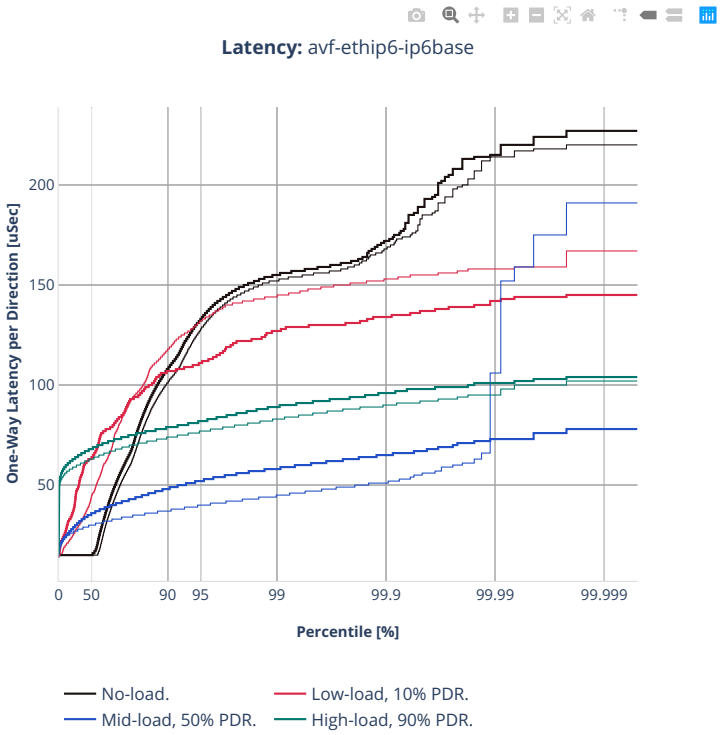



Latency: af-xdp-ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

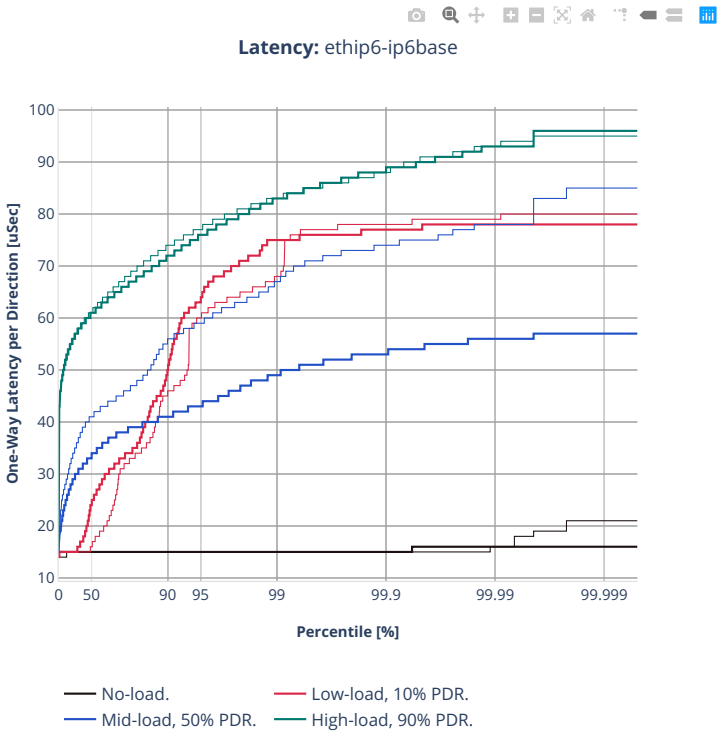

Latency: af-xdp-ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3n-icx-xxv710

78b-2t1c-ip6routing-base-avf

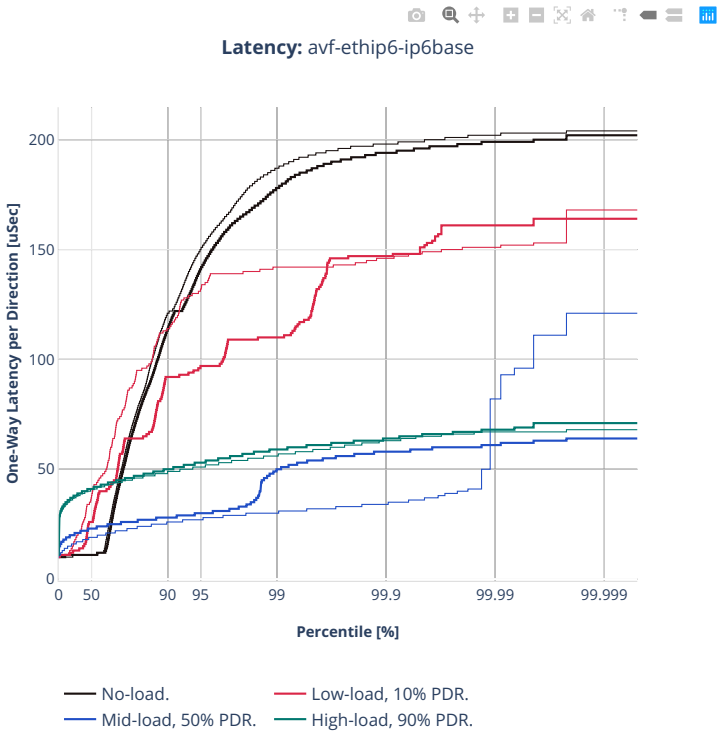


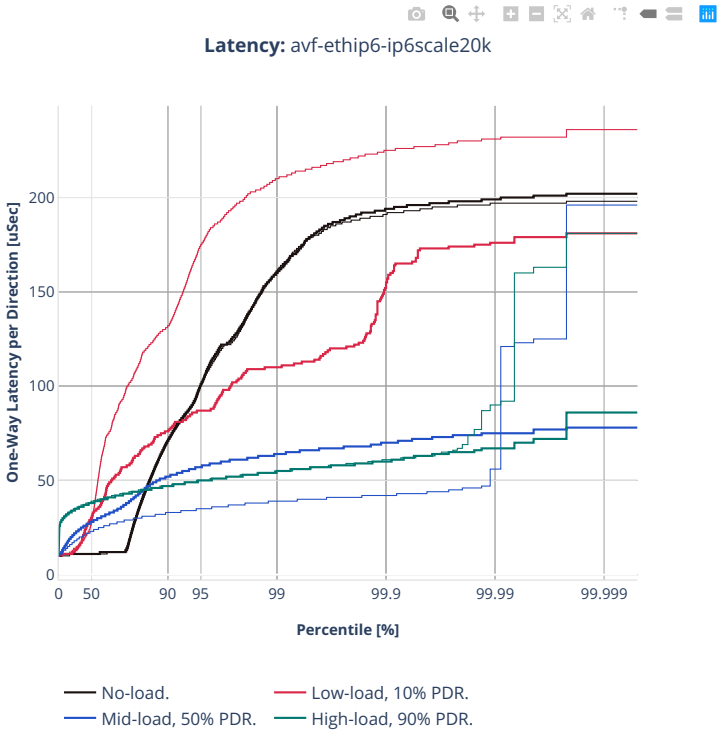
78b-2t1c-ip6routing-base-dpdk

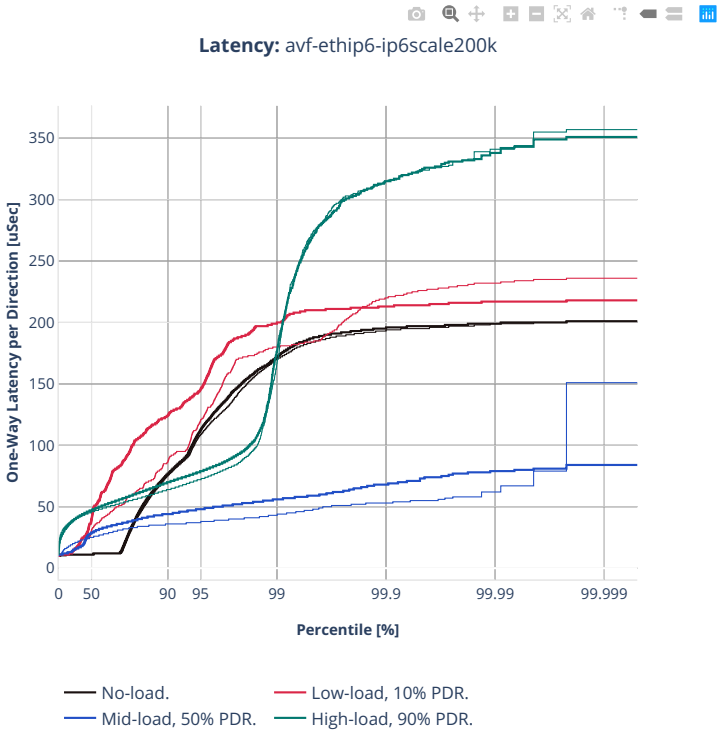


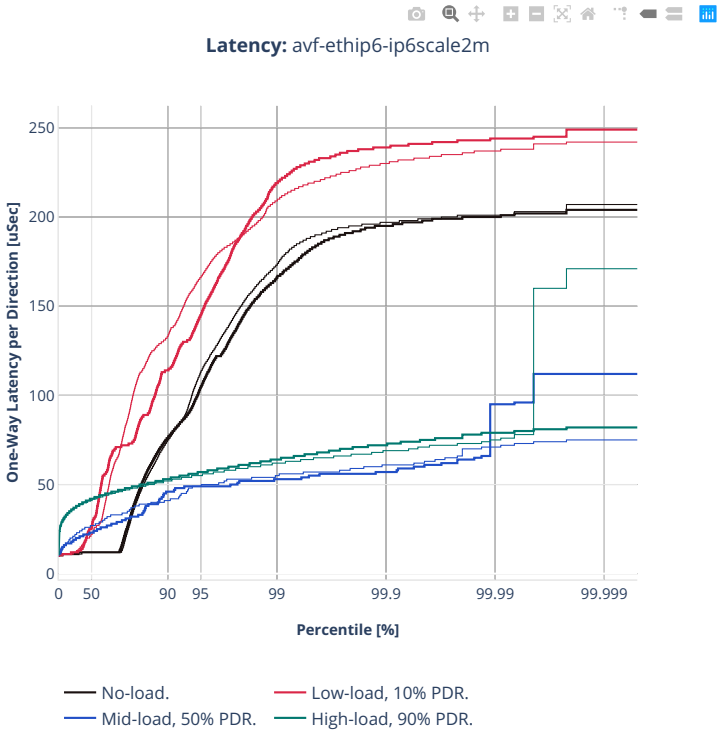
2n-clx-xxv710

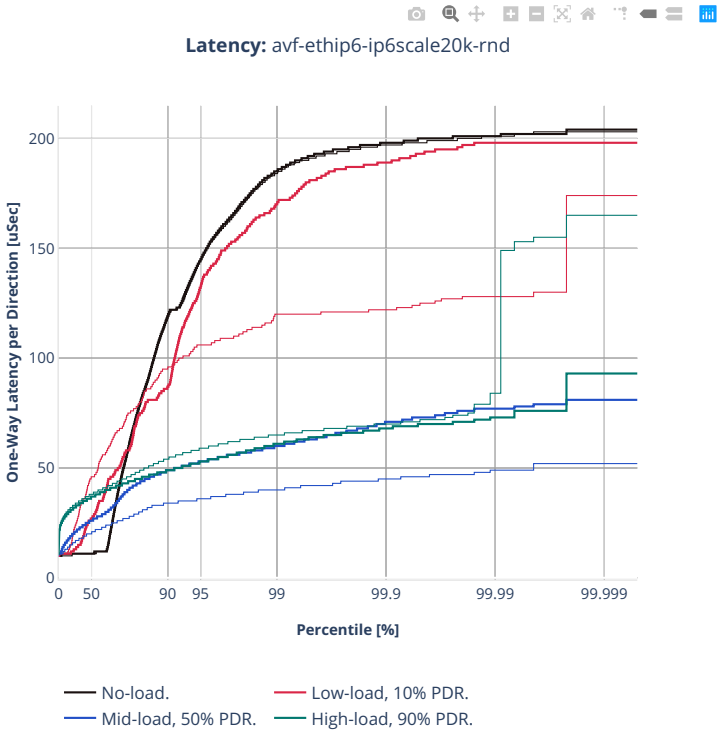
78b-2t1c-ip6routing-base-scale-avf

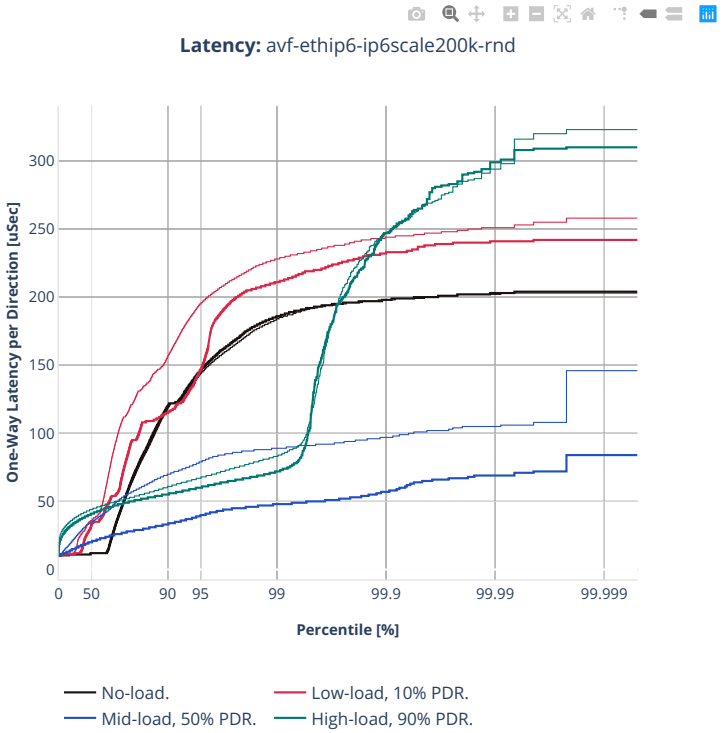



Latency: avf-ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
0
50
100
150
200
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

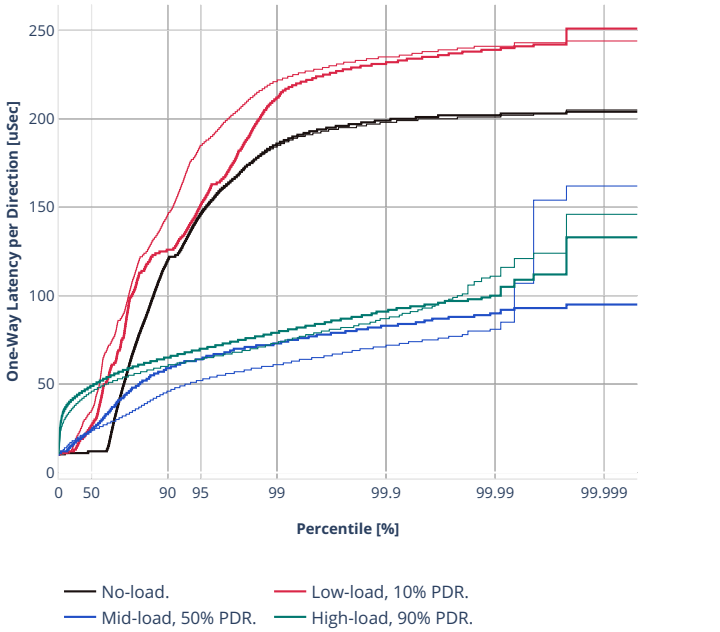

Latency: avf-ethip6-ip6scale200k
One-Way Latency per Direction [uSec]
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip6-ip6scale2m
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

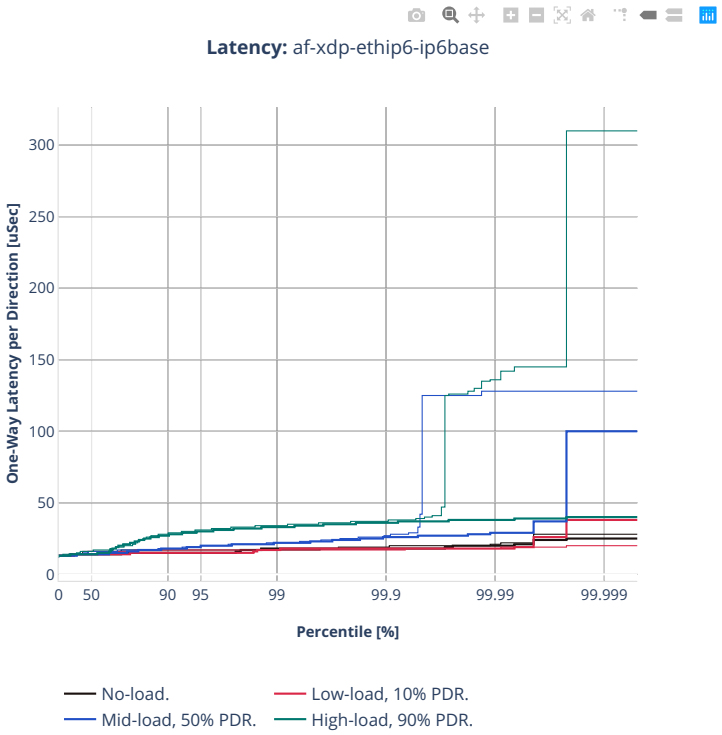

Latency: avf-ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

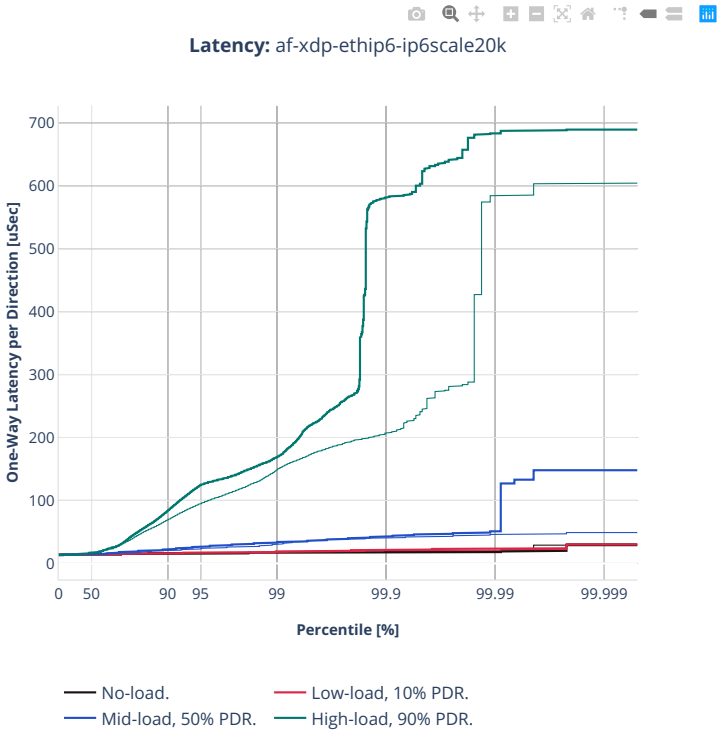

Latency: avf-ethip6-ip6scale200k-rnd
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

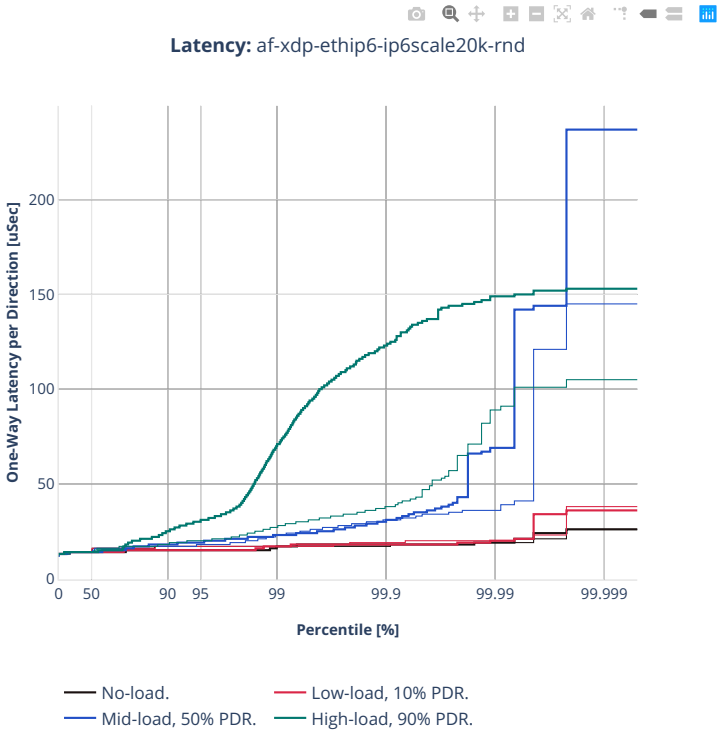
Latency: avf-ethip6-ip6scale2m-rnd



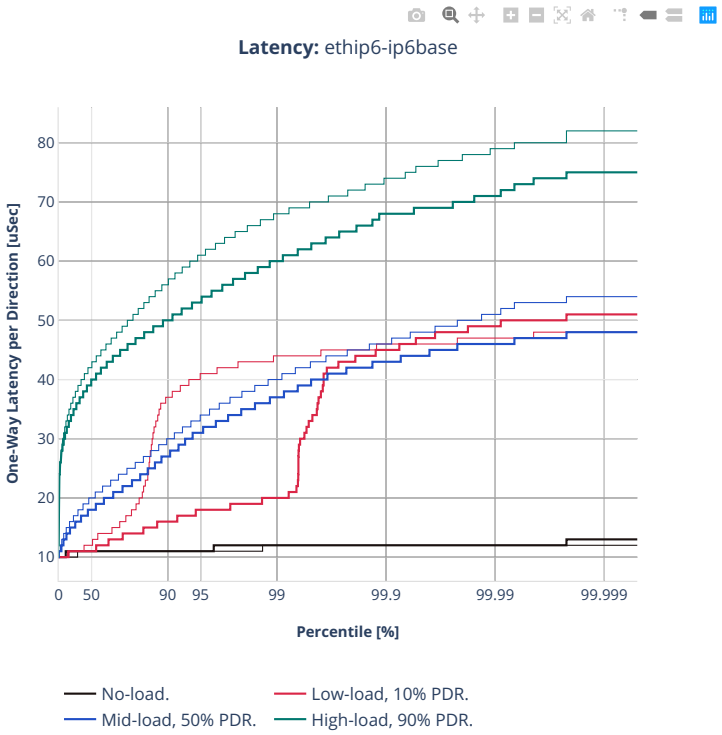
78b-2t1c-ip6routing-base-scale-af-xdp

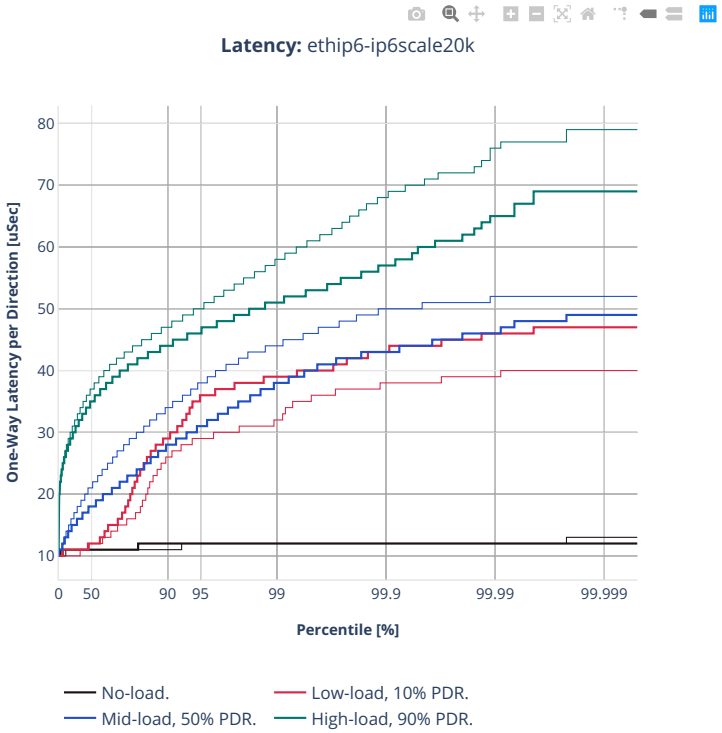


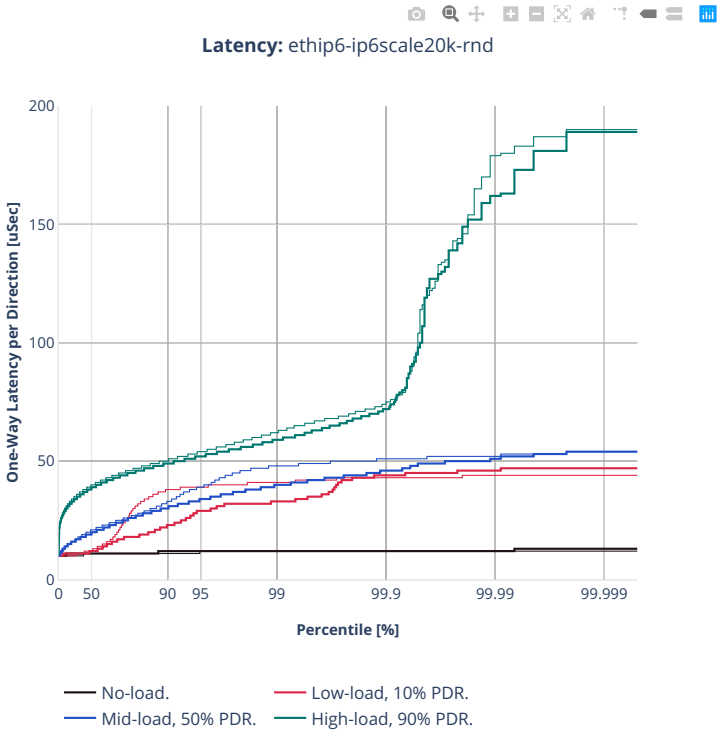

Latency: af-xdp-ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
700
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: af-xdp-ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
0
50
100
150
200
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

78b-2t1c-ip6routing-base-scale-dpdk

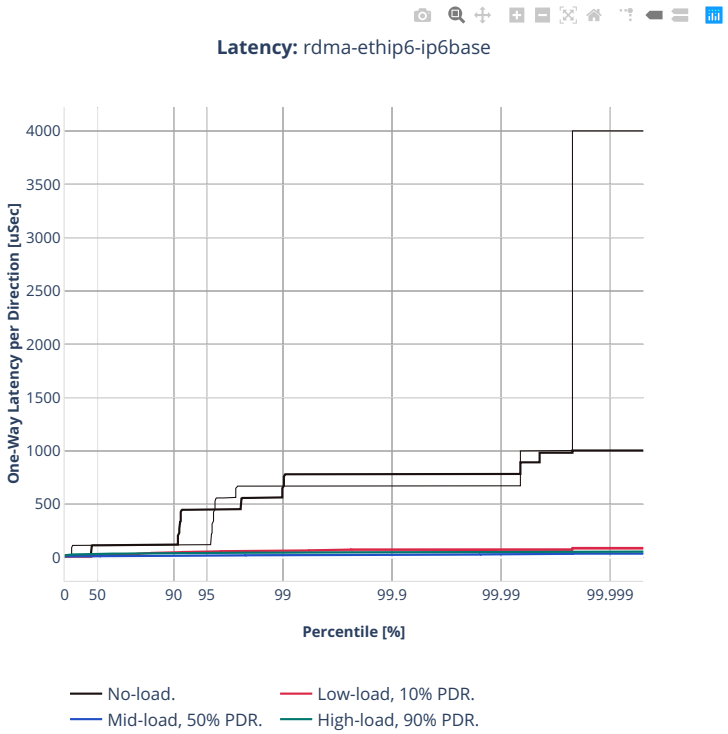


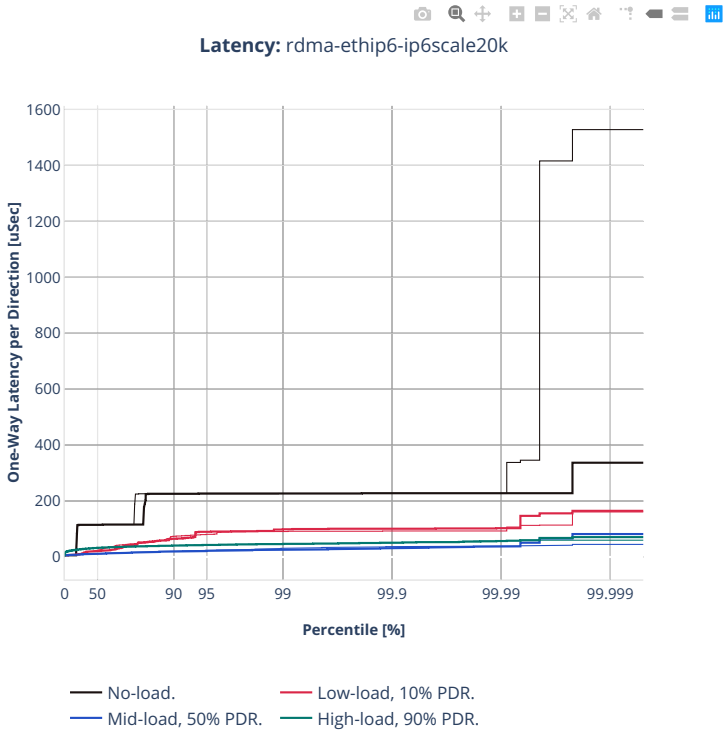

Latency: ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

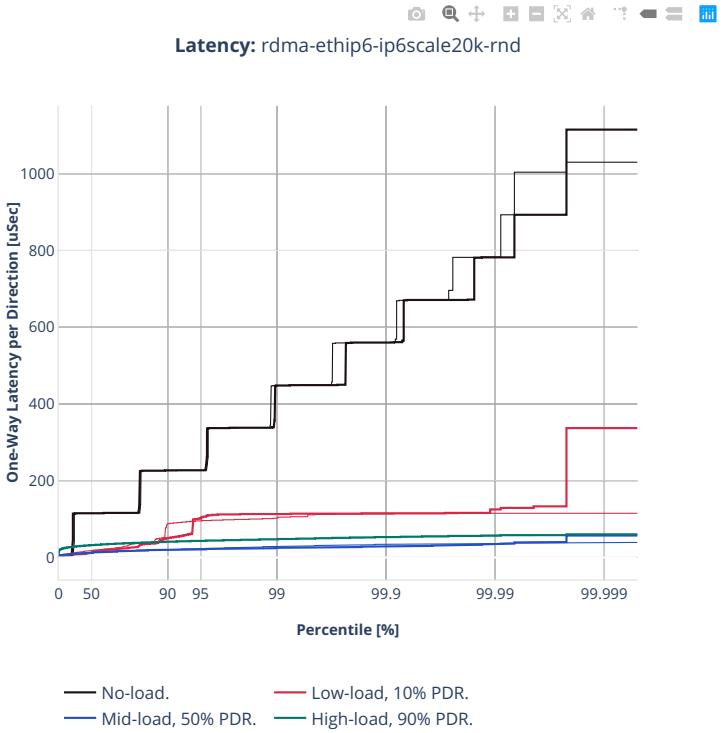

Latency: ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-clx-cx556a

78b-2t1c-ip6routing-base-scale-rdma

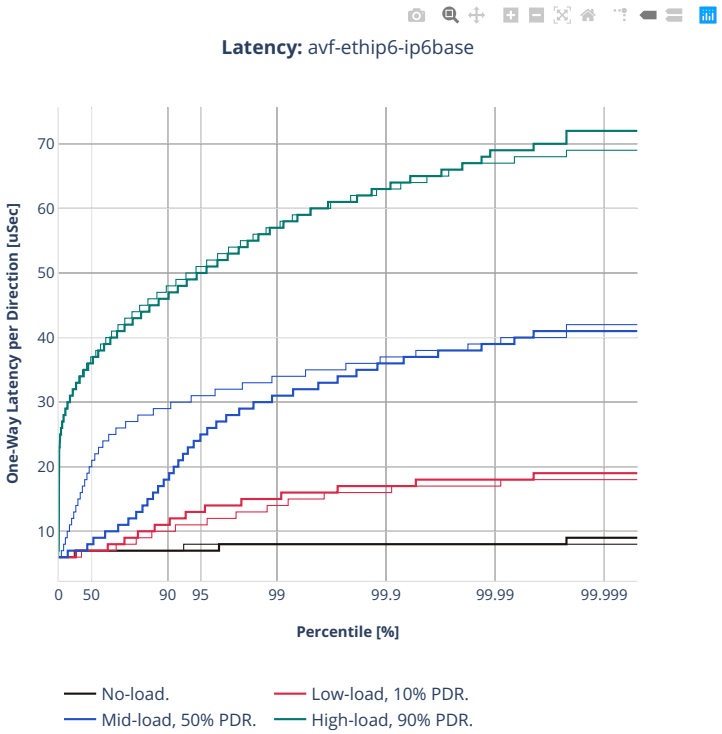


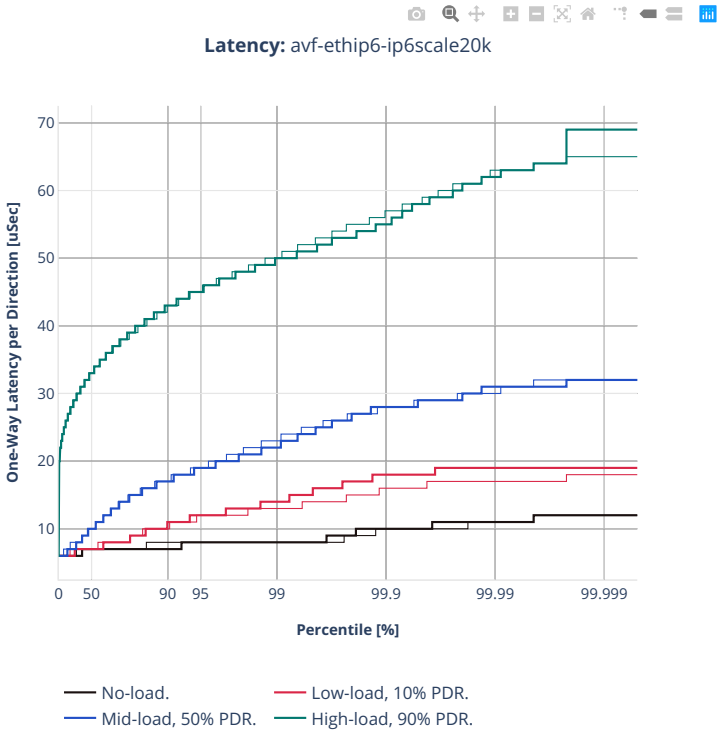

Latency: rdma-ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
1600
1400
1200
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

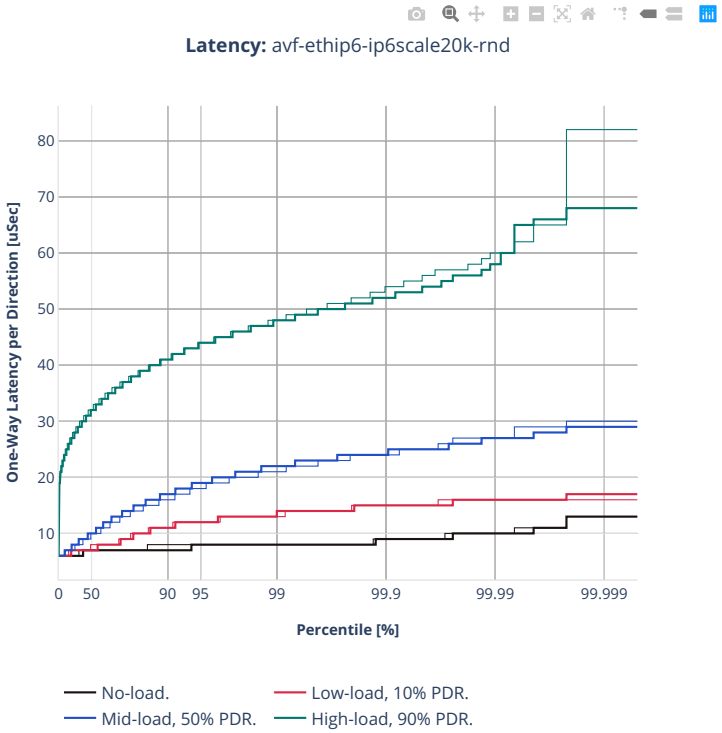

Latency: rdma-ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-clx-e810cq

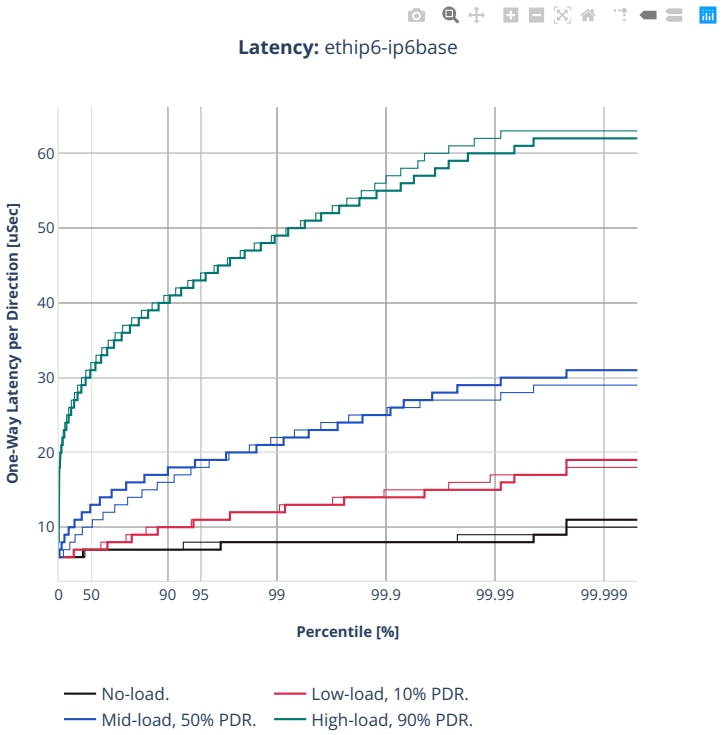
78b-2t1c-ip6routing-base-scale-avf

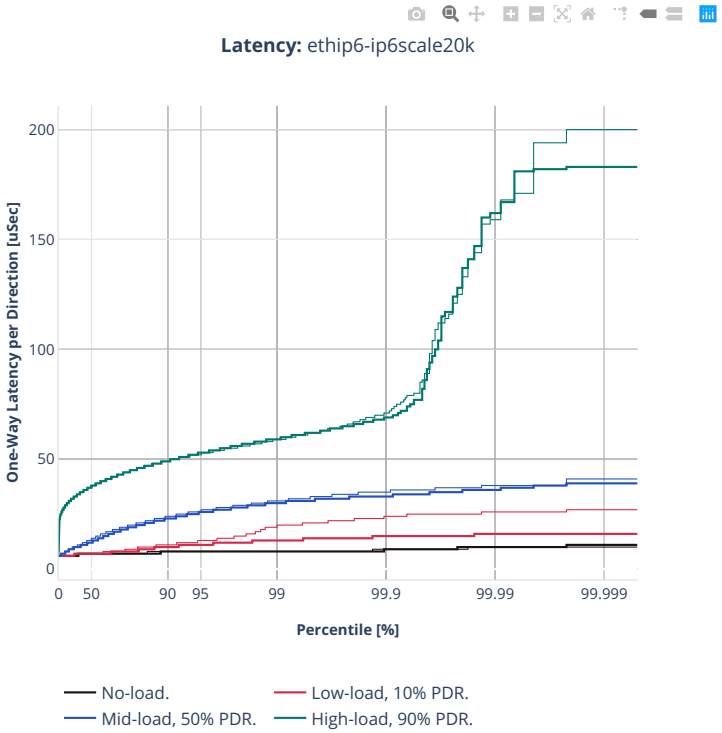


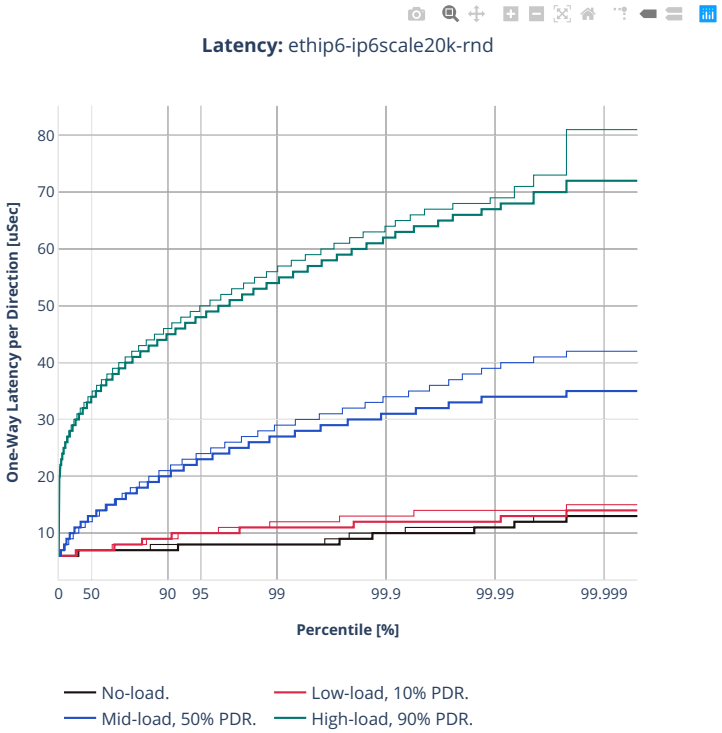

Latency: avf-ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

78b-2t1c-ip6routing-base-scale-dpdk

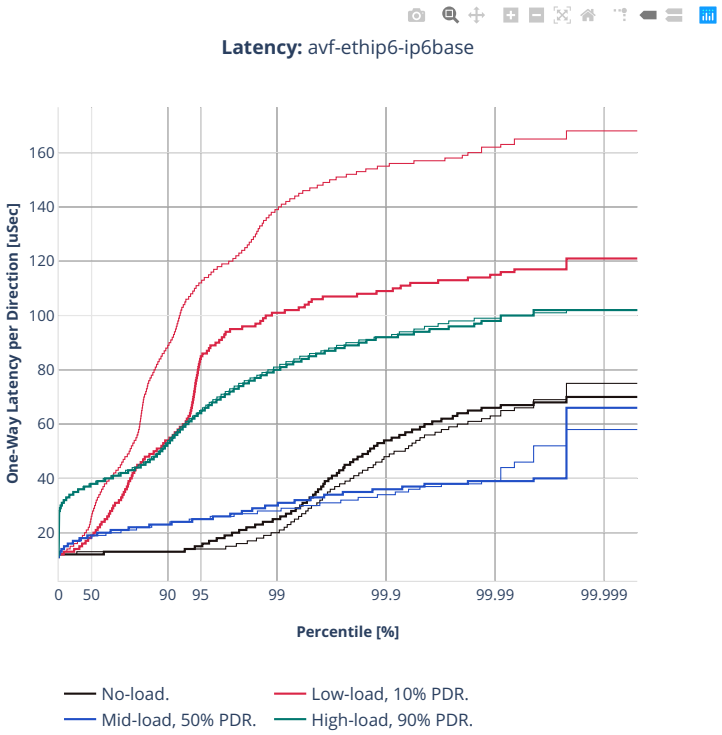


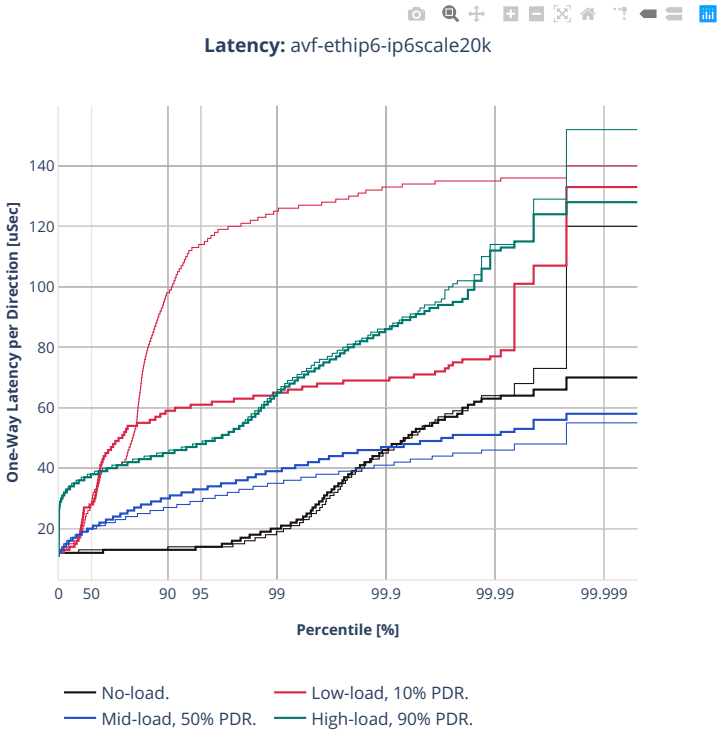

Latency: ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
80
70
60
50
40
30
20
10
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

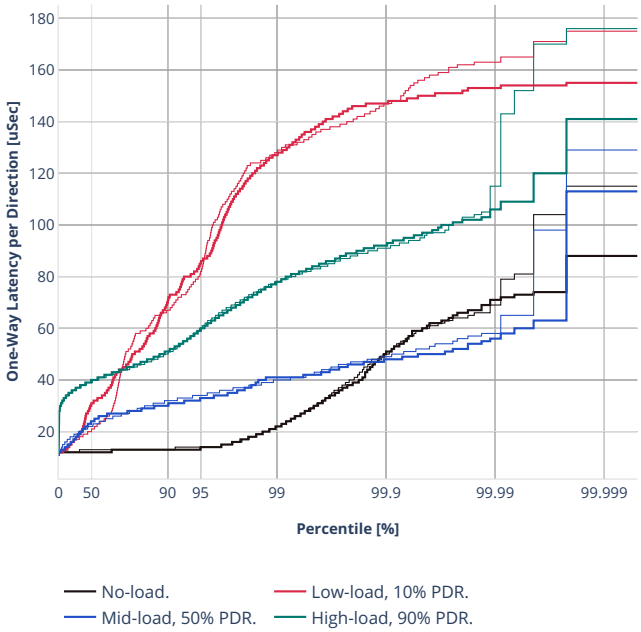
2n-zn2-xxv710

78b-2t1c-ip6routing-base-scale-avf

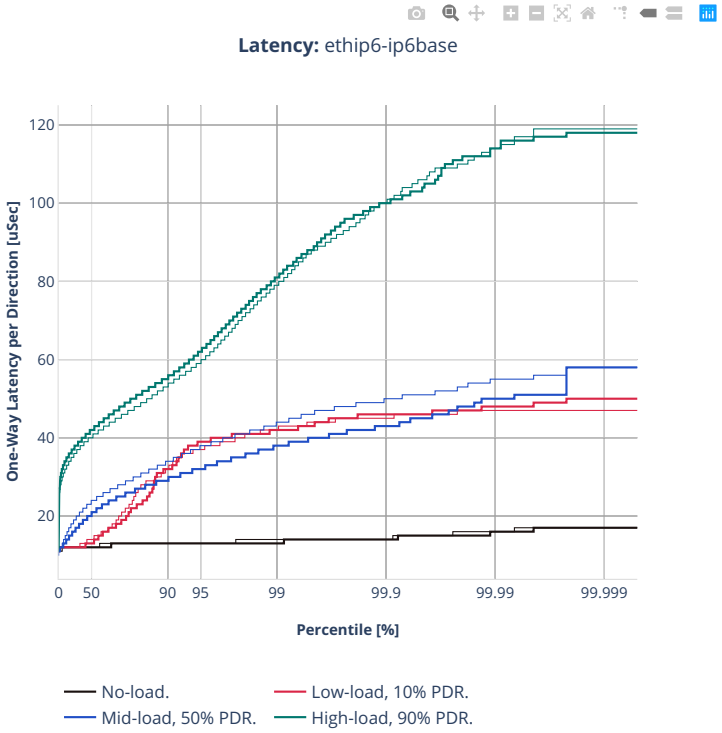


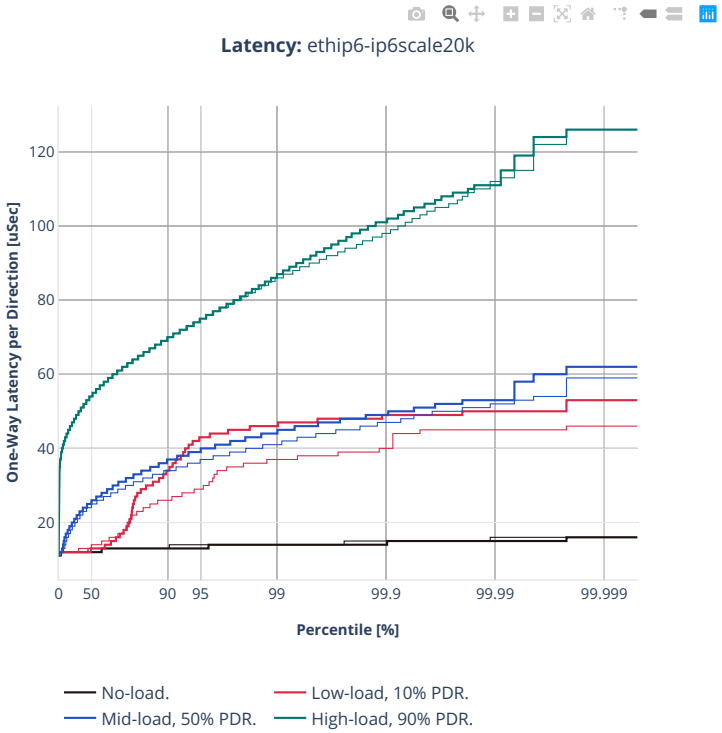

Latency: avf-ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

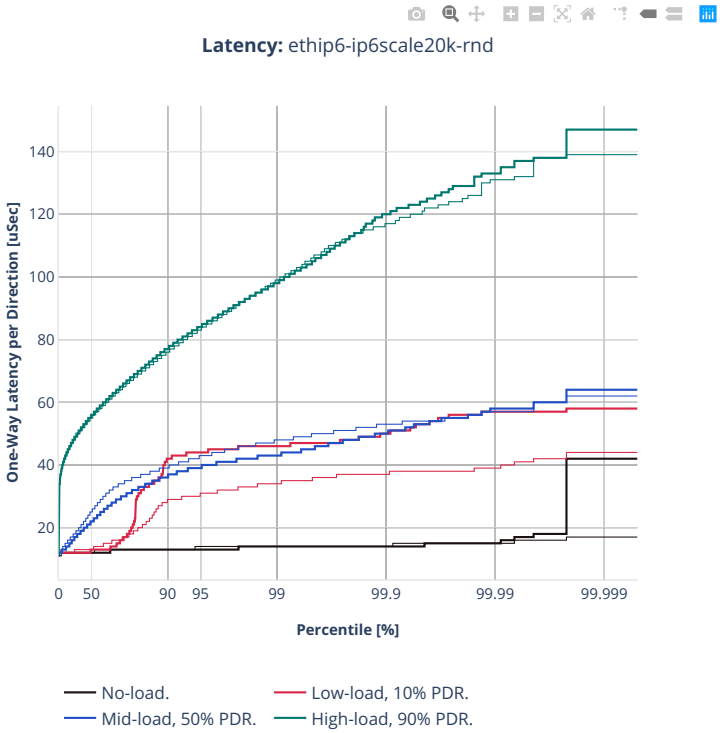
Latency: avf-ethip6-ip6scale20k-rnd



78b-2t1c-ip6routing-base-scale-dpdk

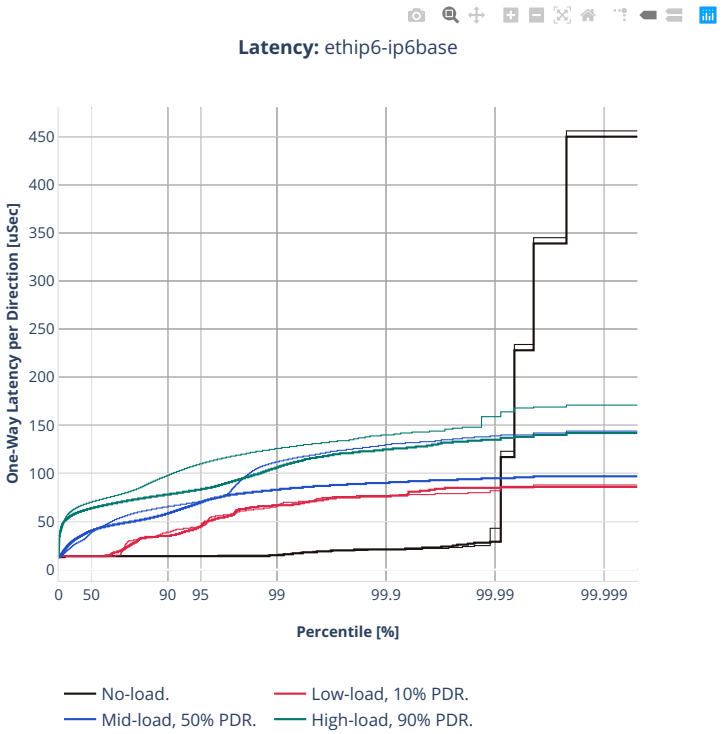


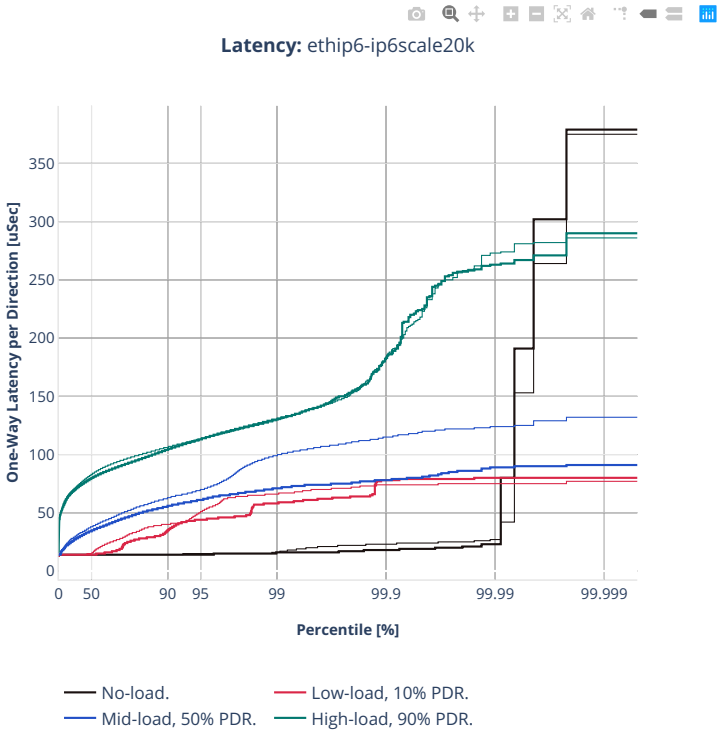

Latency: ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

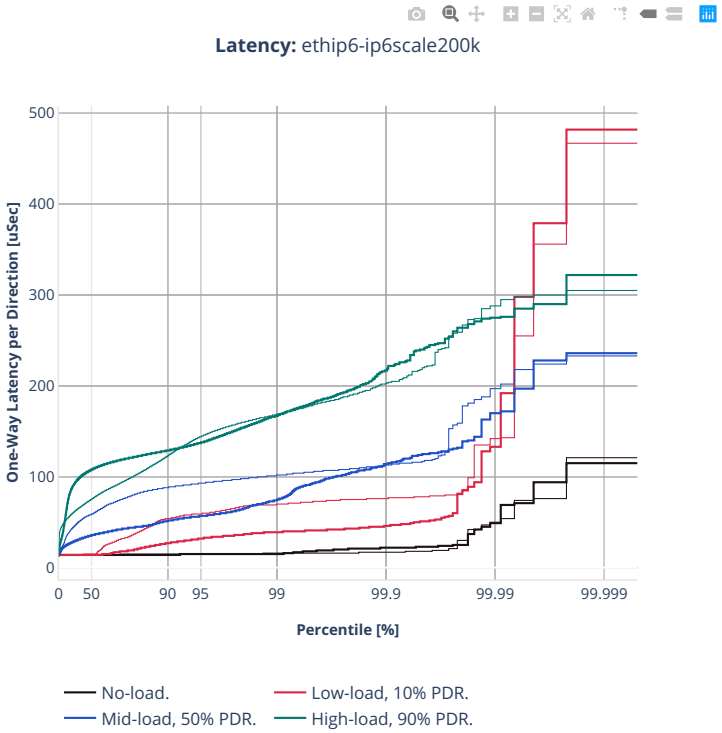

Latency: ethip6-ip6scale20k-rnd
One-Way Latency per Direction [uSec]
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

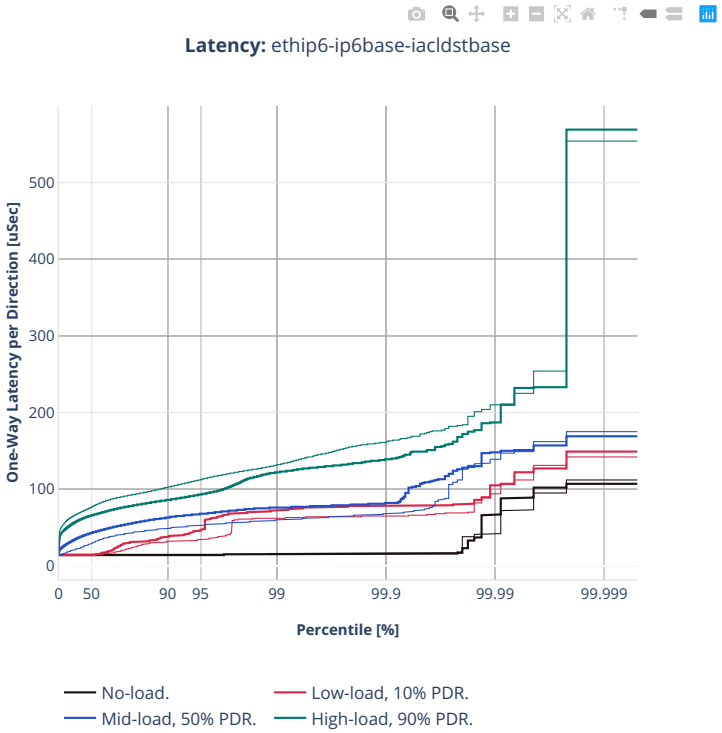
3n-alt-xl710

78b-1t1c-ip6routing-base-scale



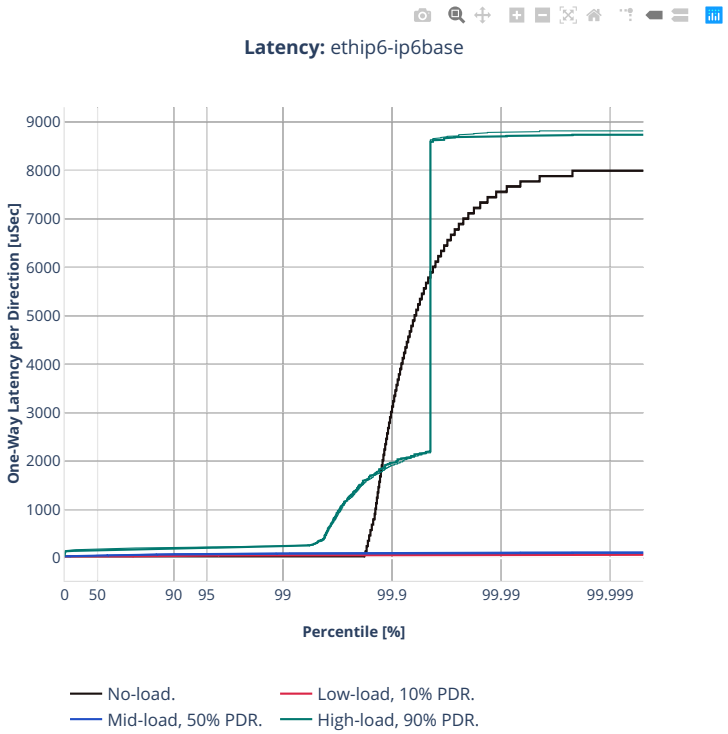

Latency: ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip6-ip6scale200k
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip6-ip6base-iacldstbase
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

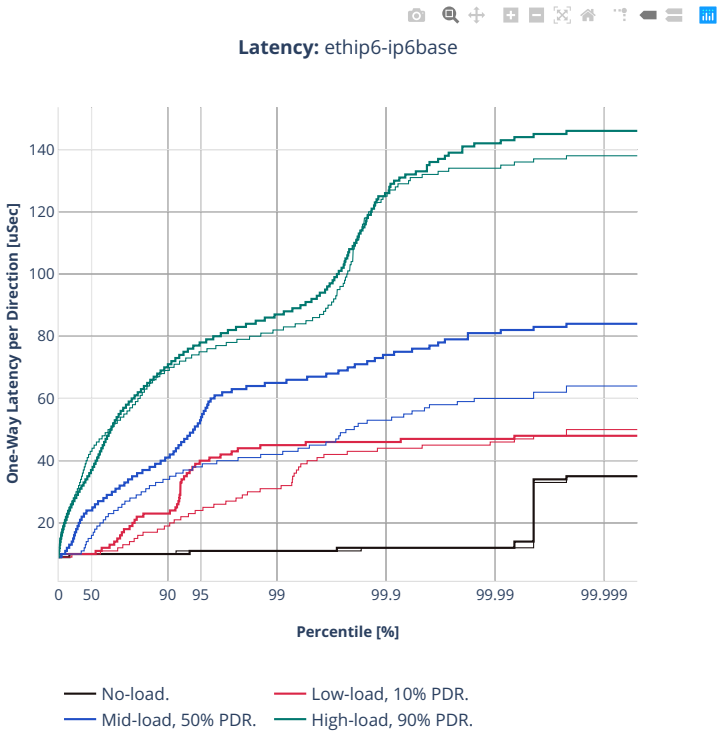
3n-tsh-x520

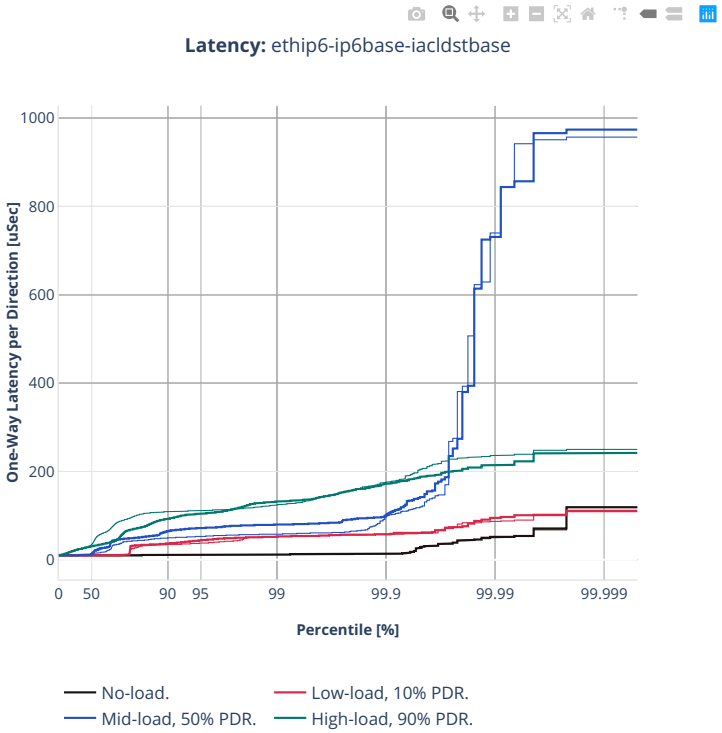
78b-1t1c-ip6routing-base-scale-ixgbe

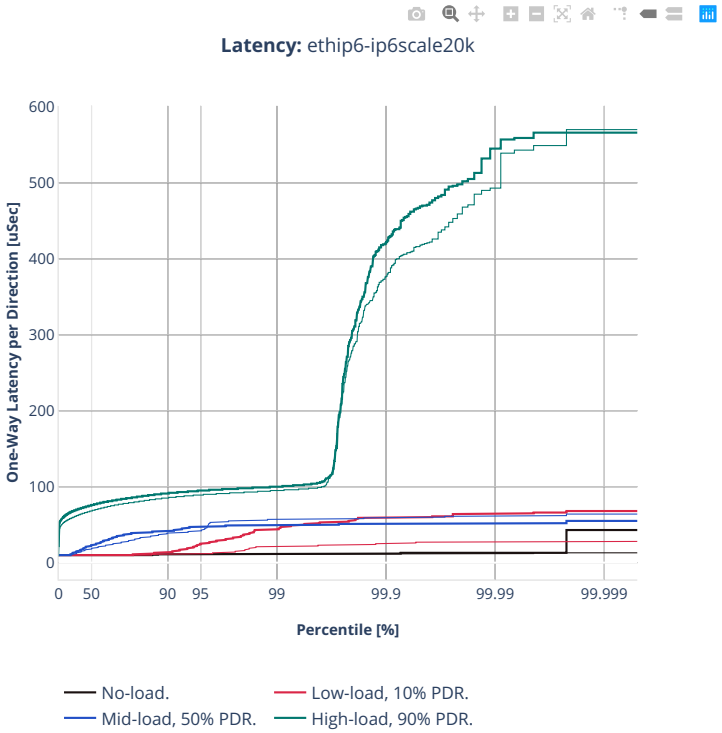


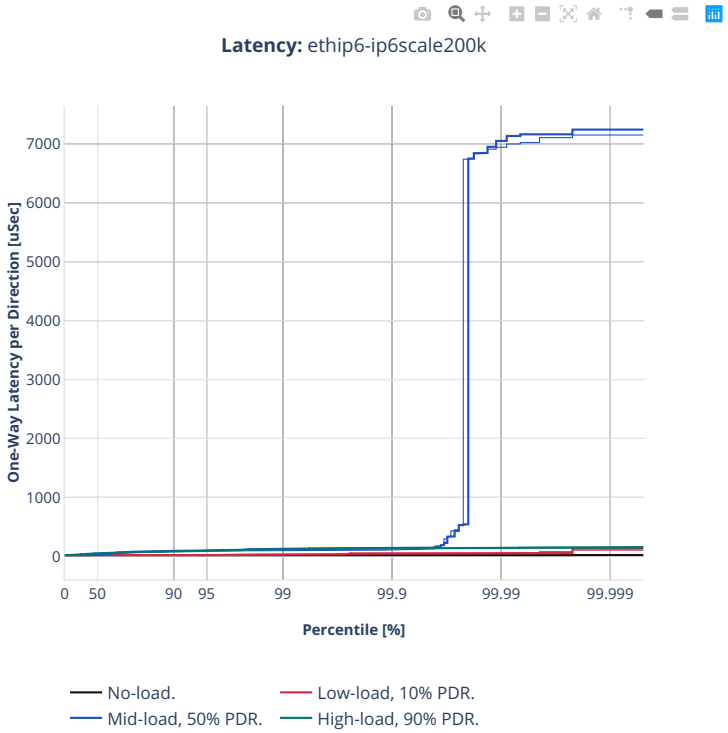
2n-tx2-xl710

78b-1t1c-ip6routing-base-scale-dpdk




Latency: ethip6-ip6base-iacldstbase
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip6-ip6scale20k
One-Way Latency per Direction [uSec]
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip6-ip6scale200k
One-Way Latency per Direction [uSec]
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

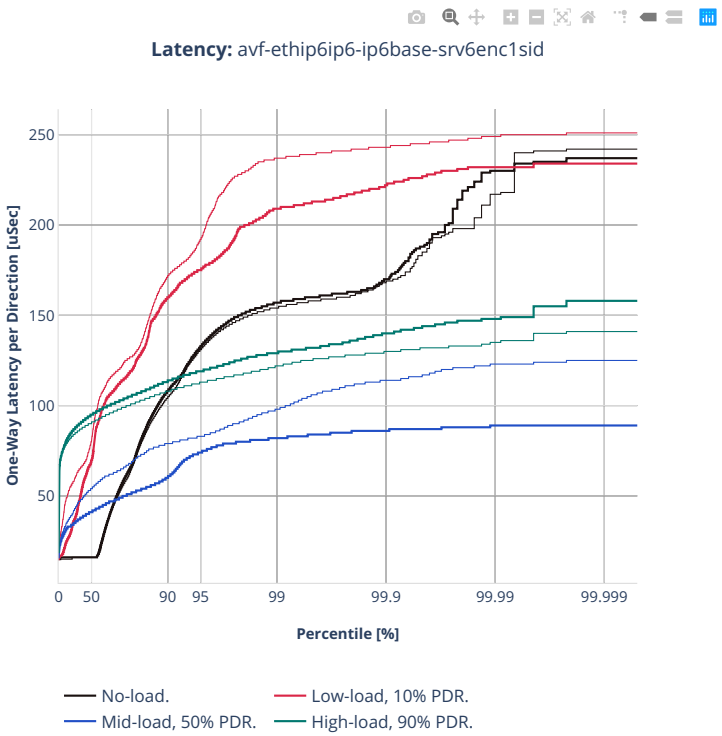
2.5.4 SRv6 Routing

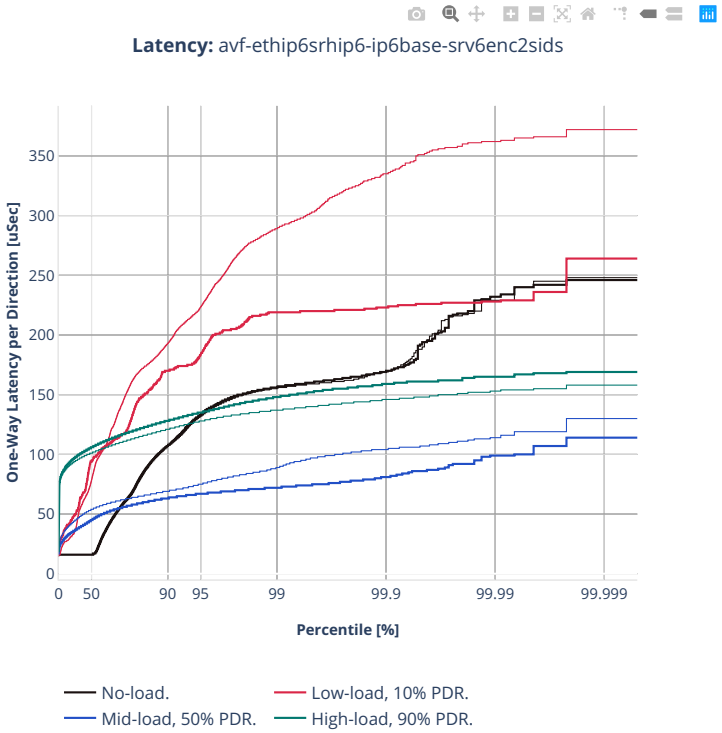
CSIT source code for the test cases used for plots can be found in CSIT git repository[153].

[153] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls2210

3n-icx-xxv710

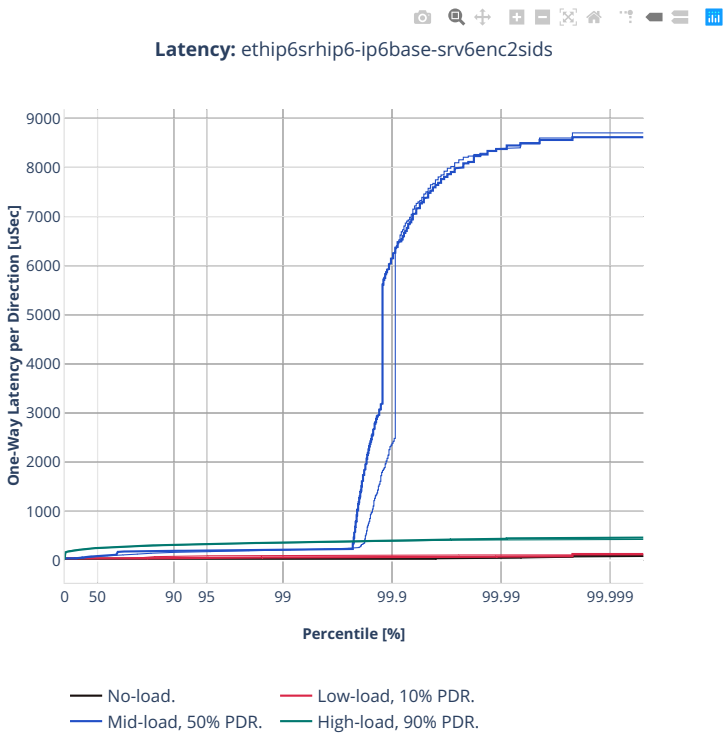
78b-2t1c-srv6-ip6routing-base-avf




Latency: avf-ethip6srhip6-ip6base-srv6enc2sids
One-Way Latency per Direction [uSec]
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3n-tsh-x520

78b-1t1c-srv6-ip6routing-base-ixgbe



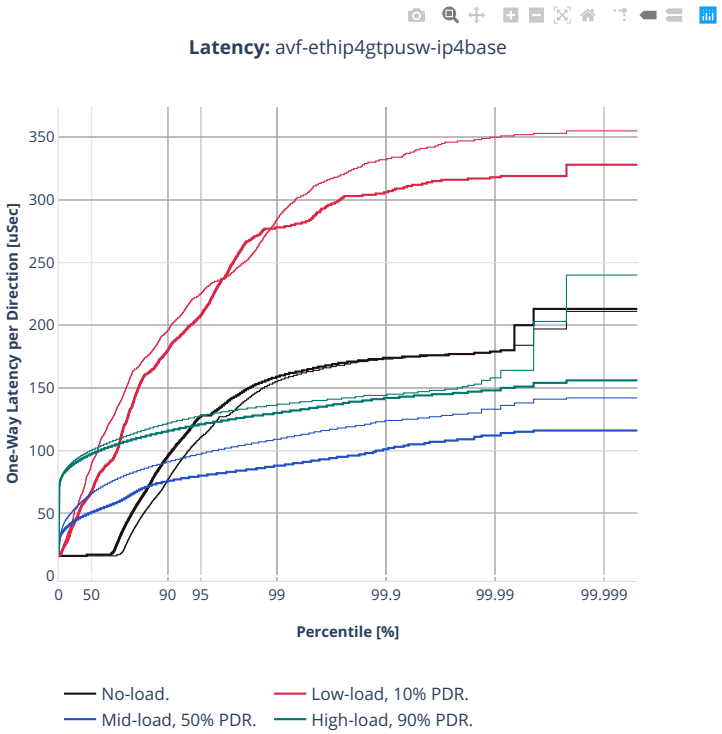
2.5.5 IPv4 Tunnels

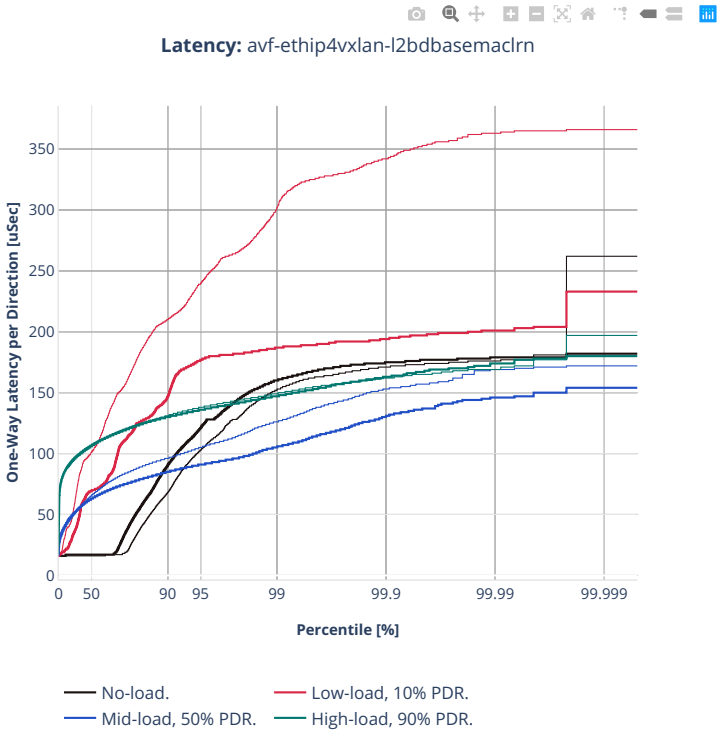
CSIT source code for the test cases used for plots can be found in CSIT git repository[154].

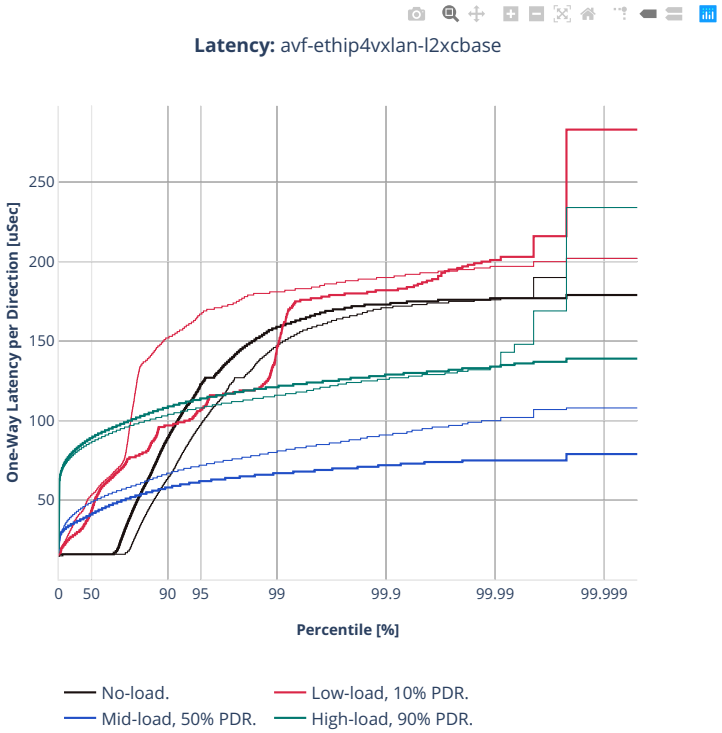
[154] https://git.fd.io/csit/tree/tests/vpp/perf/ip4_tunnels?h=rls2210

3n-icx-xxv710

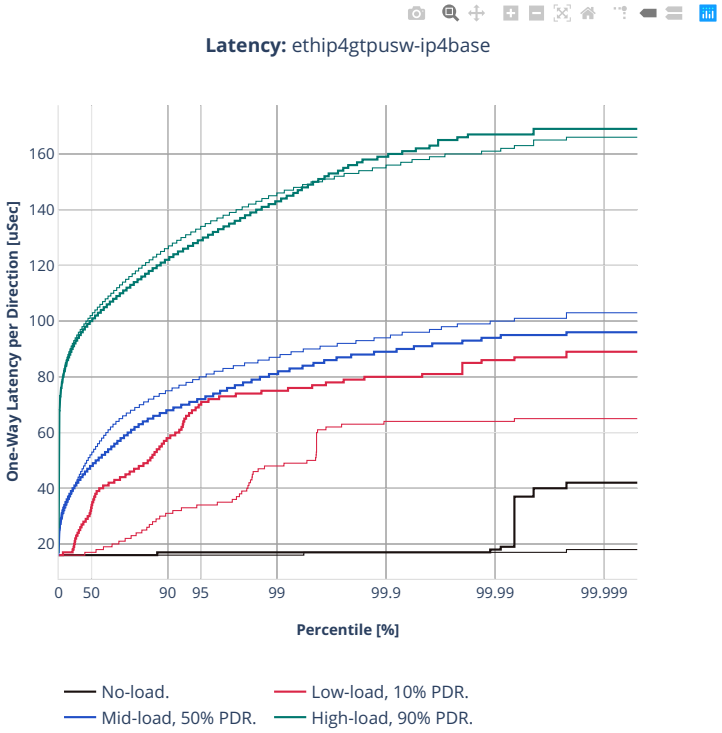
64b-2t1c-ip4tunnel-base-avf

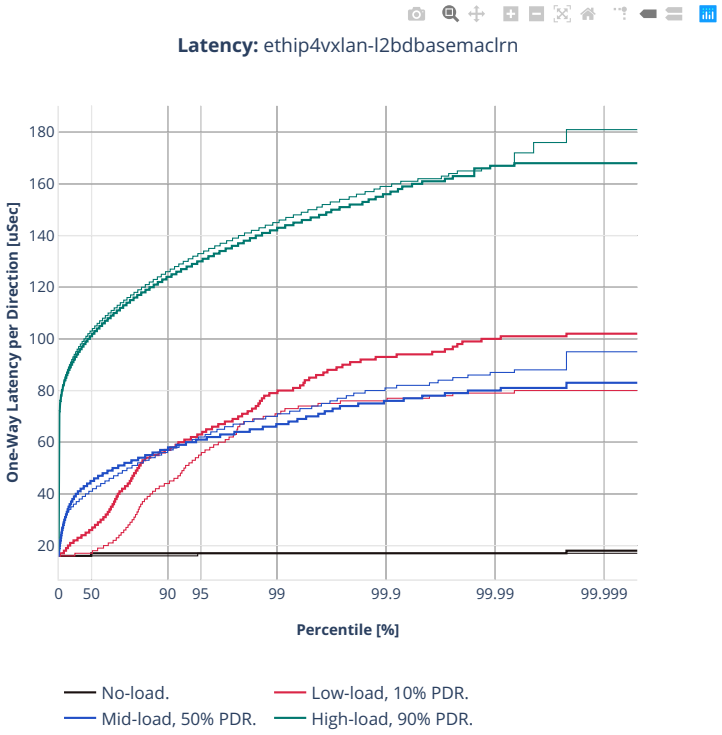


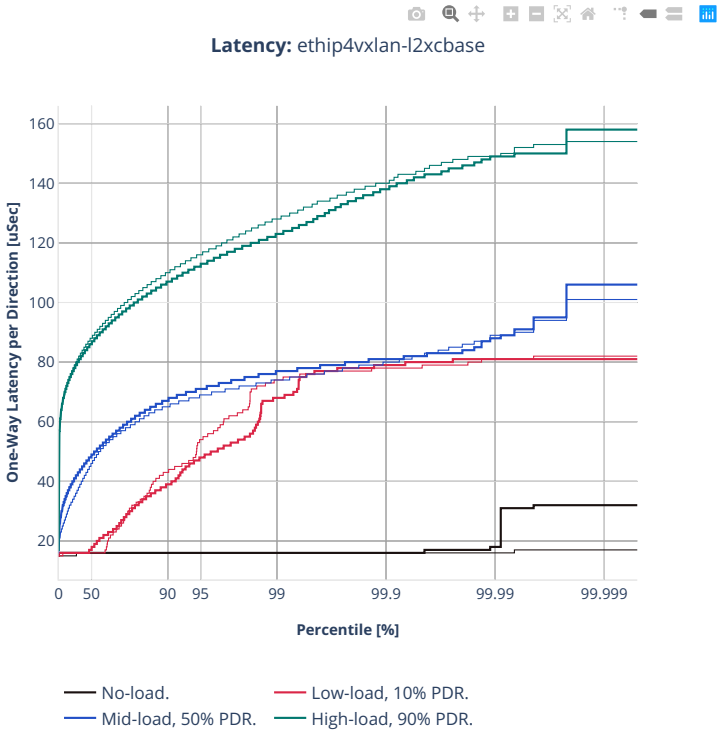

Latency: avf-ethip4vxlan-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4vxlan-l2xcbase
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

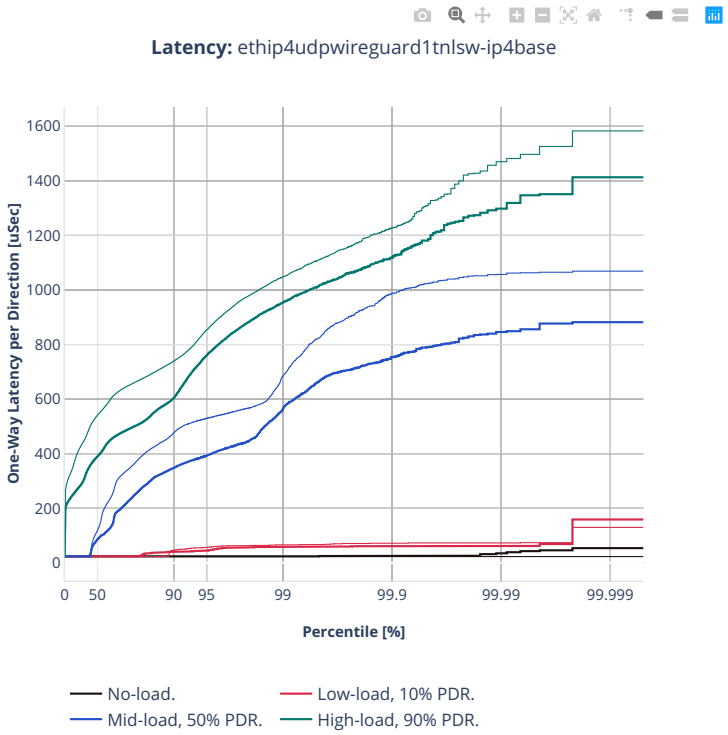
64b-2t1c-ip4tunnel-base-dpdk

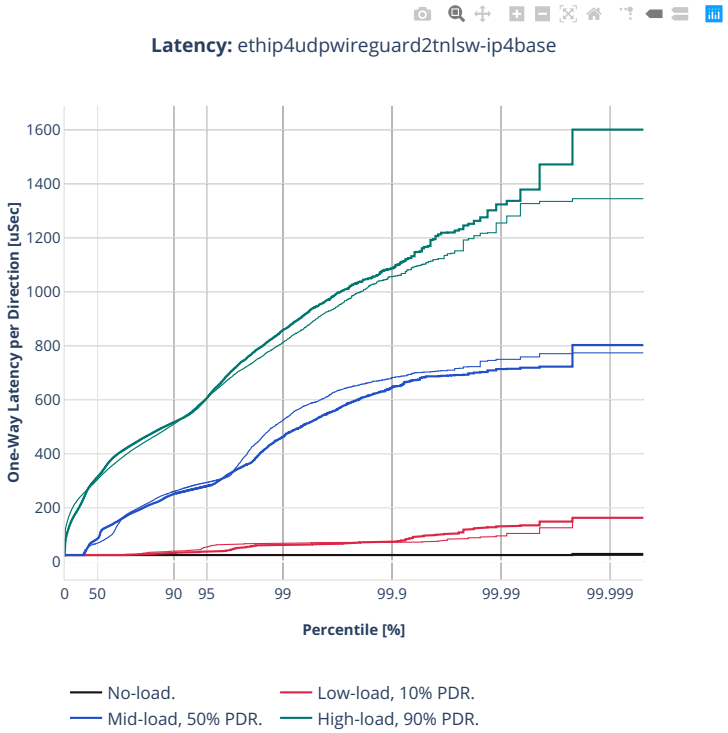


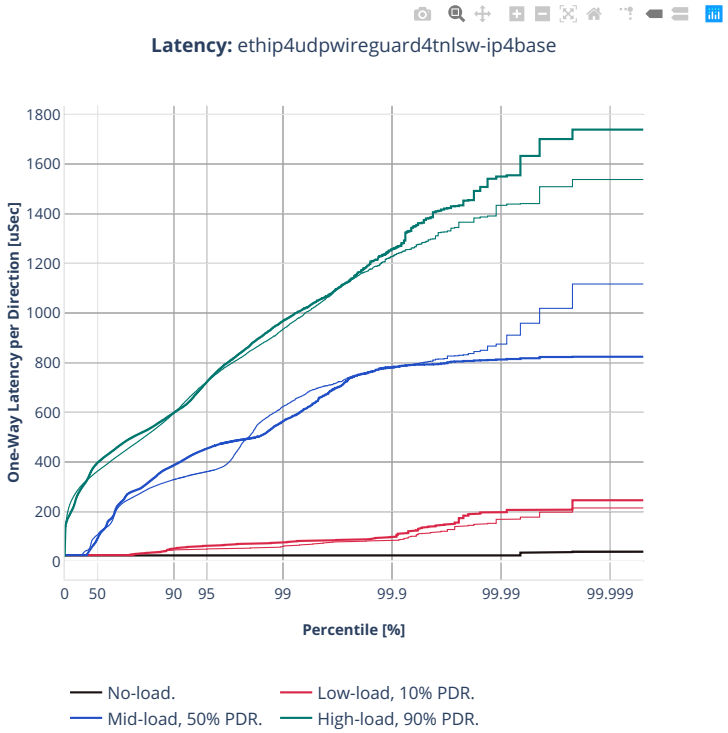

Latency: ethip4vxlan-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

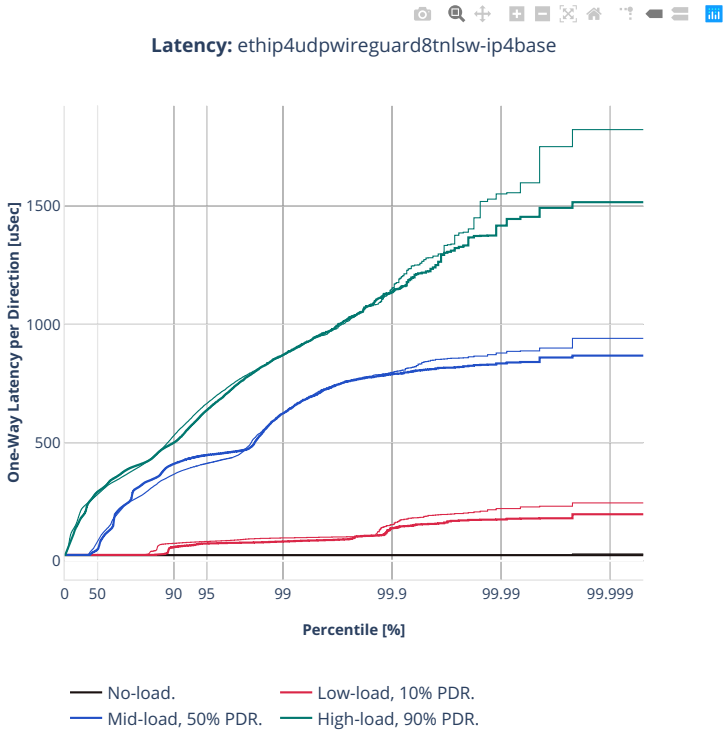

Latency: ethip4vxlan-l2xcbase
One-Way Latency per Direction [uSec]
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

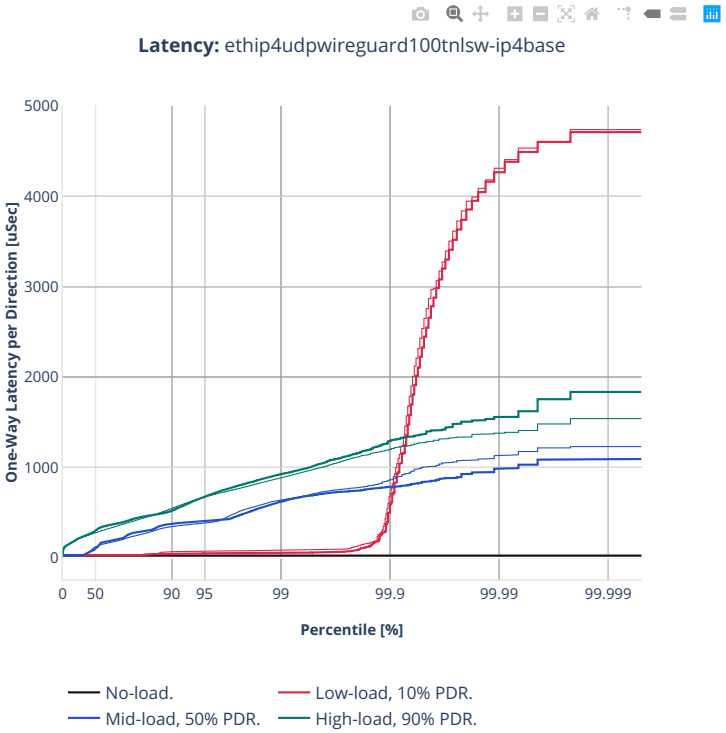
64b-2t1c-ip4tunnel-wireguard

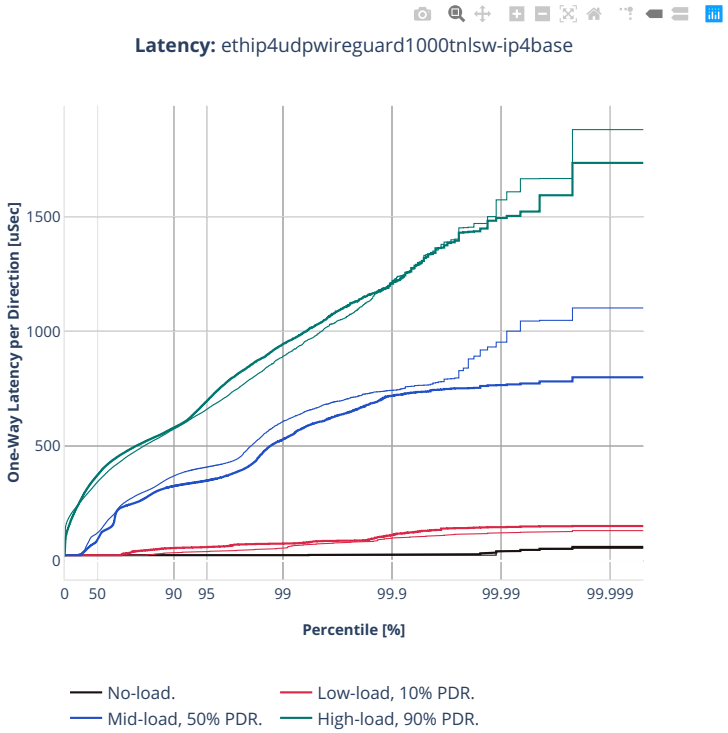



Latency: ethip4udpwireguard2tnlsw-ip4base
One-Way Latency per Direction [uSec]
1600
1400
1200
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4udpwireguard4tnlsw-ip4base
One-Way Latency per Direction [uSec]
1800
1600
1400
1200
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

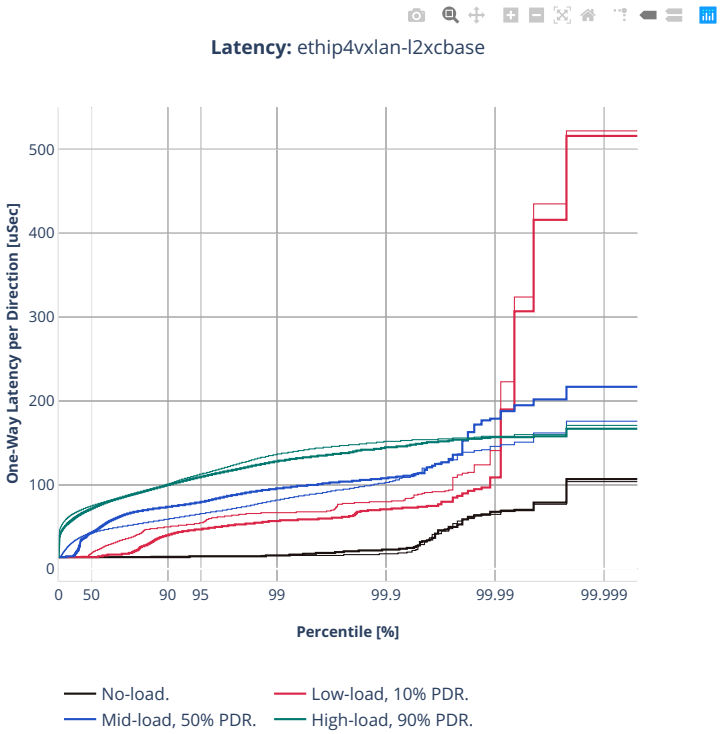

Latency: ethip4udpwireguard8tnlsw-ip4base
One-Way Latency per Direction [uSec]
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

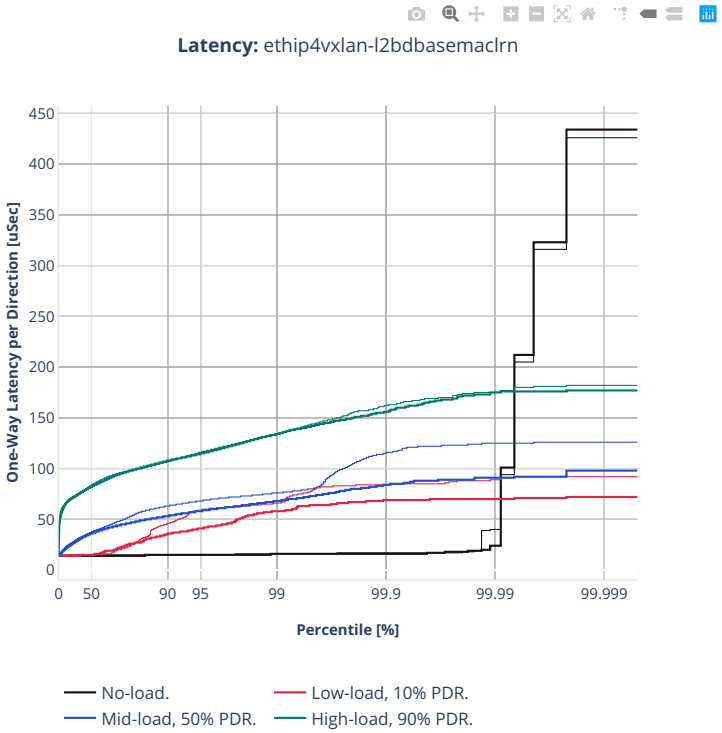

Latency: ethip4udpwireguard100tnlsw-ip4base
One-Way Latency per Direction [uSec]
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4udpwireguard1000tnlsw-ip4base
One-Way Latency per Direction [uSec]
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3n-alt-xl710

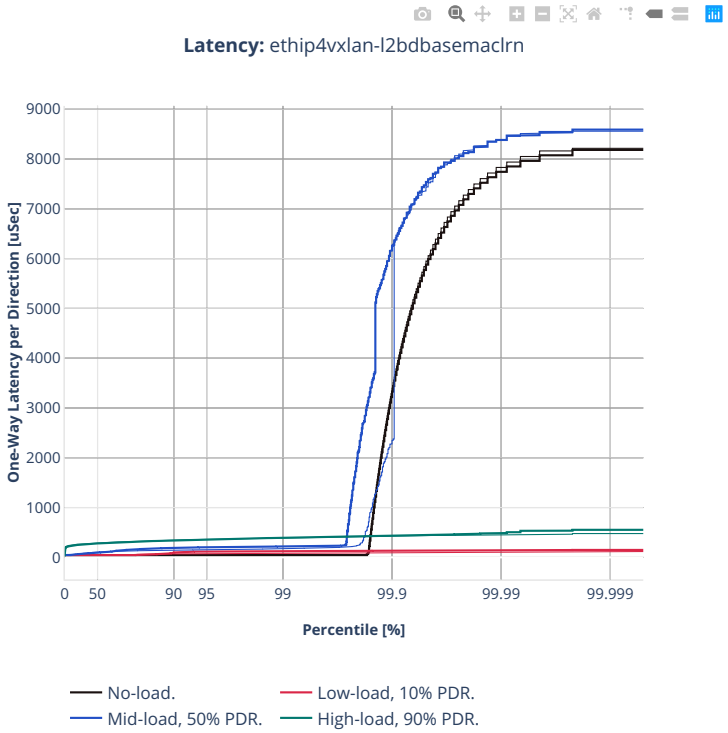
64b-1t1c-ip4tunnel-base




Latency: ethip4vxlan-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
450
400
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3n-tsh-x520

64b-1t1c-ip4tunnel-base-scale-ixgbe


Latency: ethip4vxlan-l2bdbasemaclrn
One-Way Latency per Direction [uSec]
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

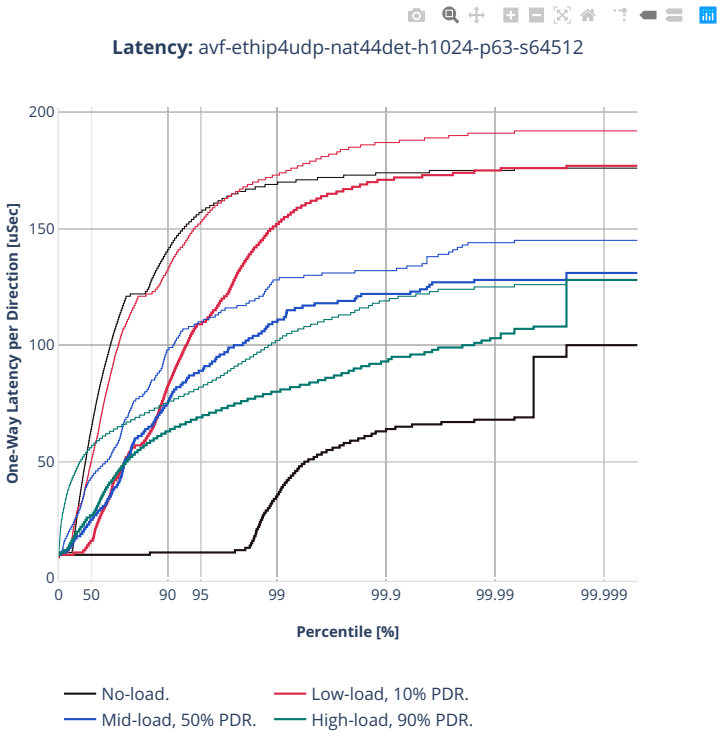
2.5.6 NAT44 IPv4 Routing

CSIT source code for the test cases used for plots can be found in CSIT git repository[155].

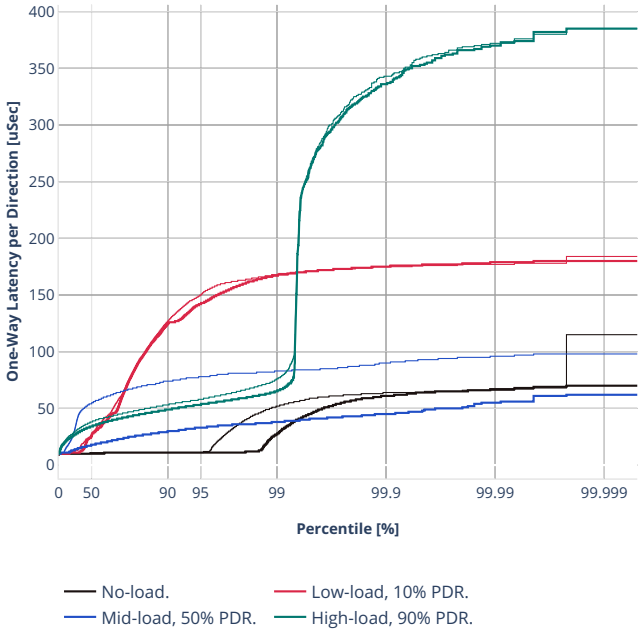
[155] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls2210

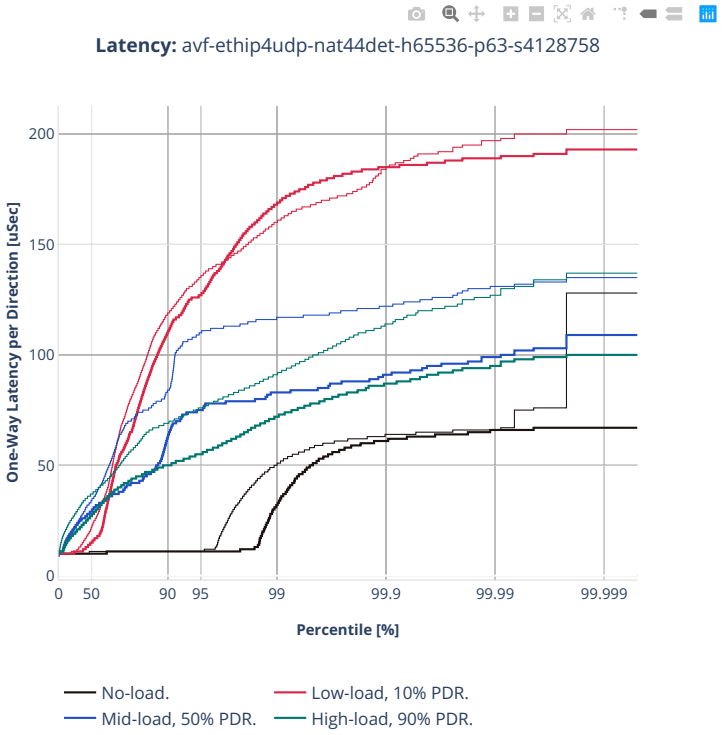
2n-icx-xxv710

64b-2t1c-ethip4udp-nat44det-avf

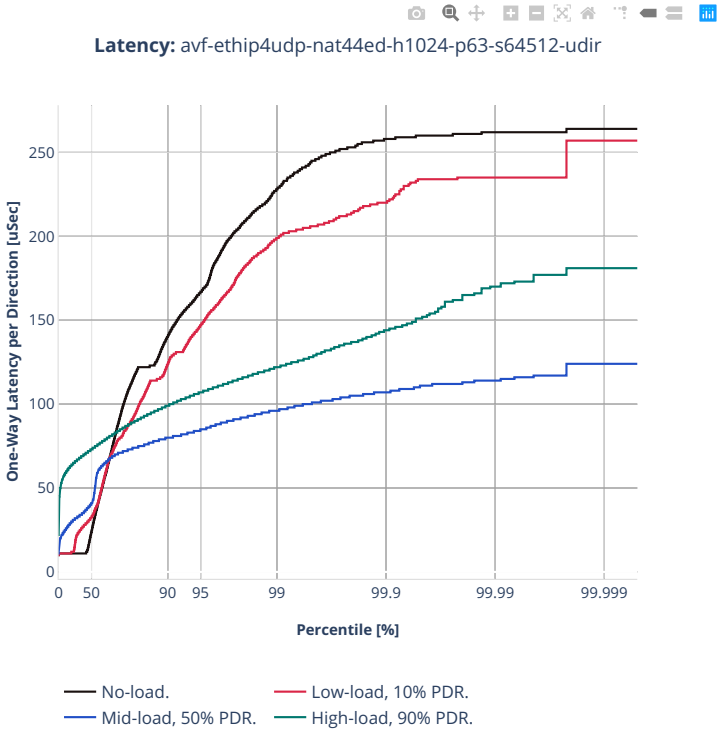


Latency: avf-ethip4udp-nat44det-h16384-p63-s1032192



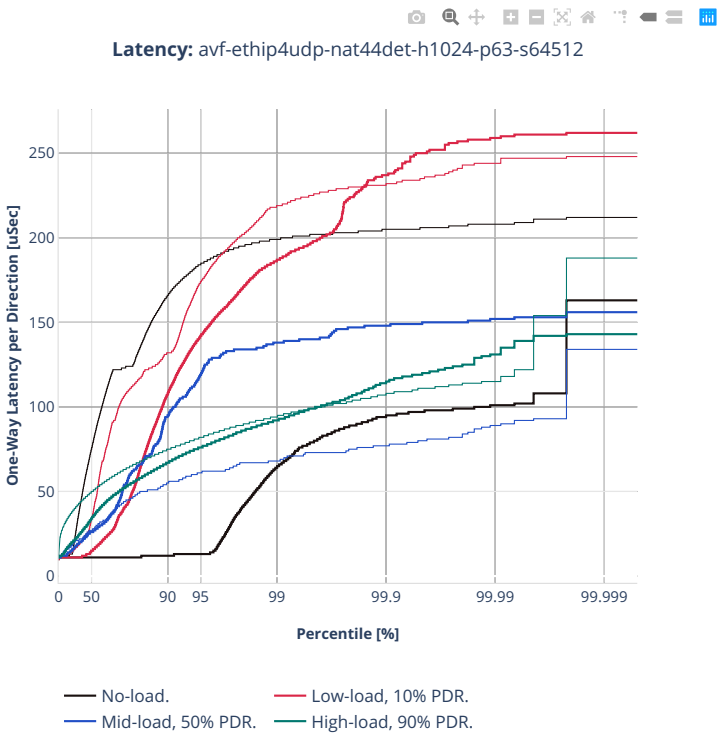

Latency: avf-ethip4udp-nat44det-h65536-p63-s4128758
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-ethip4udp-nat44ed-udir-avf

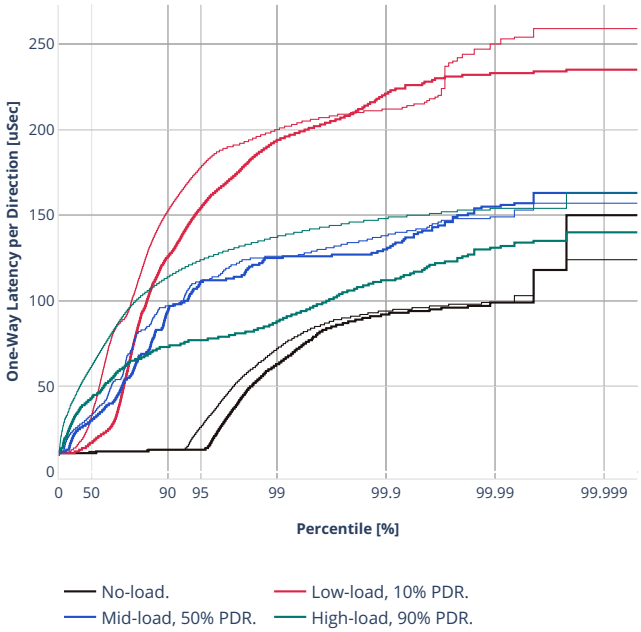


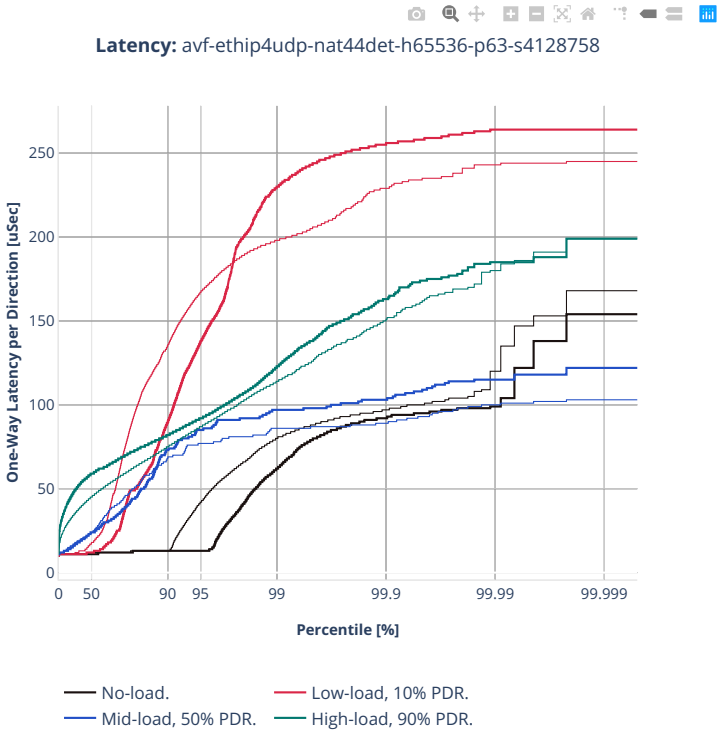
2n-clx-xxv710

64b-2t1c-ethip4udp-nat44det-avf

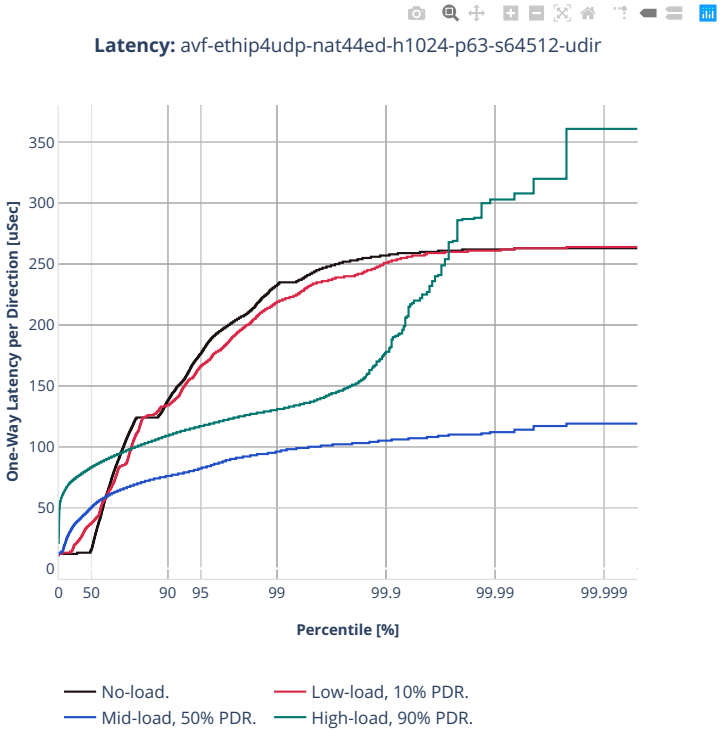


Latency: avf-ethip4udp-nat44det-h16384-p63-s1032192



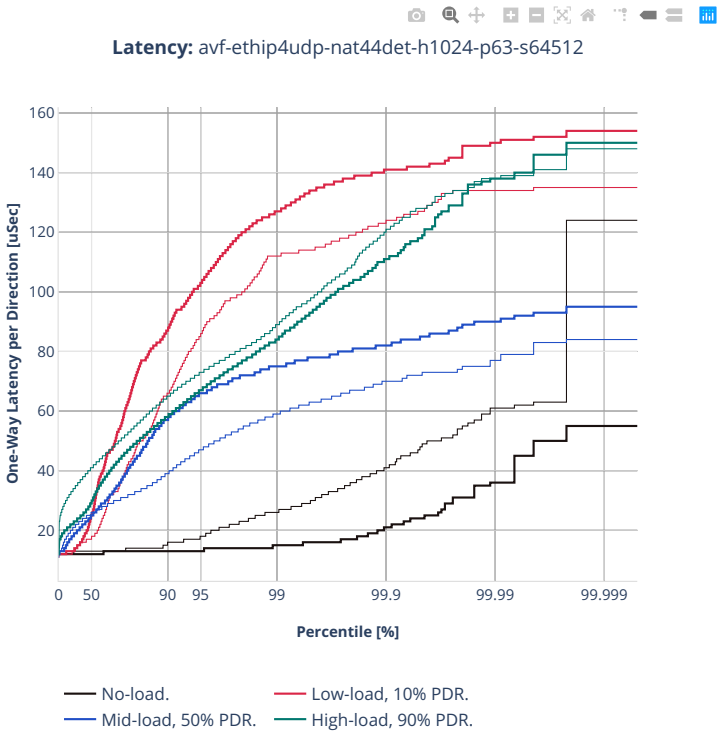

Latency: avf-ethip4udp-nat44det-h65536-p63-s4128758
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-ethip4udp-nat44ed-avf



2n-zn2-xxv710

64b-2t1c-ethip4udp-nat44det-avf



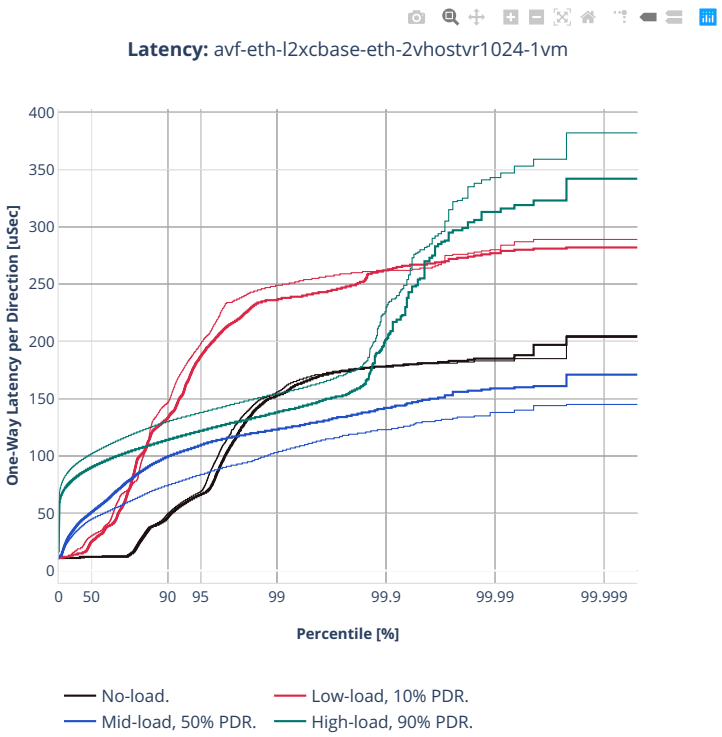
2.5.7 KVM VMs vhost-user

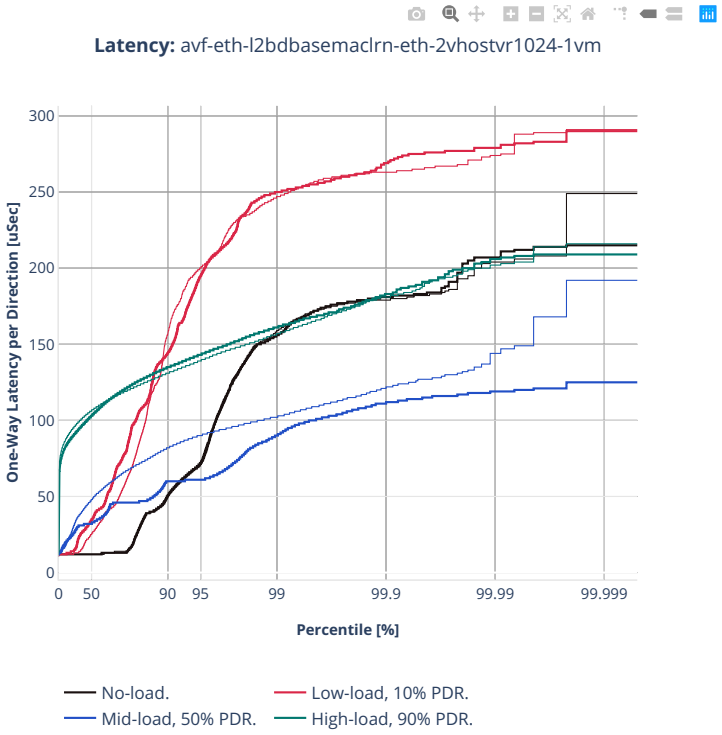
CSIT source code for the test cases used for plots can be found in CSIT git repository[156].

[156] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls2210

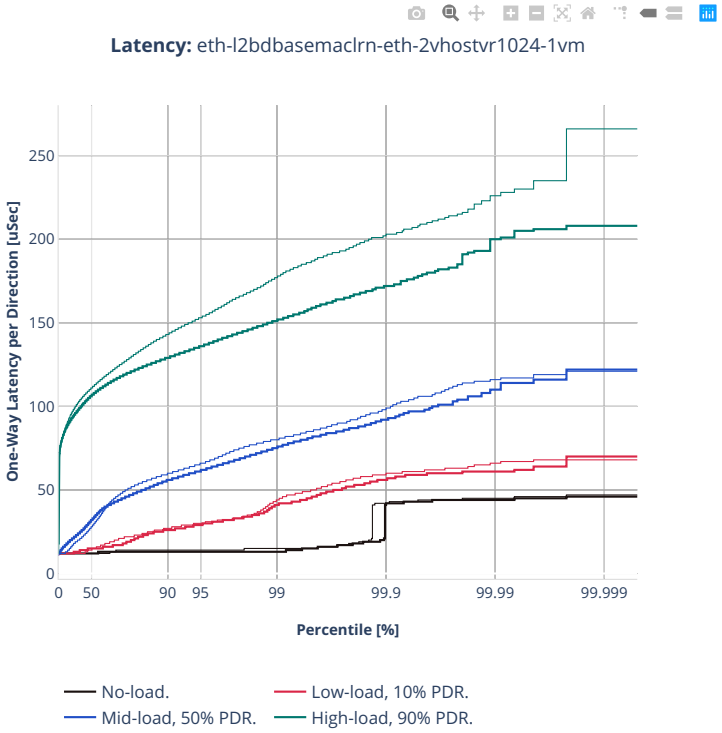
2n-icx-xxv710

64b-2t1c-vhost-base-avf-testpmd

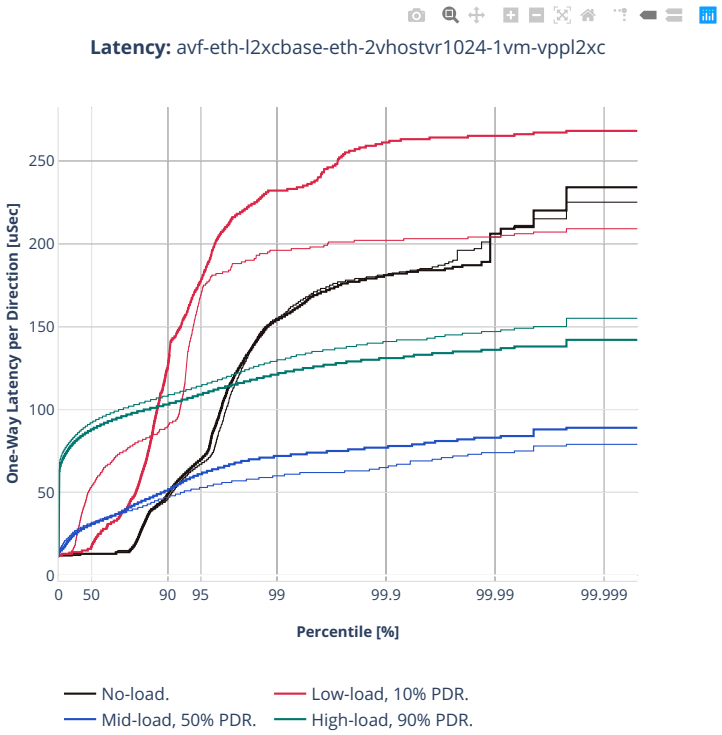


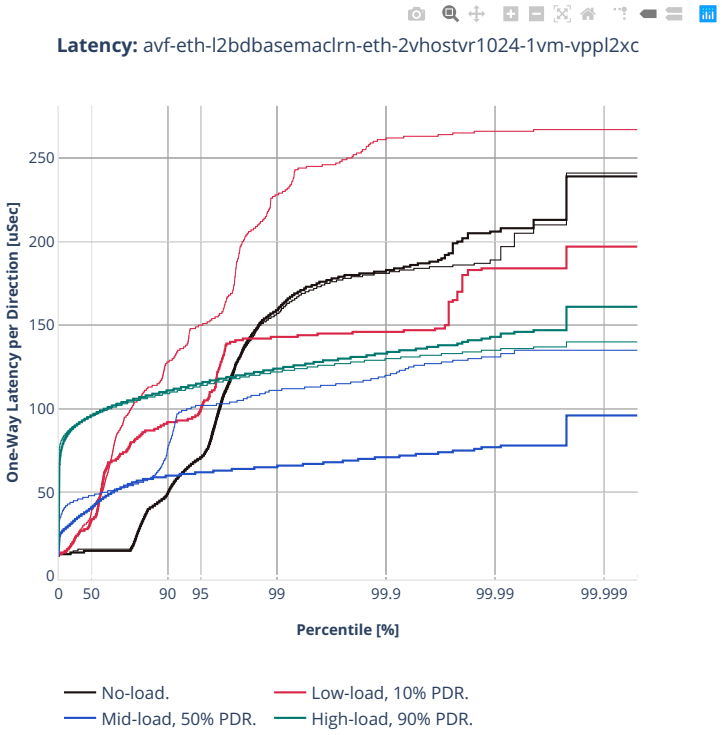

Latency: avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-vhost-base-dpdk-testpmd

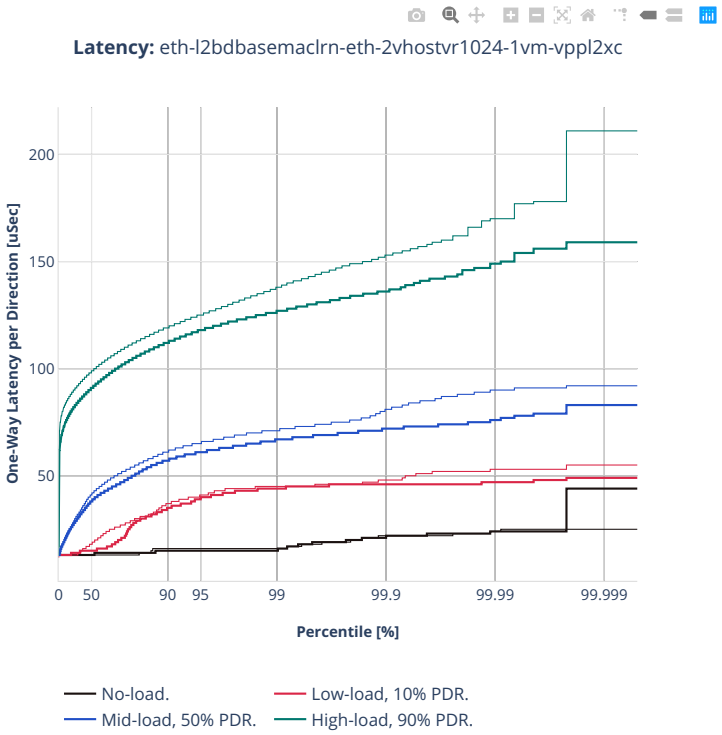


64b-2t1c-vhost-base-avf-vpp



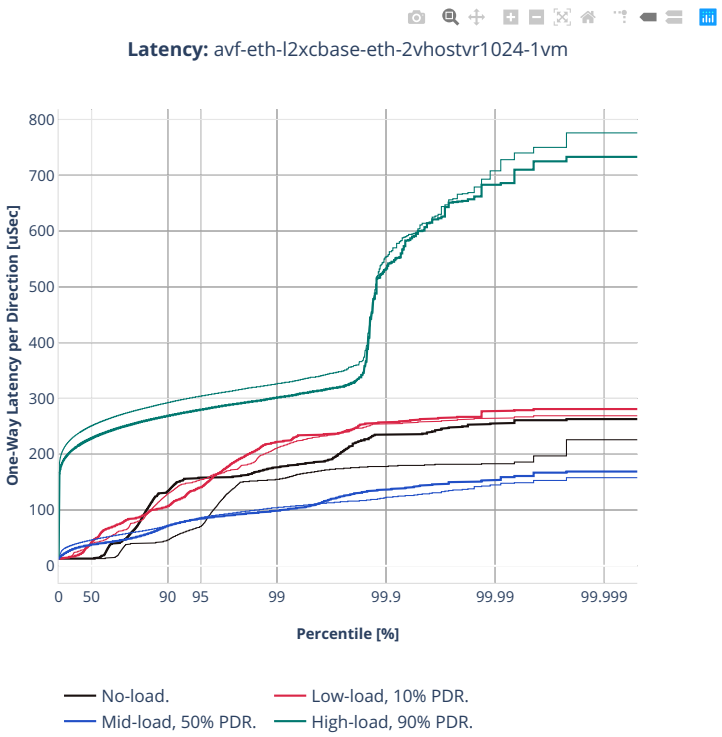

Latency: avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

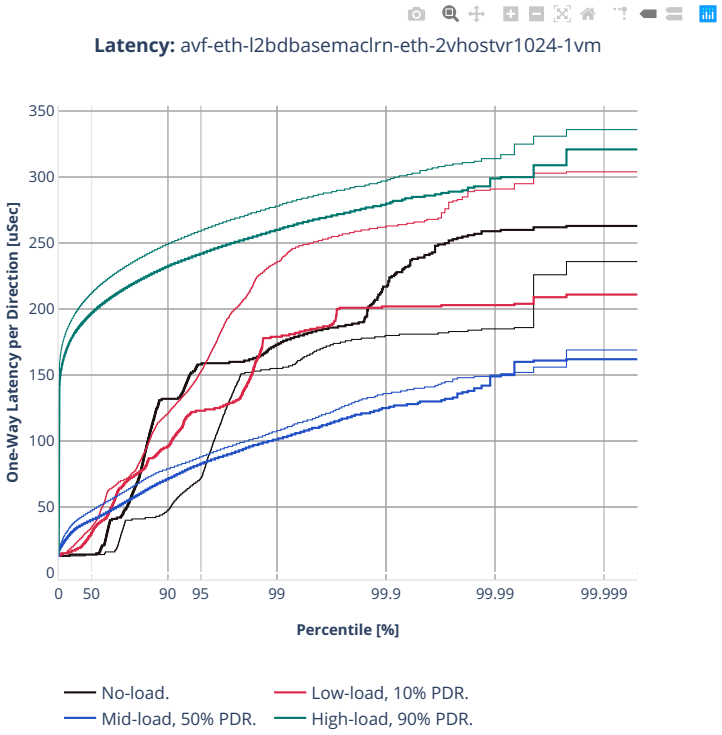
64b-2t1c-vhost-base-dpdk-vpp



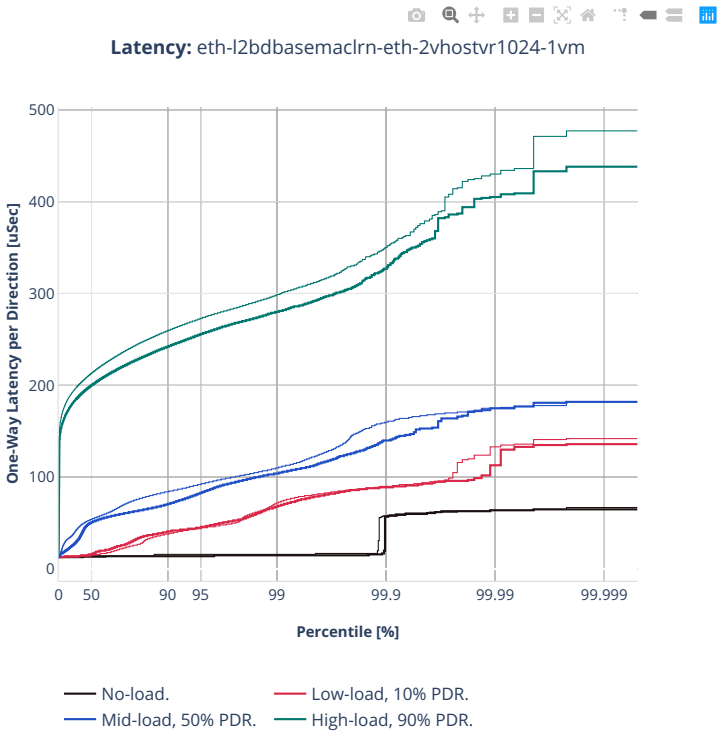
2n-clx-xxv710

64b-2t1c-vhost-base-avf-testpmd

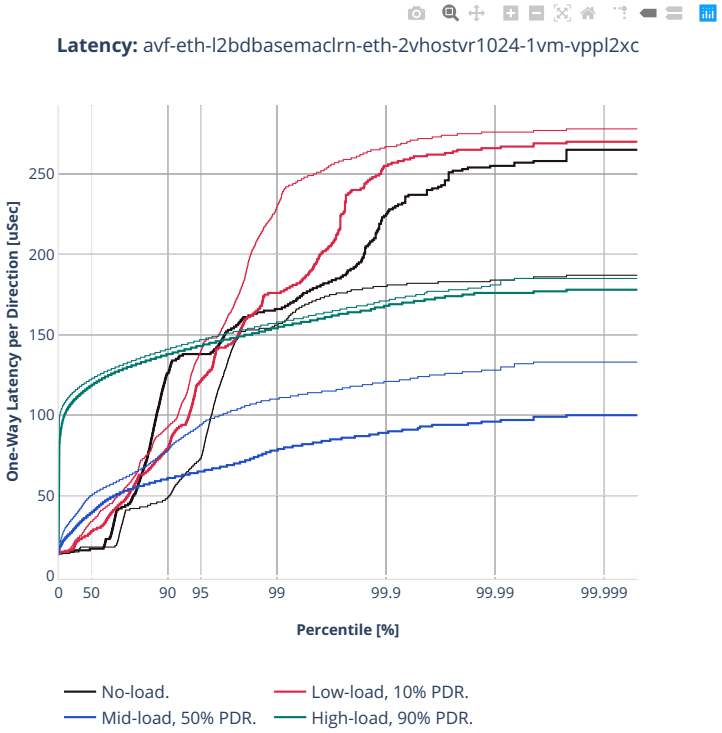



Latency: avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
One-Way Latency per Direction [uSec]
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

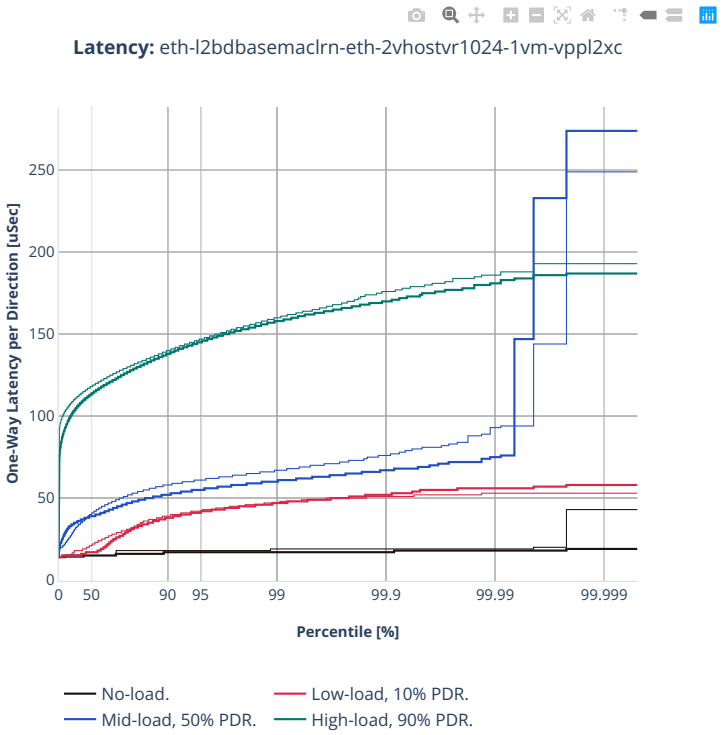
64b-2t1c-vhost-base-dpdk-testpmd



64b-2t1c-vhost-base-avf-vpp

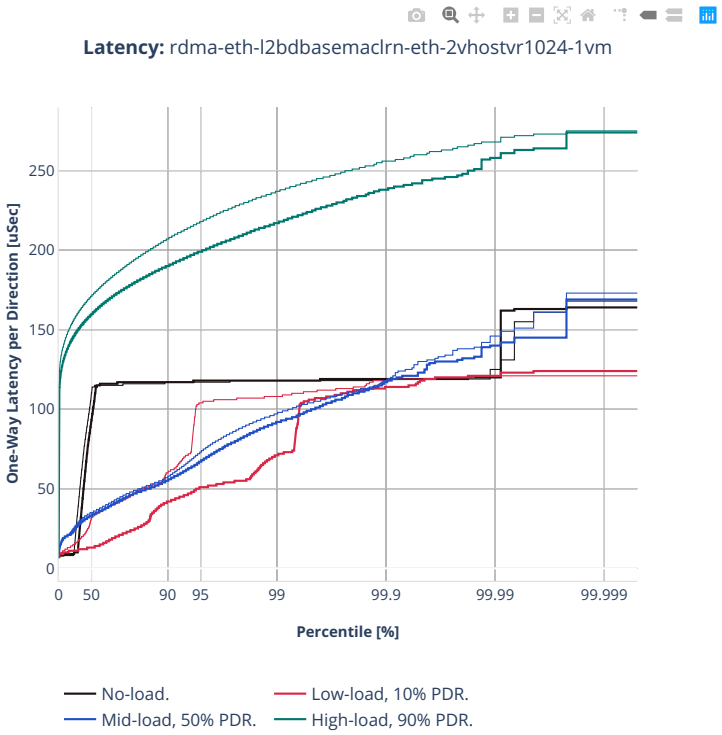


64b-2t1c-vhost-base-dpdk-vpp

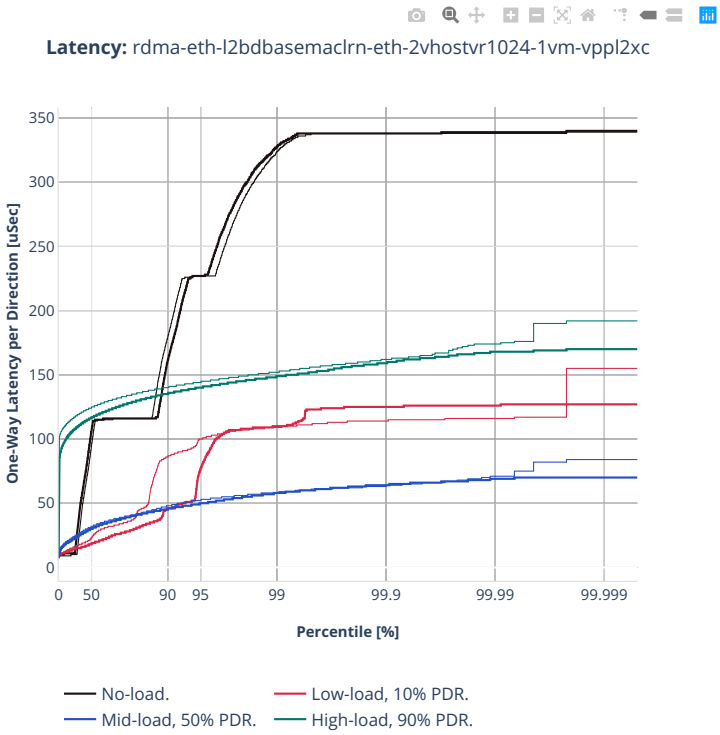


2n-clx-cx556a

64b-2t1c-vhost-base-rdma-testpmd

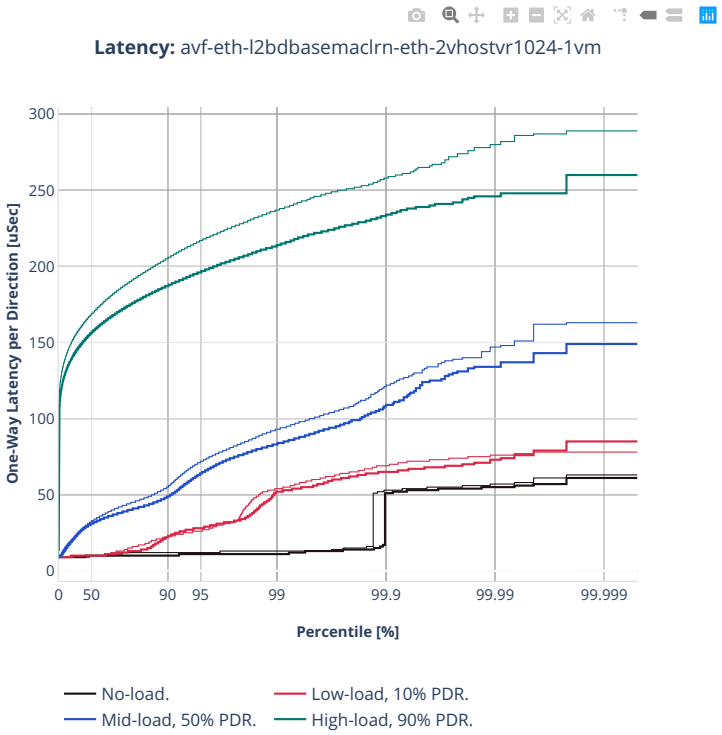


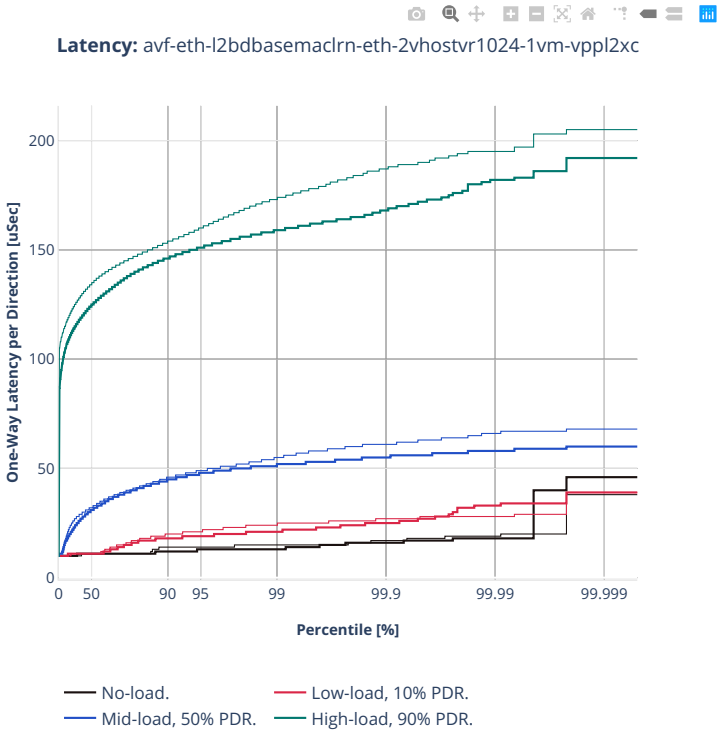
64b-2t1c-vhost-base-rdma-vpp

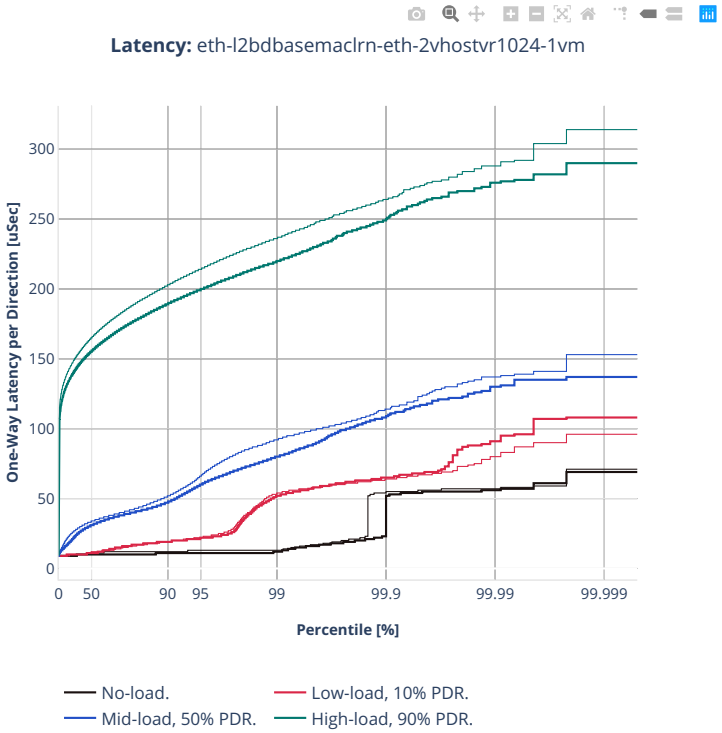


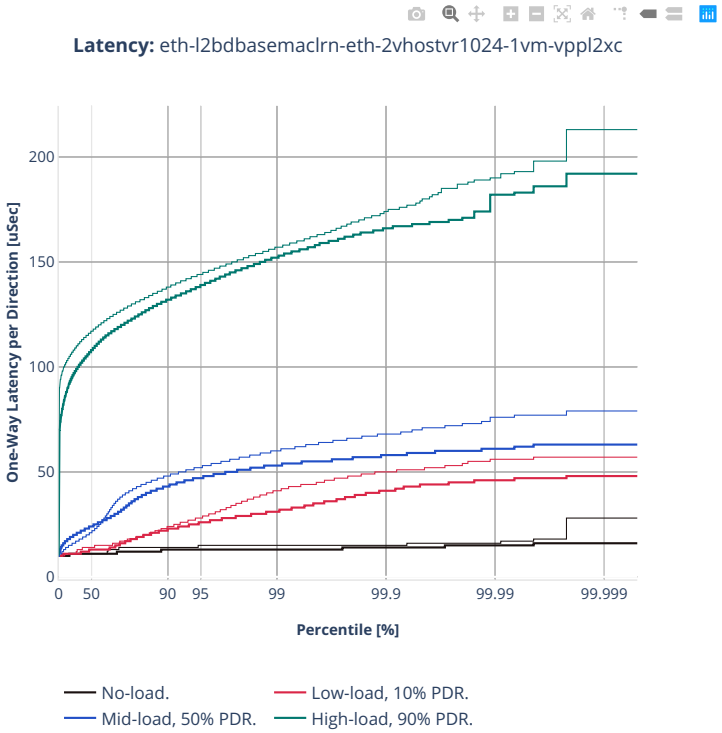
2n-clx-e810cq

64b-2t1c-vhost-base



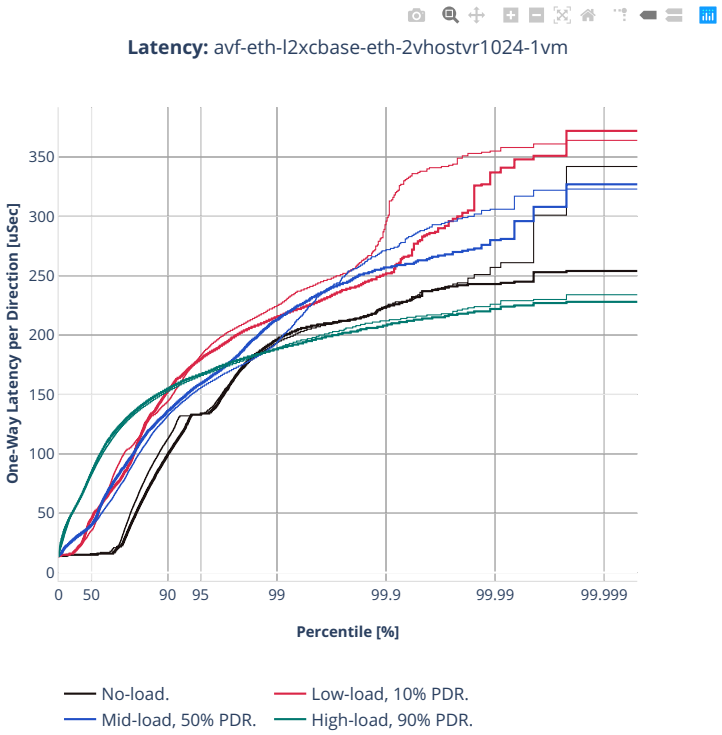

Latency: avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

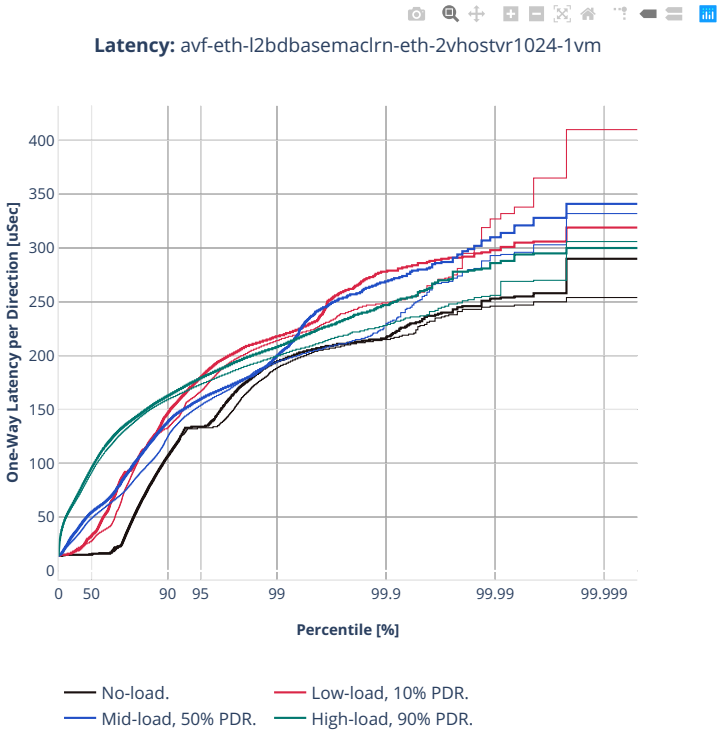

Latency: eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

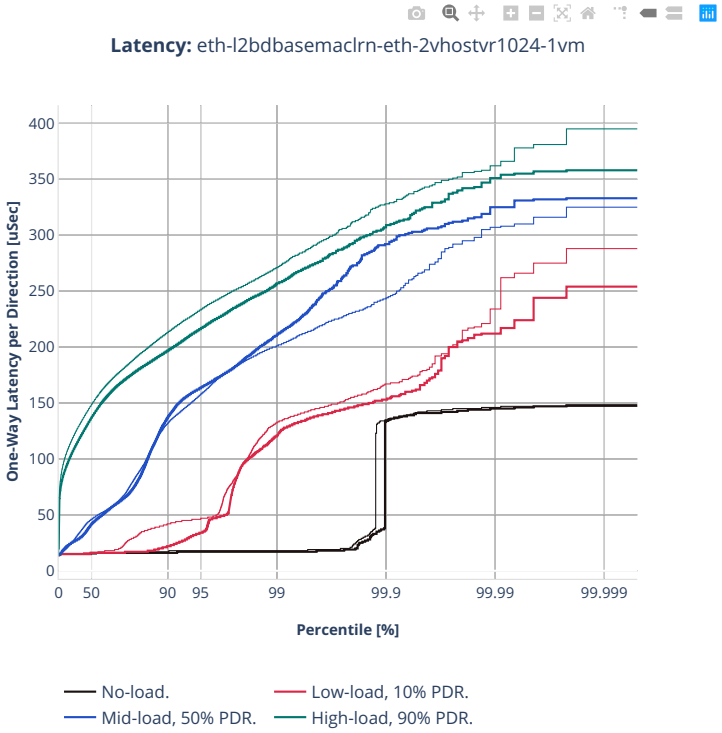
2n-zn2-xxv710

64b-2t1c-vhost-base-avf-testpmd

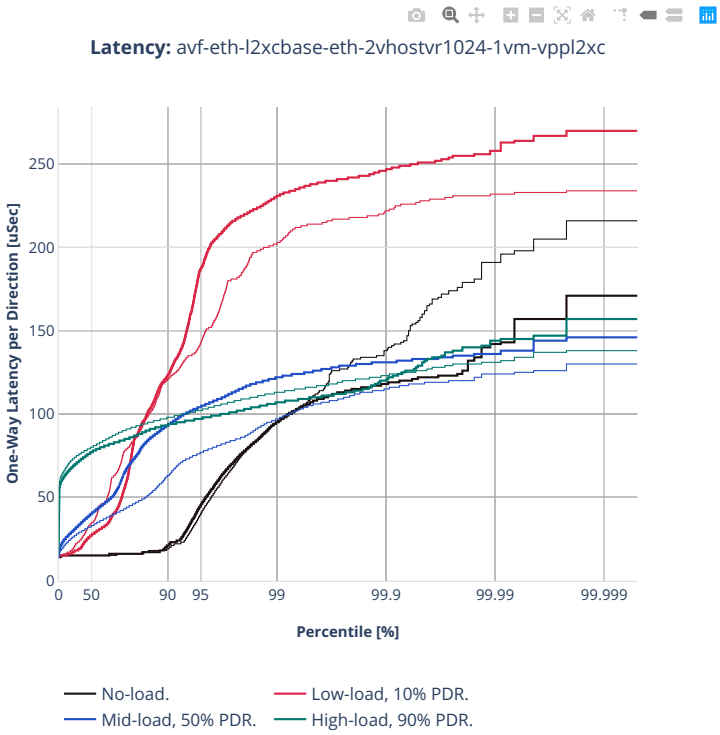



Latency: avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm
One-Way Latency per Direction [uSec]
400
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

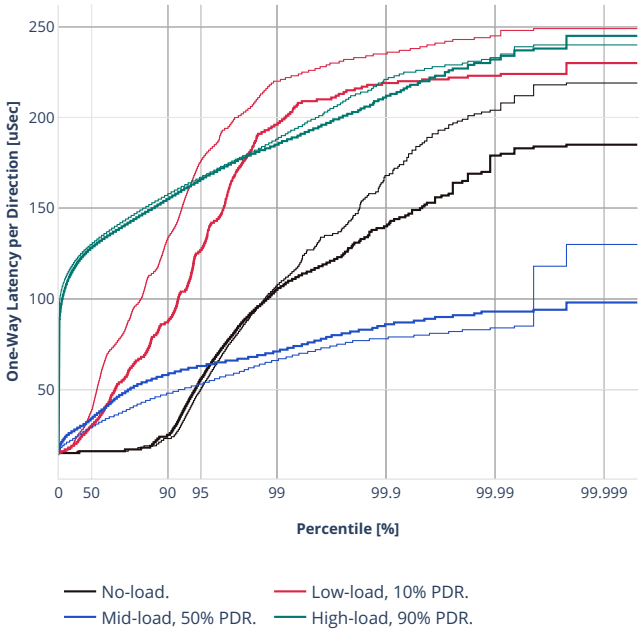
64b-2t1c-vhost-base-dpdk-testpmd



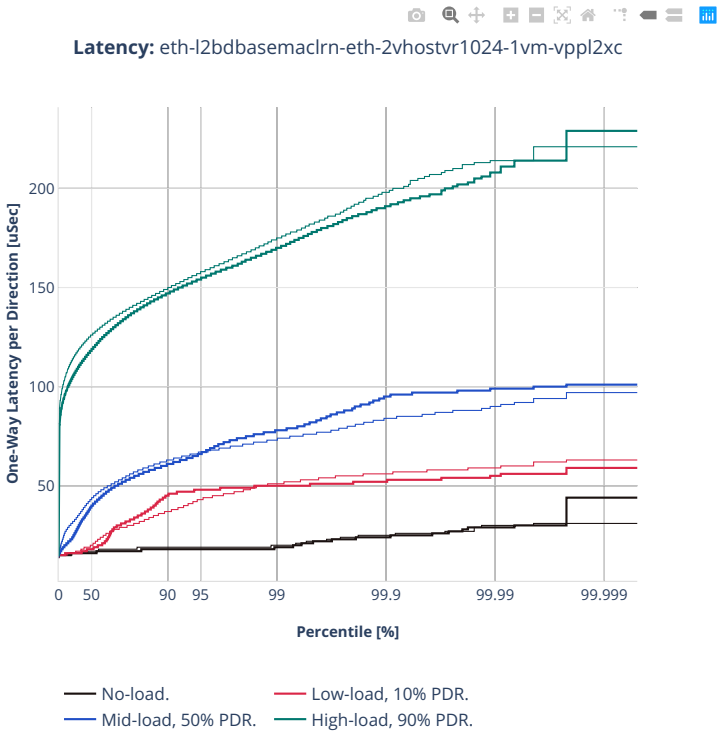
64b-2t1c-vhost-base-avf-vpp



Latency: avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc

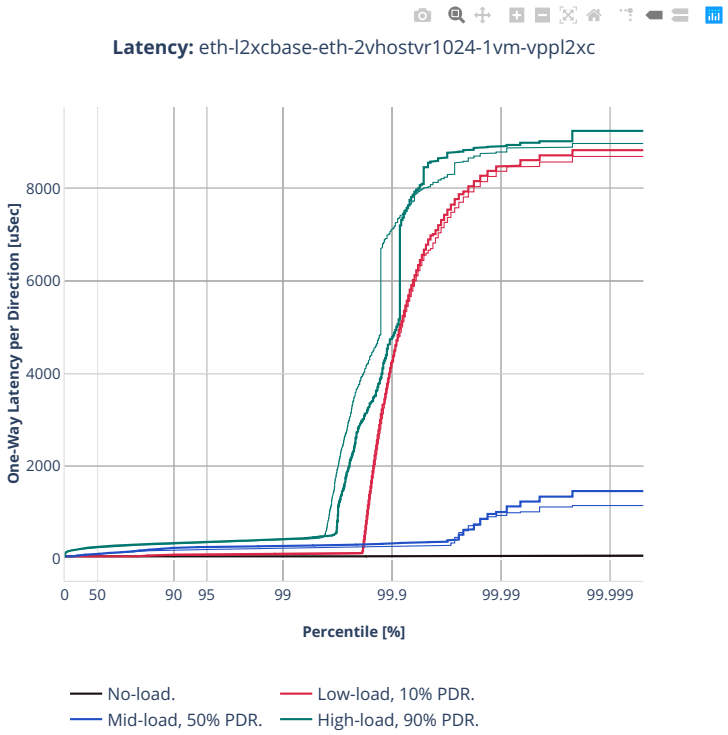


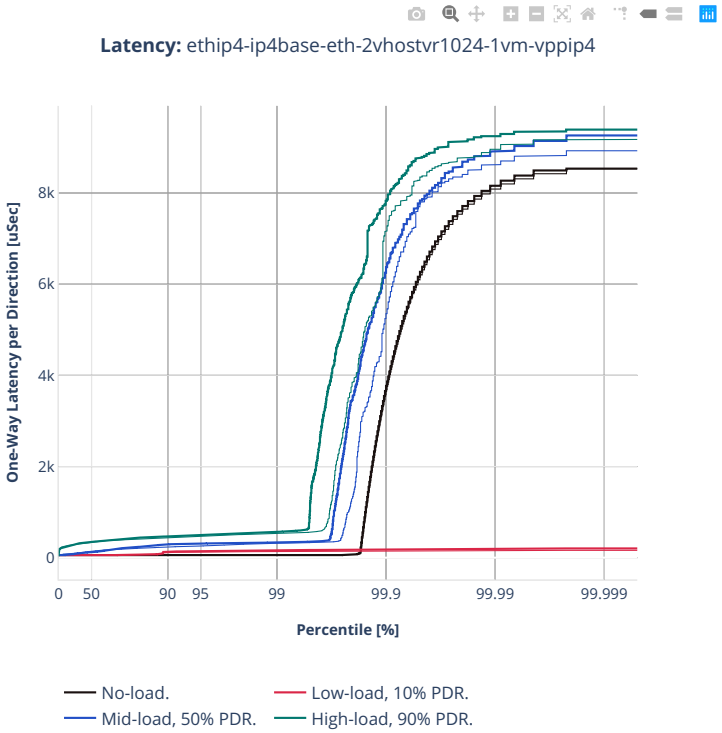
64b-2t1c-vhost-base-dpdk-vpp

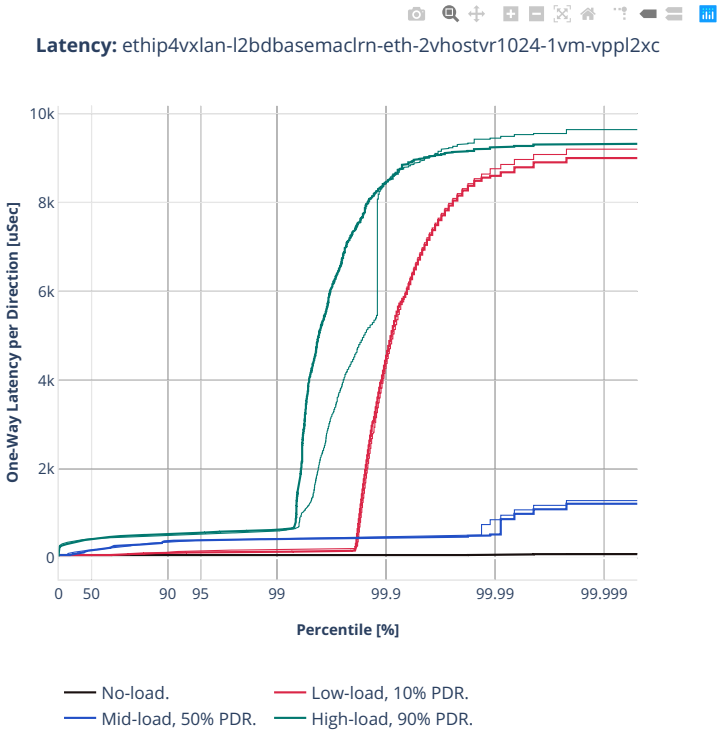


3n-tsh-x520

64b-1t1c-vhost-base-ixgbe




Latency: ethip4-ip4base-eth-2vhostvr1024-1vm-vppip4
One-Way Latency per Direction [uSec]
8k
6k
4k
2k
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4vxlan-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc
10k
8k
6k
4k
2k
0
One-Way Latency per Direction [uSec]
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

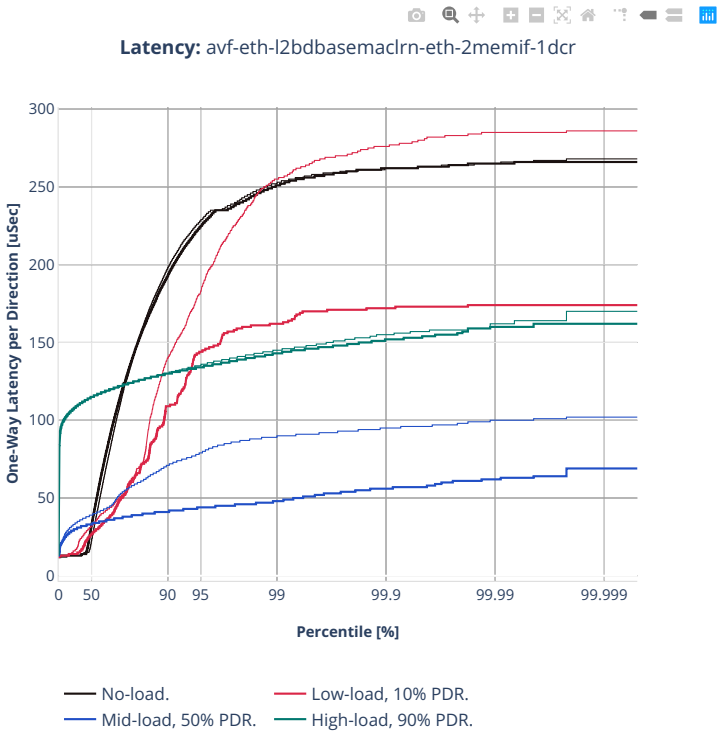
2.5.8 LXC/DRC Container Memif

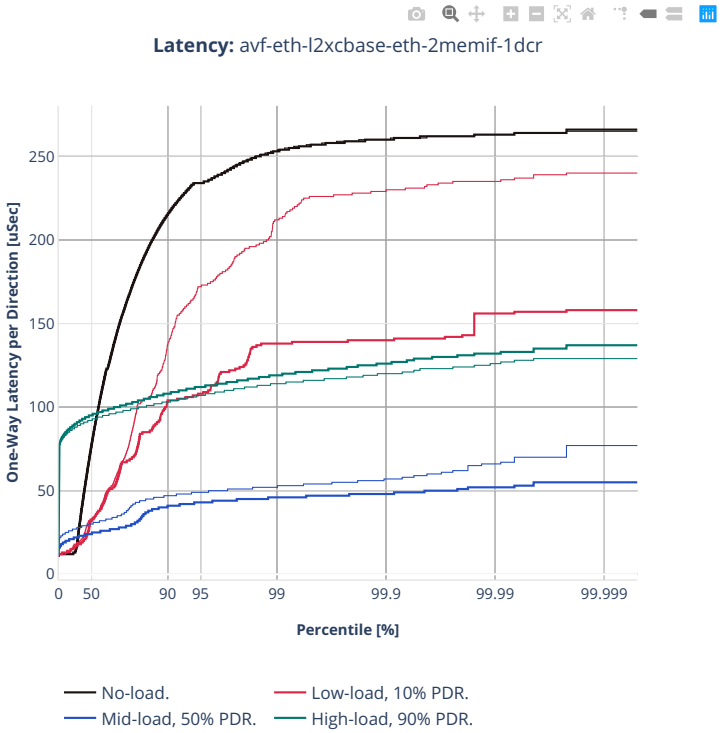
CSIT source code for the test cases used for plots can be found in CSIT git repository[157].

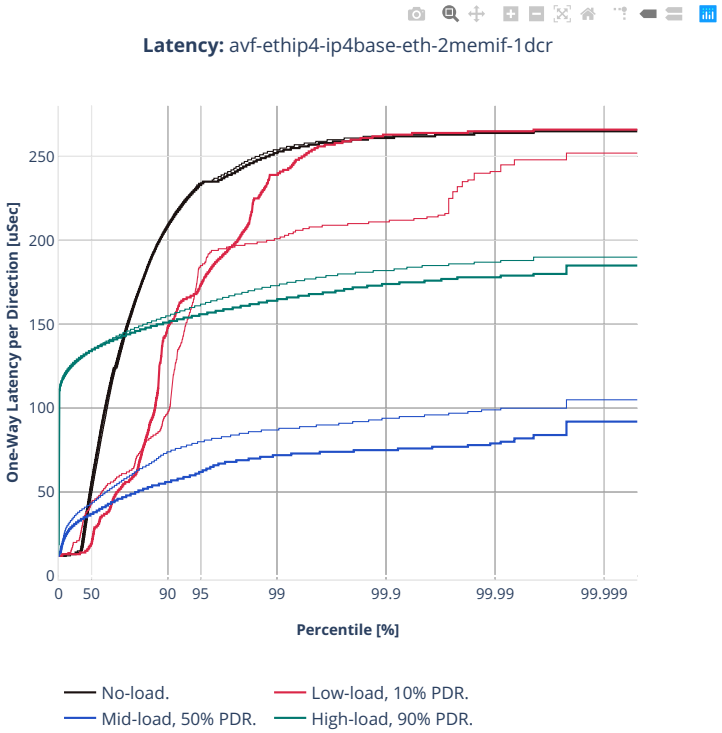
[157] https://git.fd.io/csit/tree/tests/vpp/perf/container_memif?h=rls2210

2n-icx-xxv710

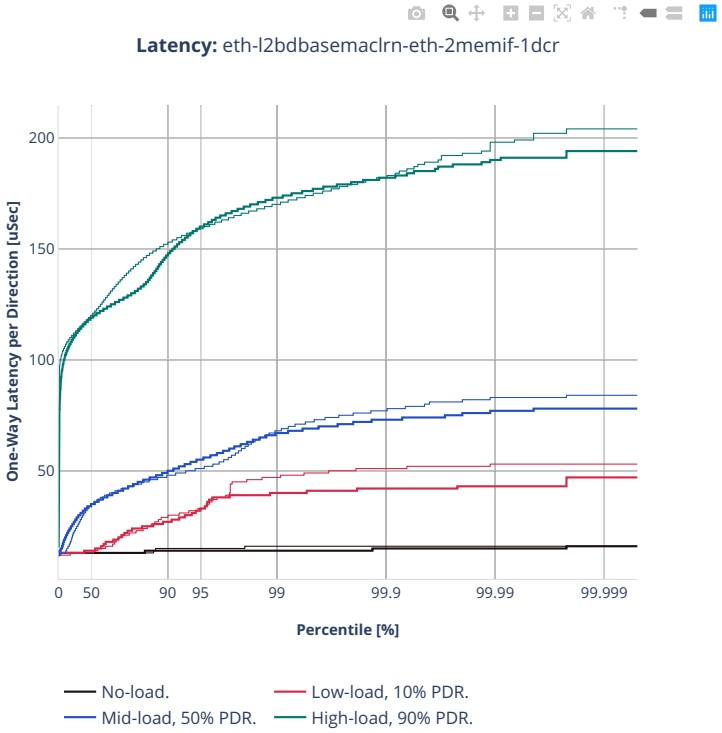
64b-2t1c-memif-base-avf

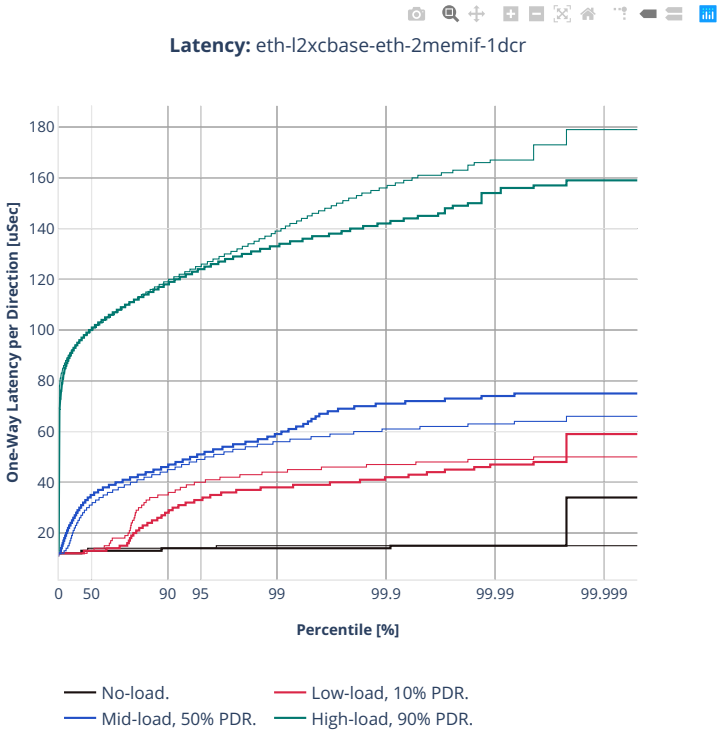


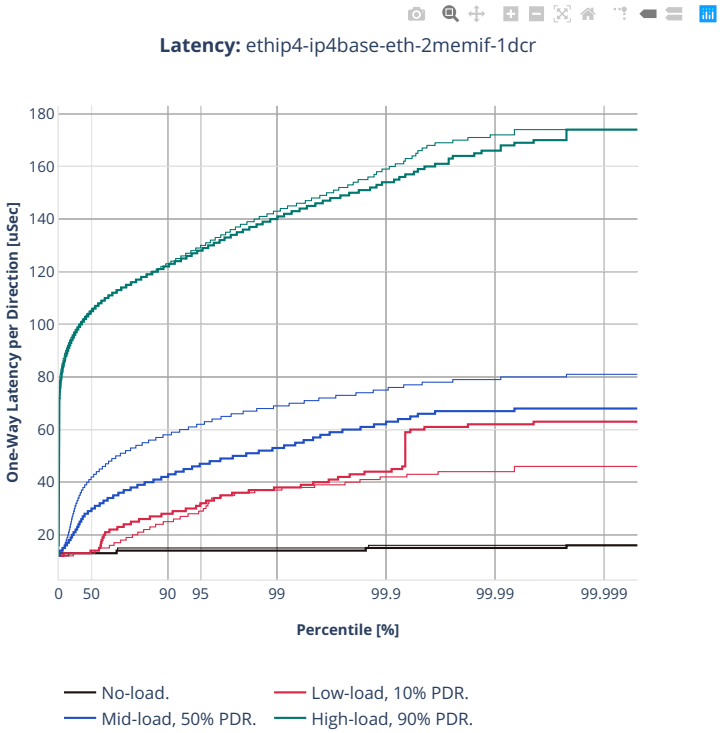

Latency: avf-eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-memif-base-dpdk

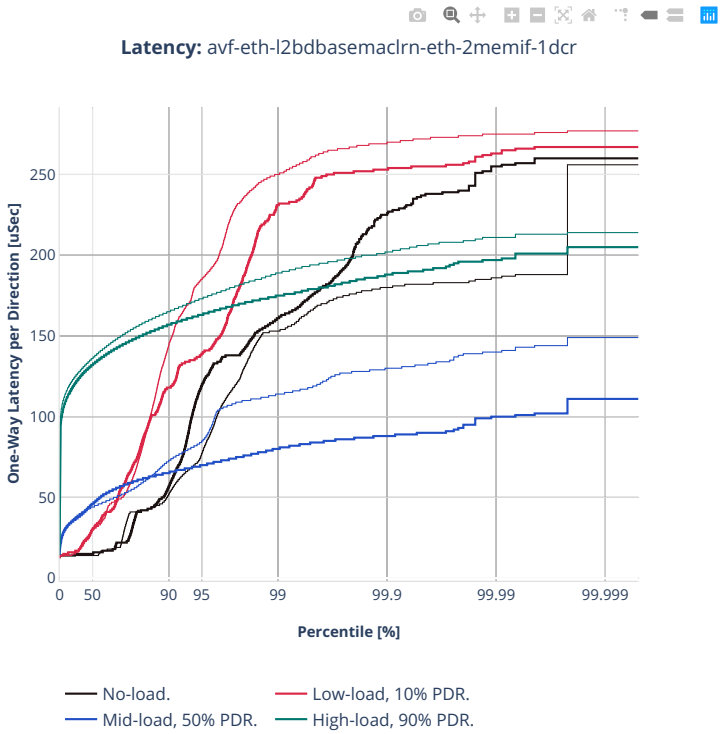


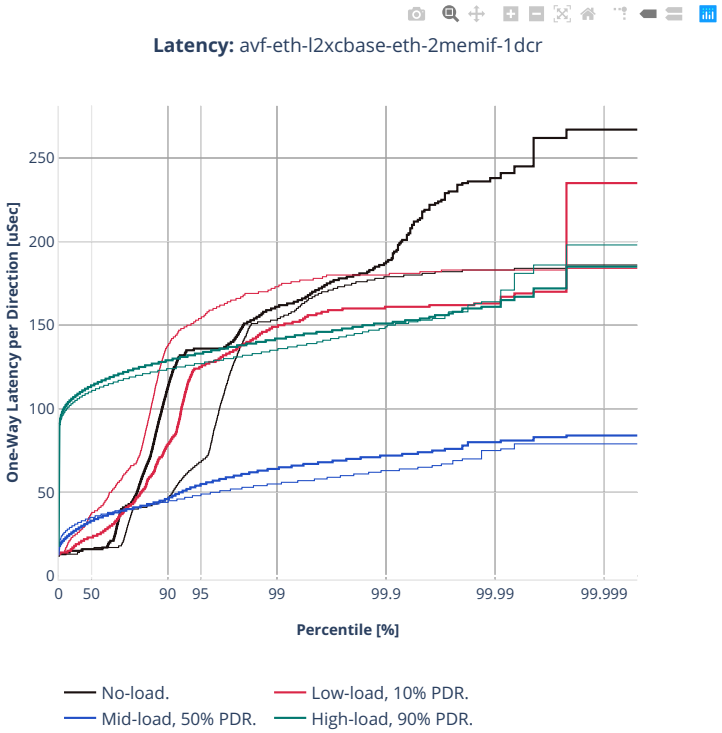

Latency: eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

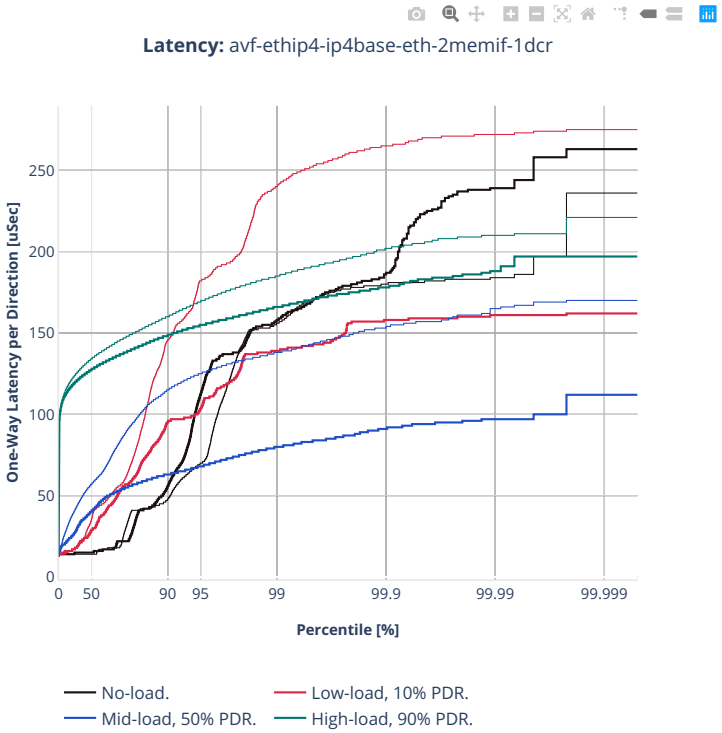

Latency: ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-clx-xxv710

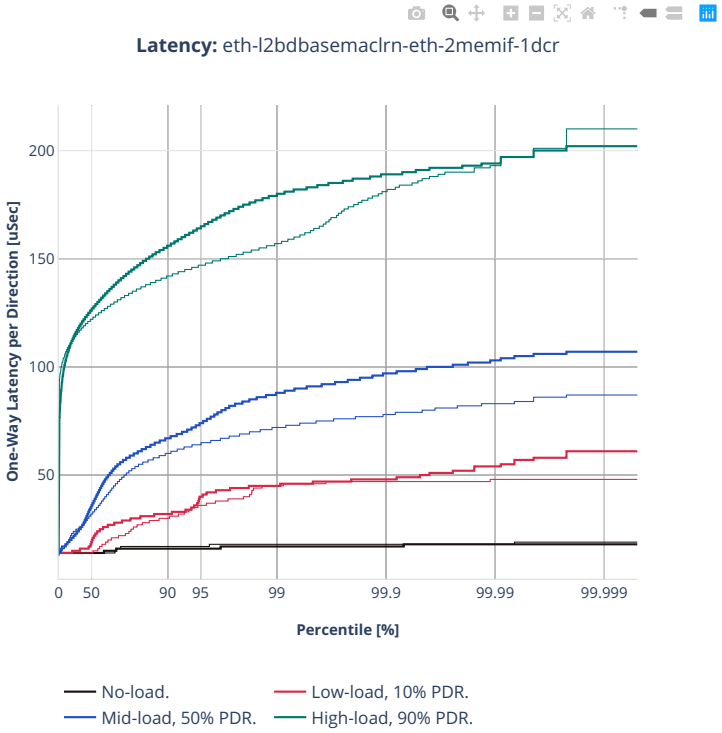
64b-2t1c-memif-base-avf

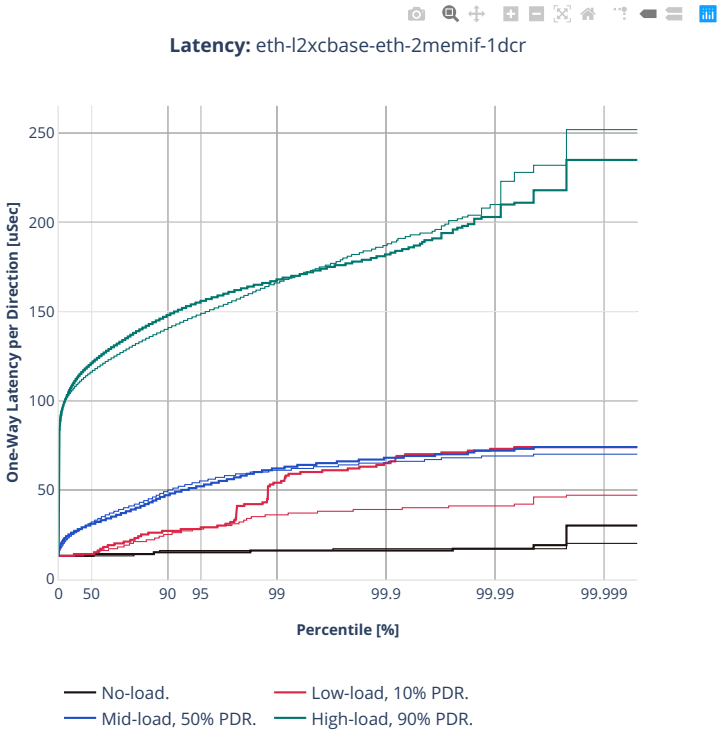


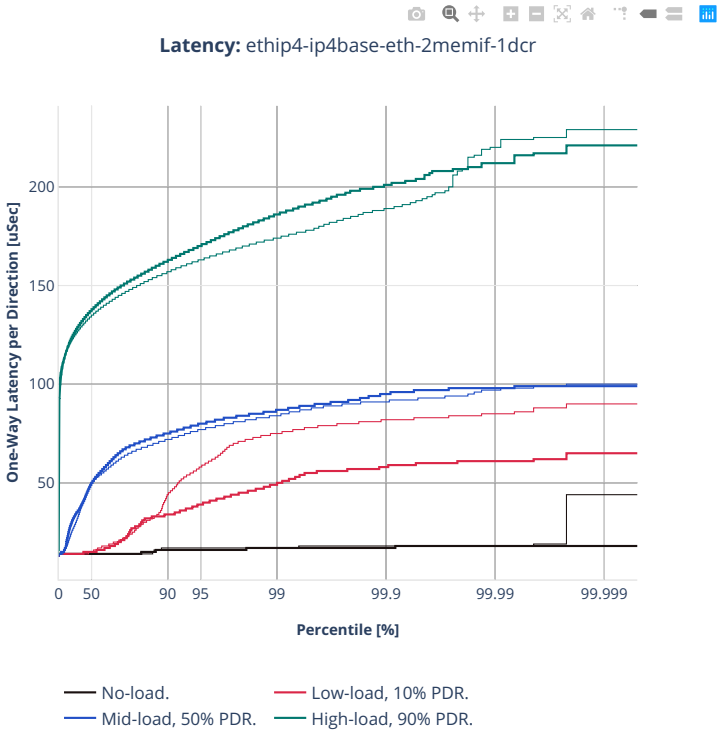

Latency: avf-eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-memif-base-dpdk

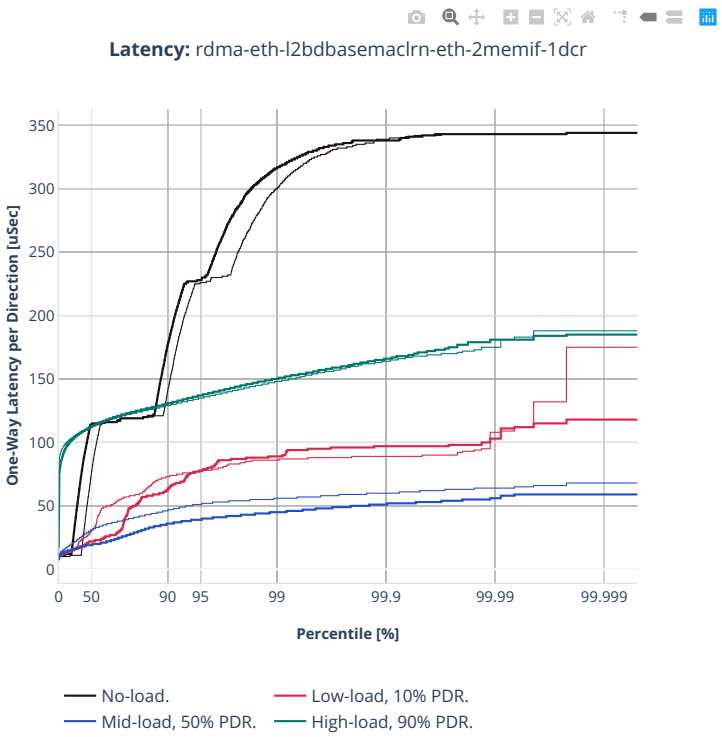


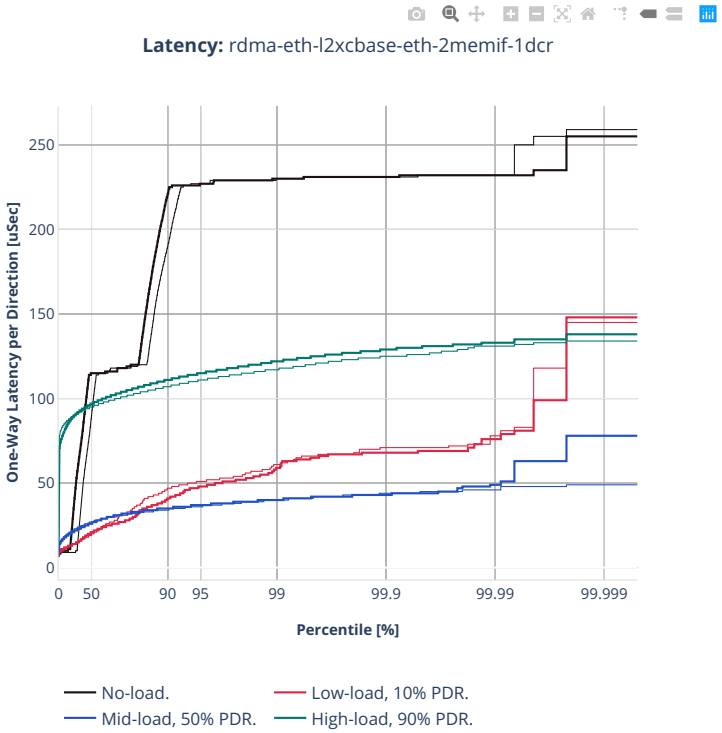

Latency: eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

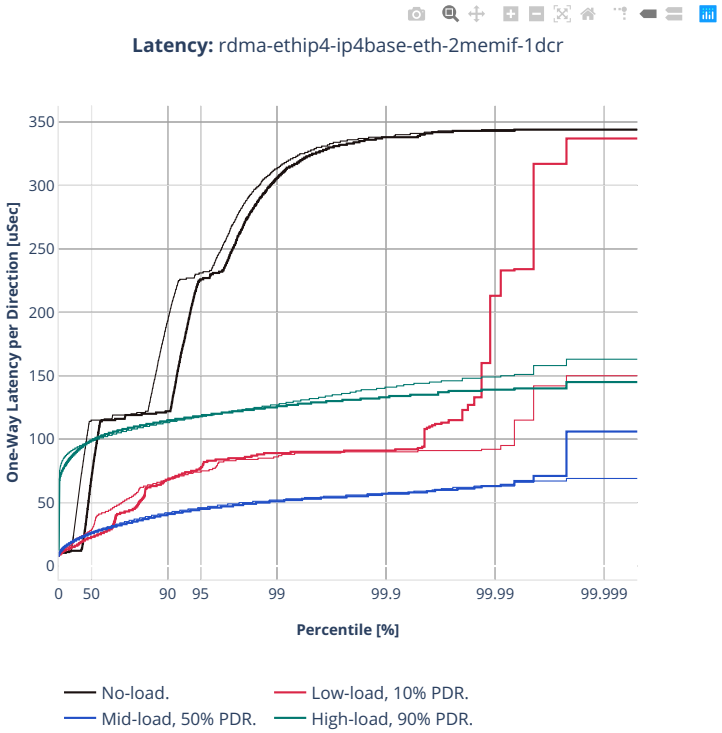

Latency: ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
200
150
100
50
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-clx-cx556a

64b-2t1c-memif-base-rdma

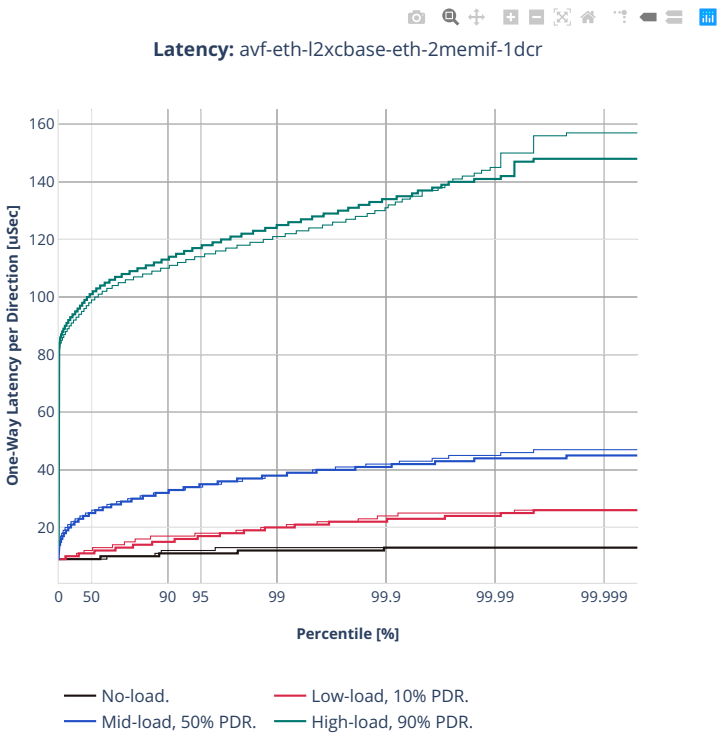


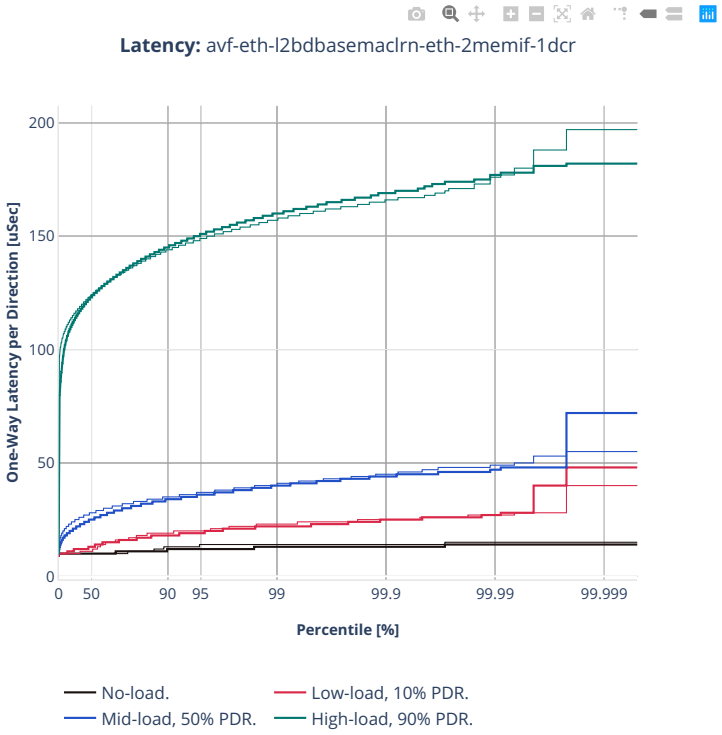

Latency: rdma-eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

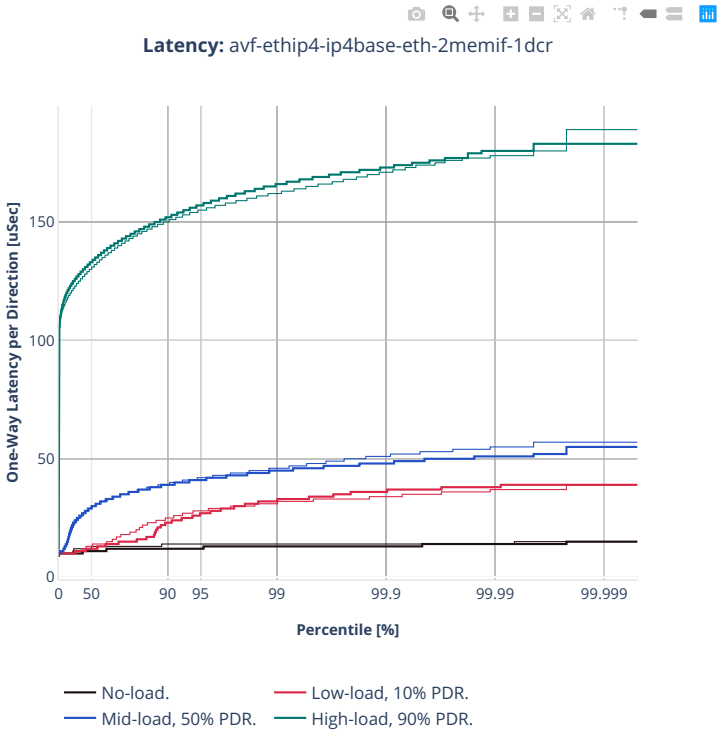

Latency: rdma-ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

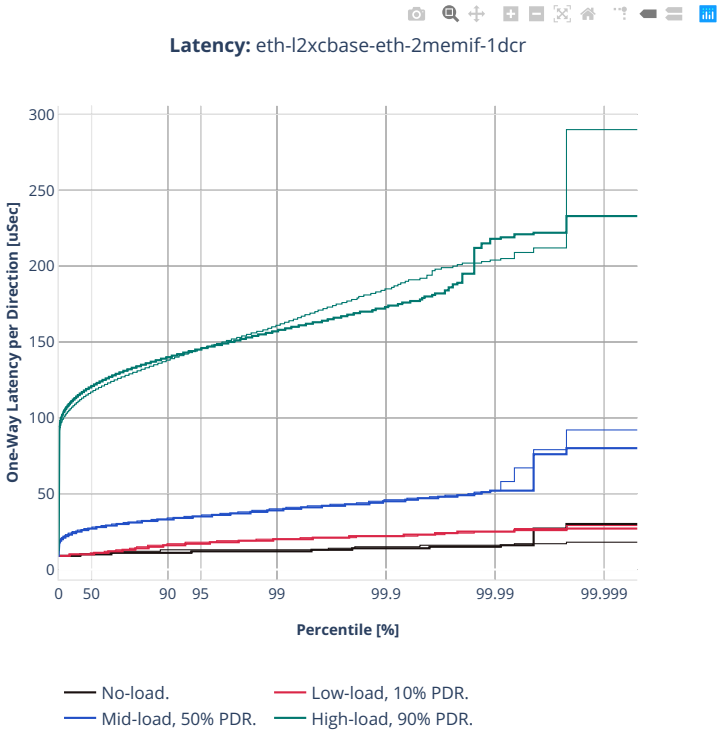
2n-clx-e810cq

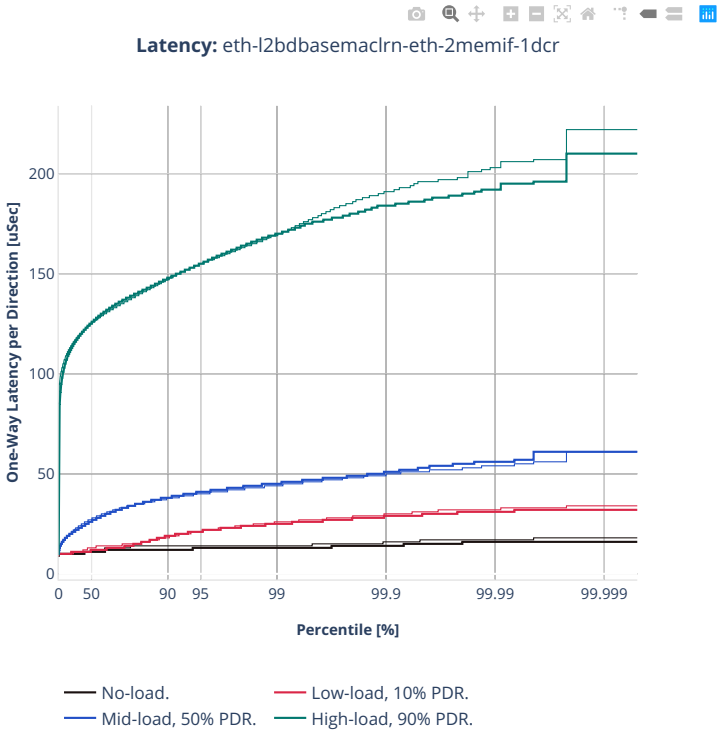
64b-2t1c-memif-base-rdma

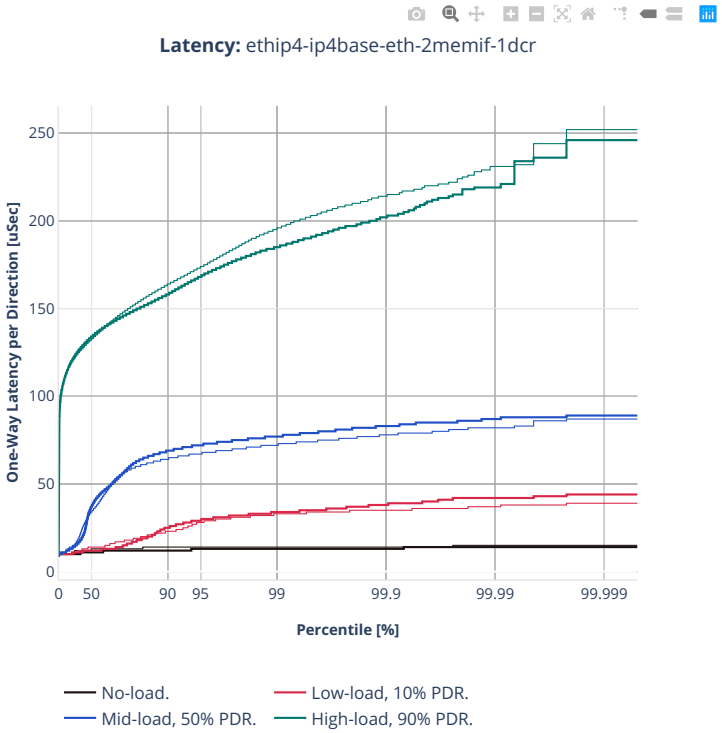



Latency: avf-eth-l2bdbasemaclrn-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

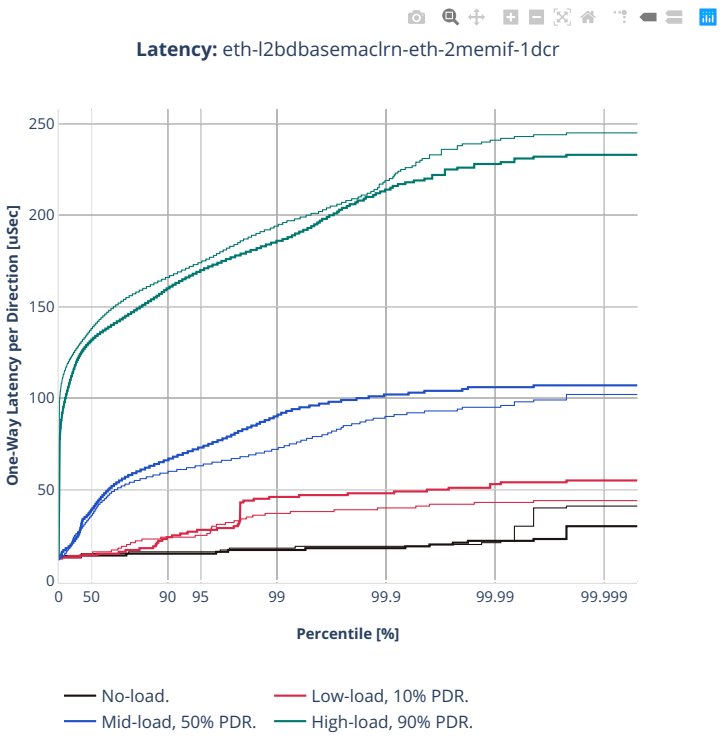

Latency: eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

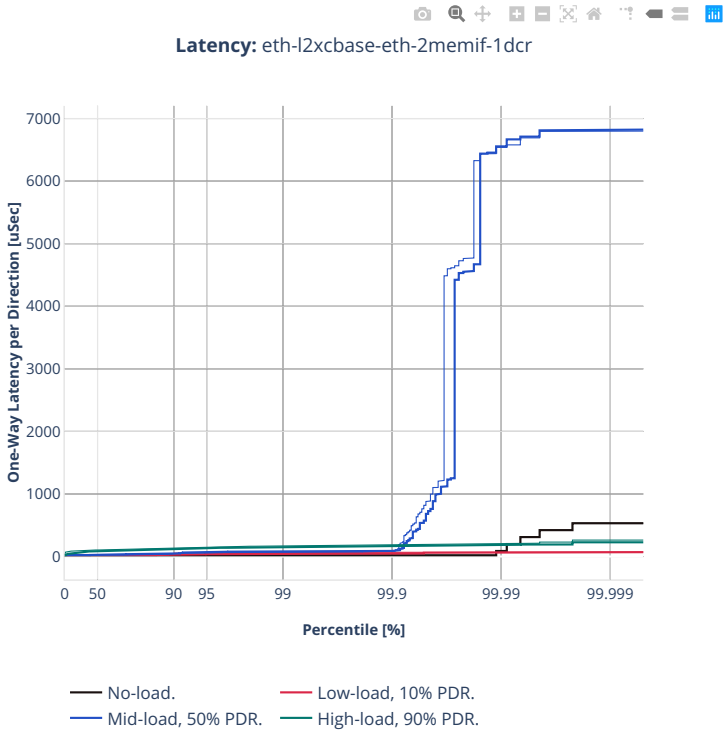

Latency: eth-l2bdbasemaclrn-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

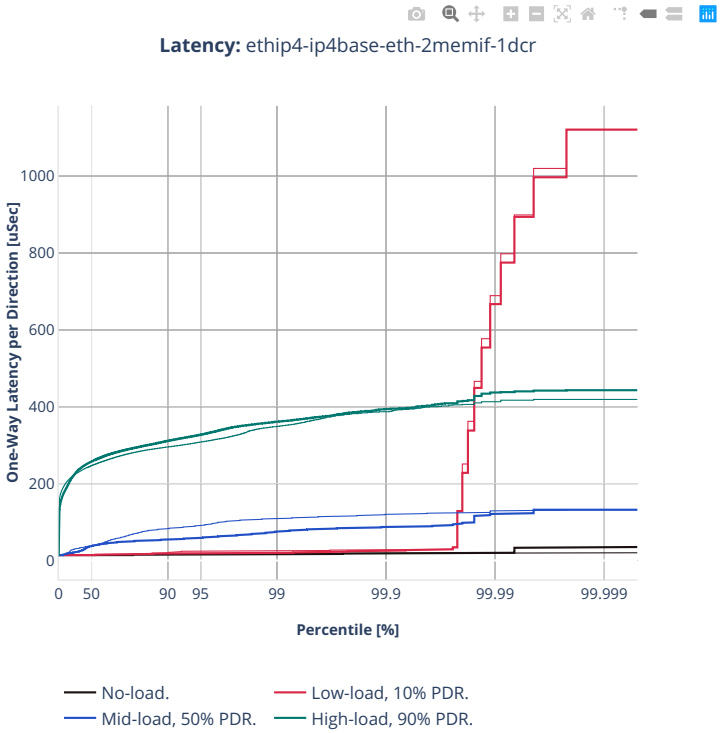

Latency: ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-tx2-xl710

64b-1t1c-memif-base-dpdk

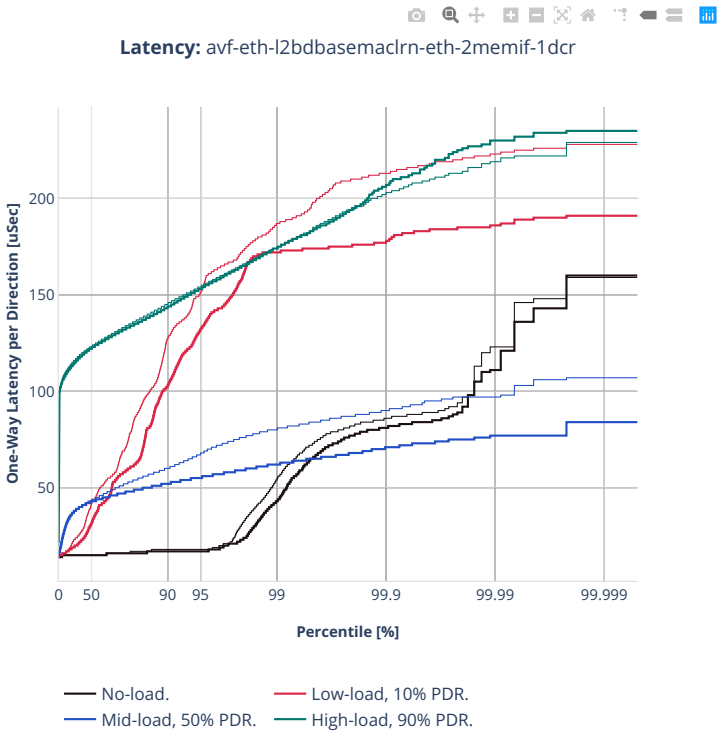


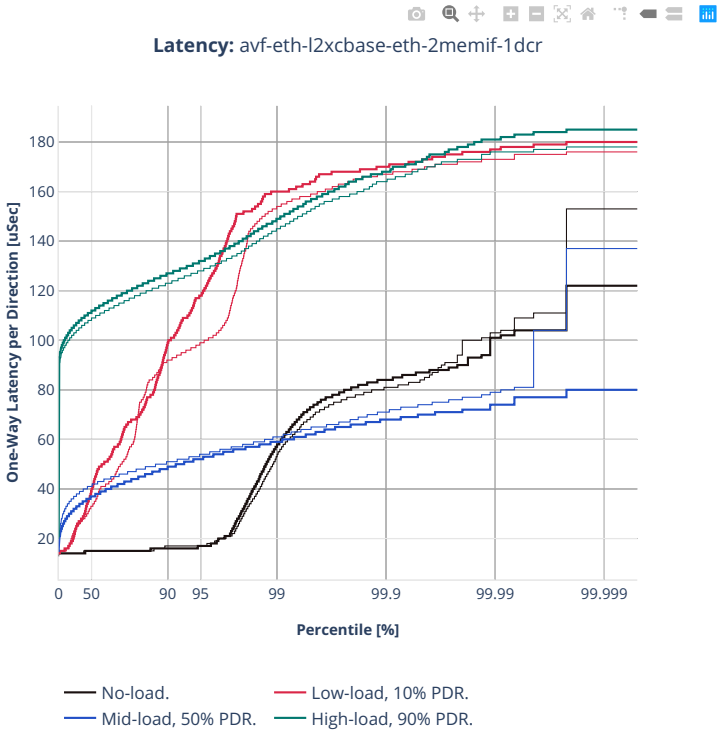

Latency: eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

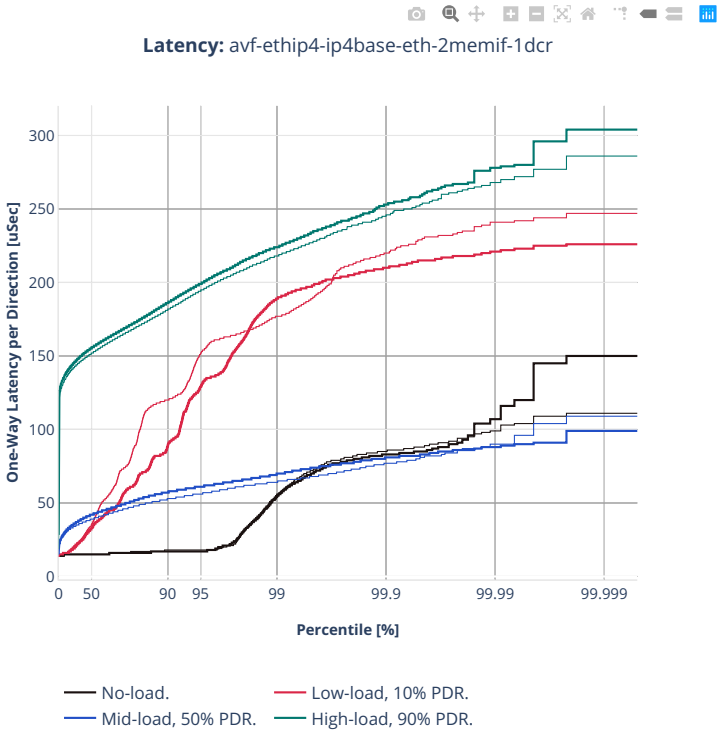

Latency: ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2n-zn2-xxv710

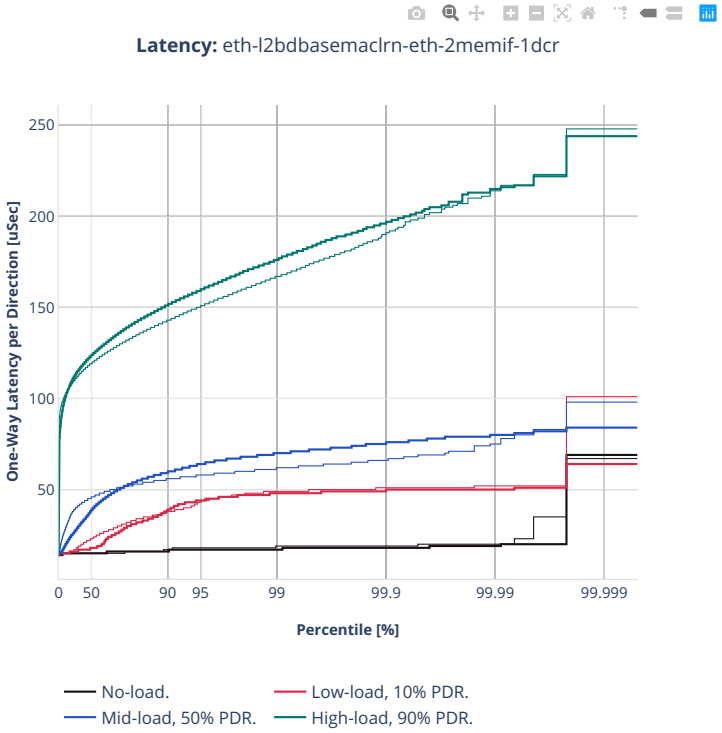
64b-2t1c-memif-base-avf

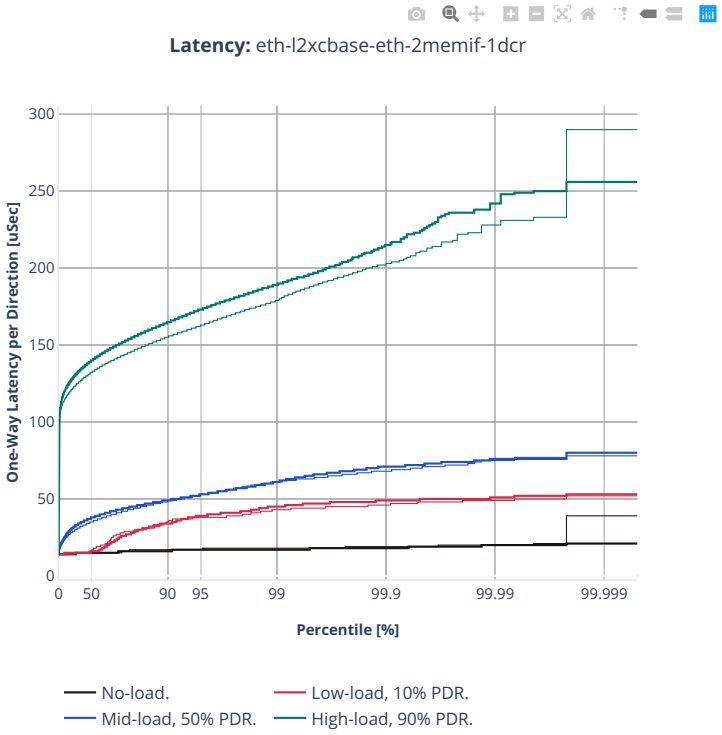


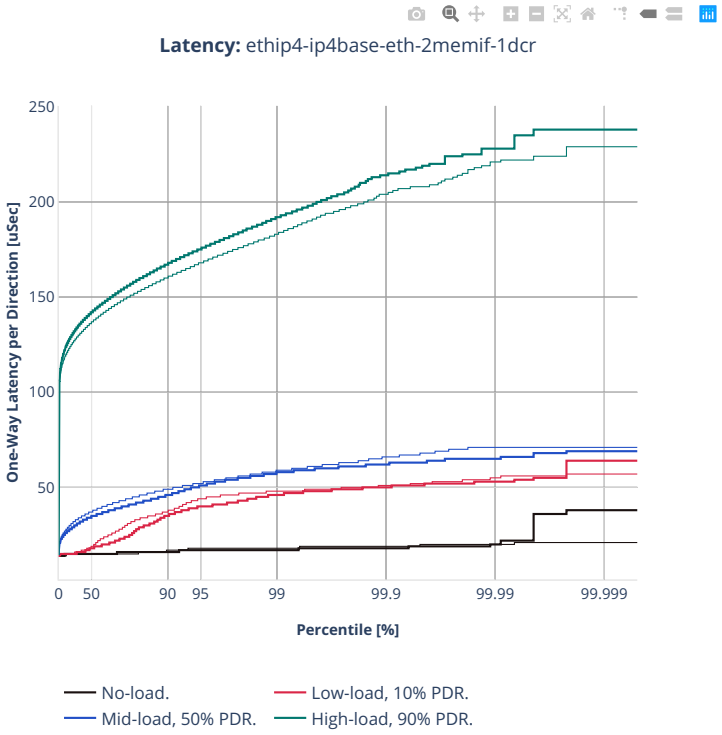

Latency: avf-eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
180
160
140
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: avf-ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

64b-2t1c-memif-base-dpdk




Latency: eth-l2xcbase-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4-ip4base-eth-2memif-1dcr
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

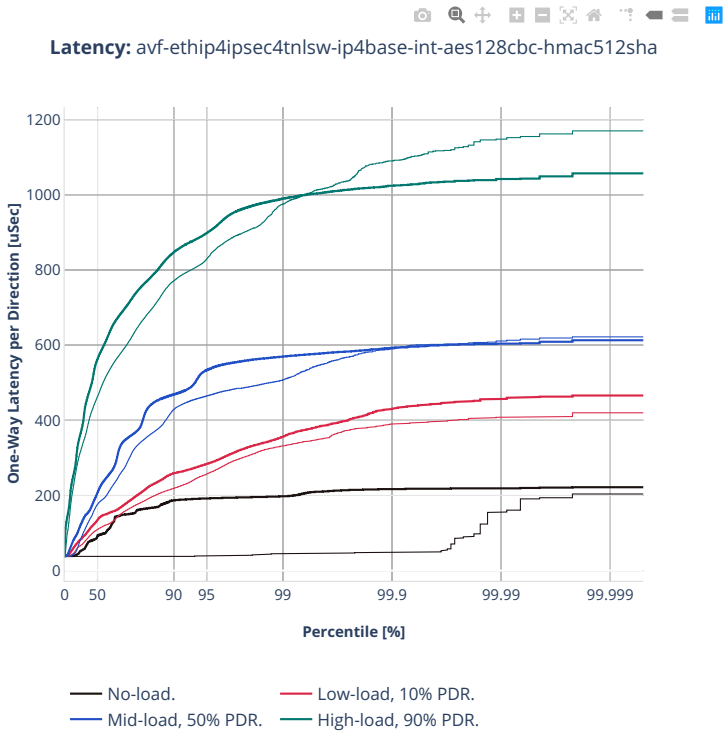
2.5.9 IPSec IPv4 Routing

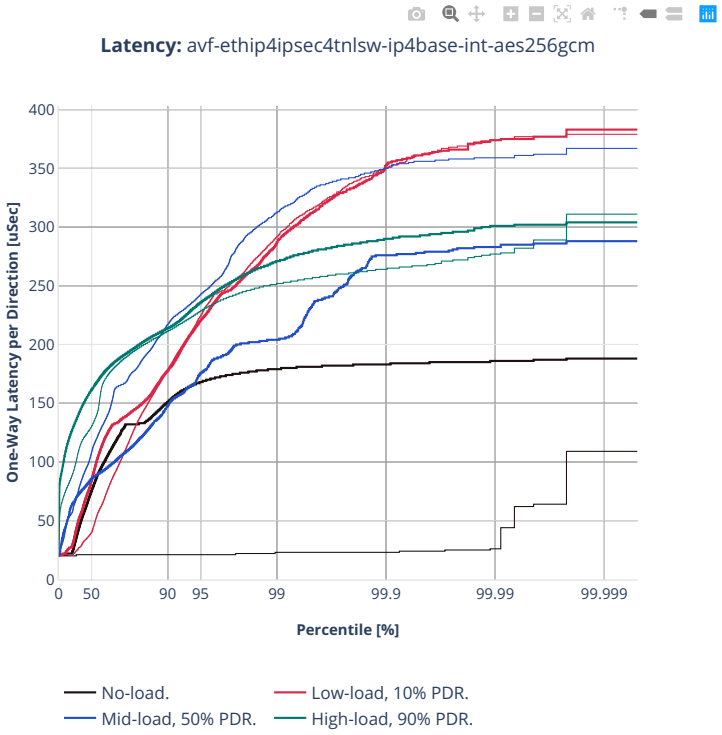
CSIT source code for the test cases used for plots can be found in CSIT git repository[158].

[158] https://git.fd.io/csit/tree/tests/vpp/perf/crypto?h=rls2210

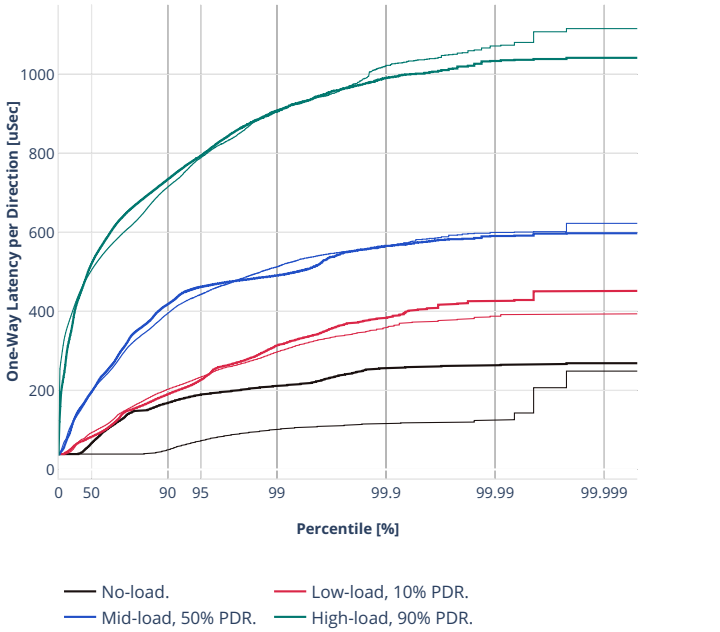
3n-icx-xxv710

1518b-2t1c-ipsec-ip4routing-scale-sw-avf

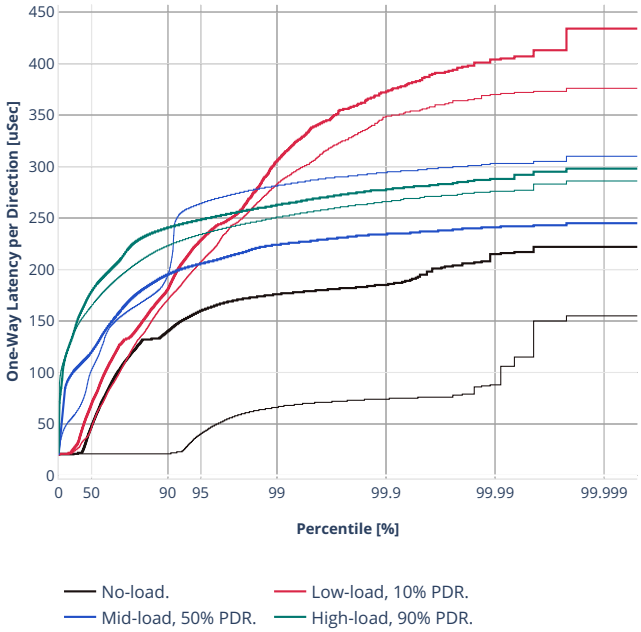



Latency: avf-ethip4ipsec4tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
400
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

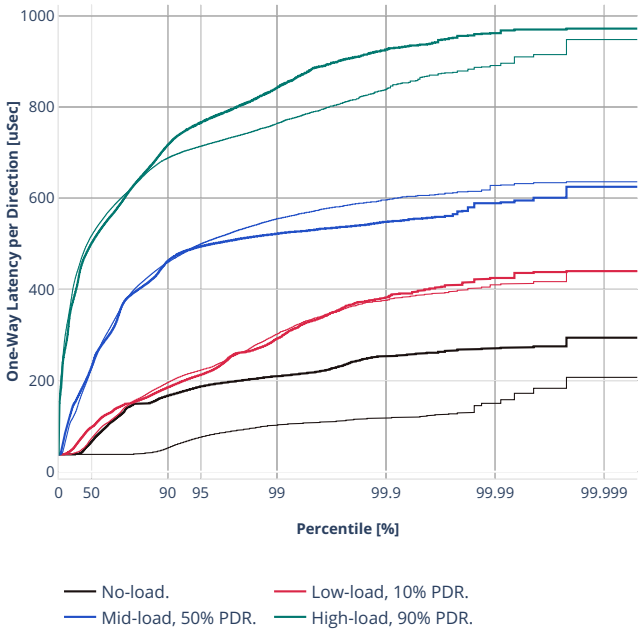
Latency: avf-ethip4ipsec40tnlsw-ip4base-int-aes128cbc-hmac512sha



Latency: avf-ethip4ipsec40tnlsw-ip4base-int-aes256gcm

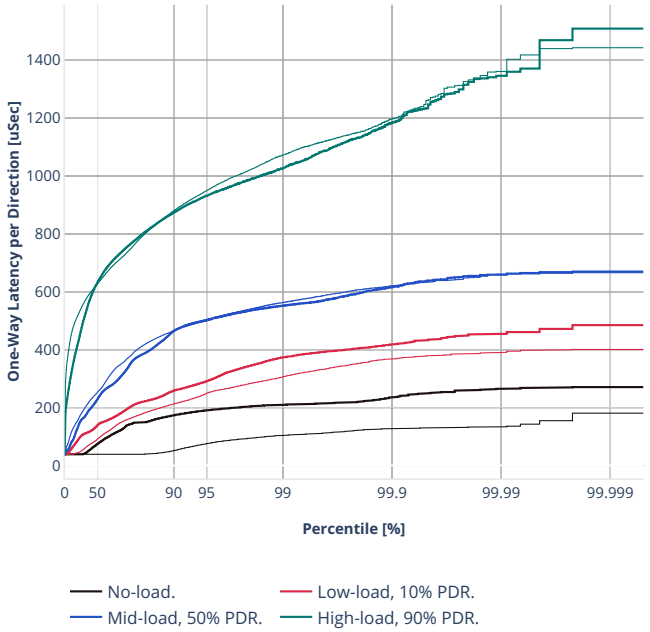


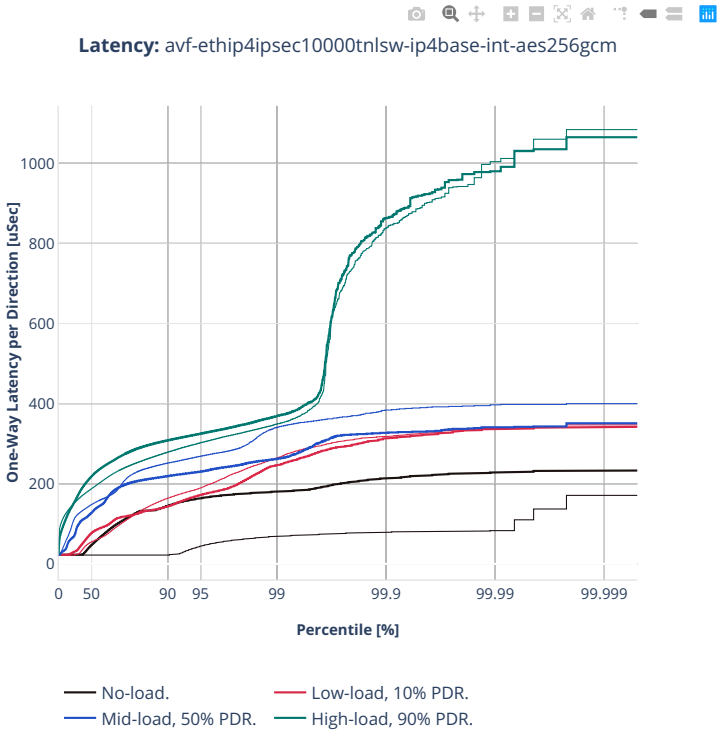
Latency: avf-ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha



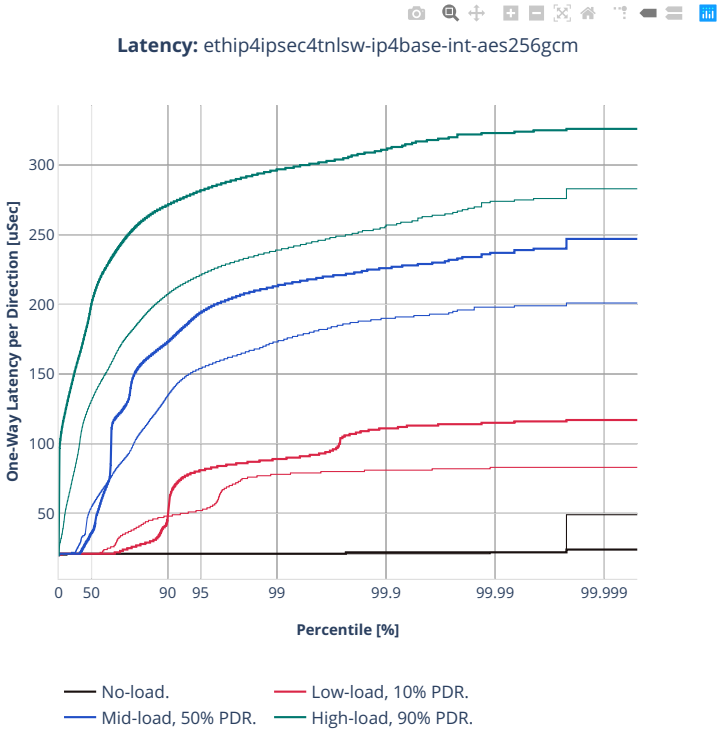

Latency: avf-ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
400
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

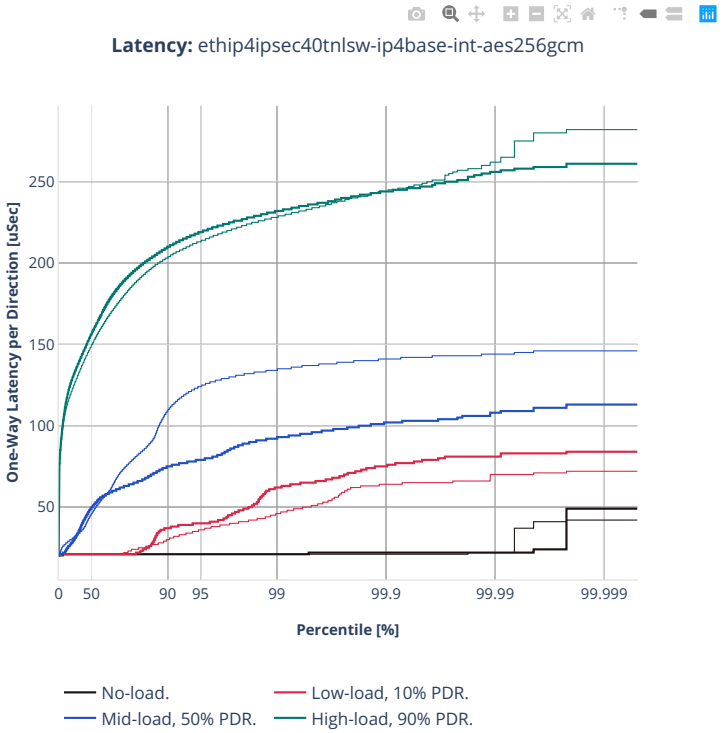
Latency: avf-ethip4ipsec10000tnlsw-ip4base-int-aes128cbc-hmac512sha

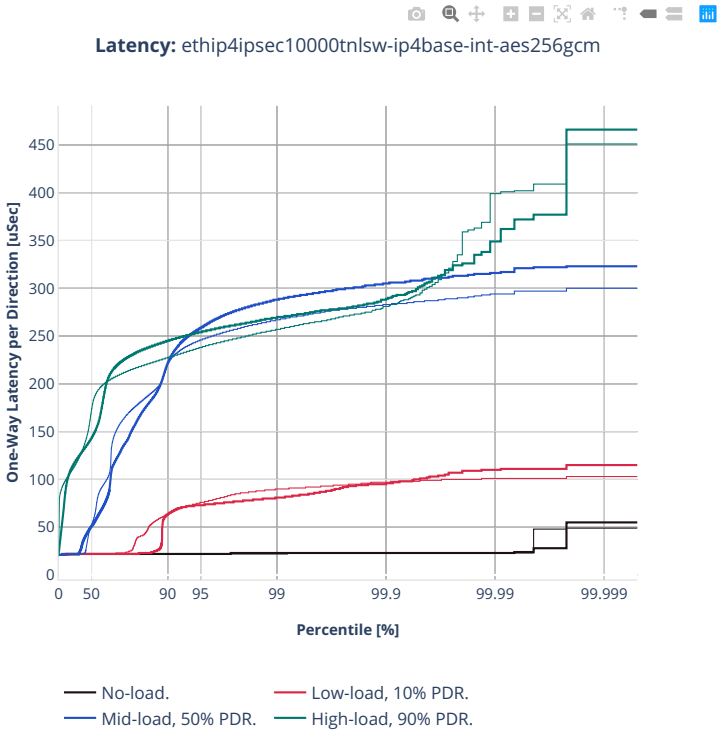



Latency: avf-ethip4ipsec10000tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

1518b-2t1c-ipsec-ip4routing-scale-sw-dpdk

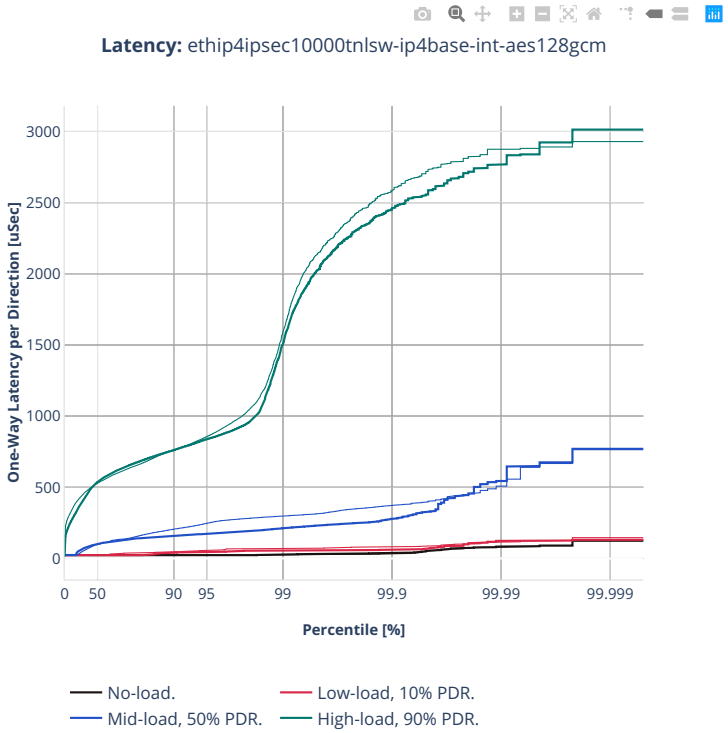



Latency: ethip4ipsec40tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
250
200
150
100
50
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

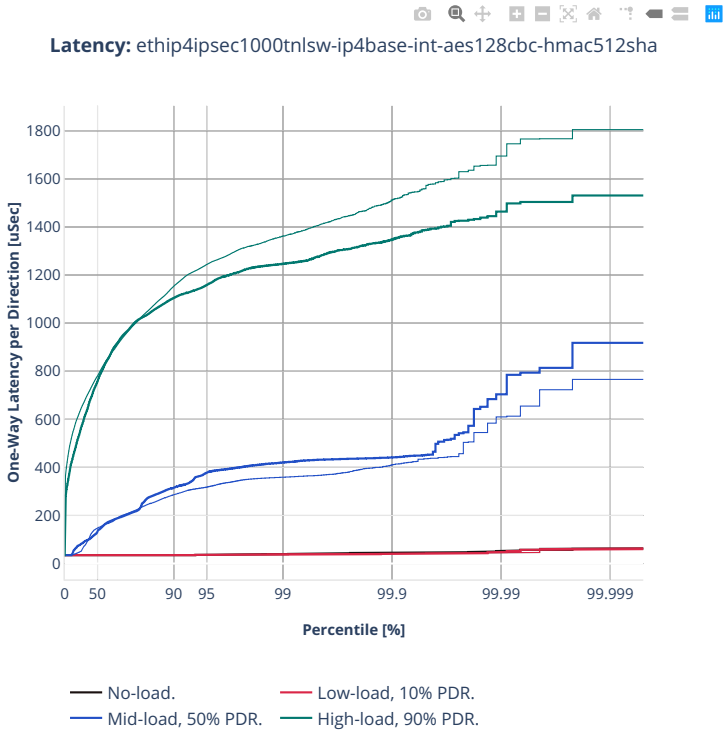

Latency: ethip4ipsec10000tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
450
400
350
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

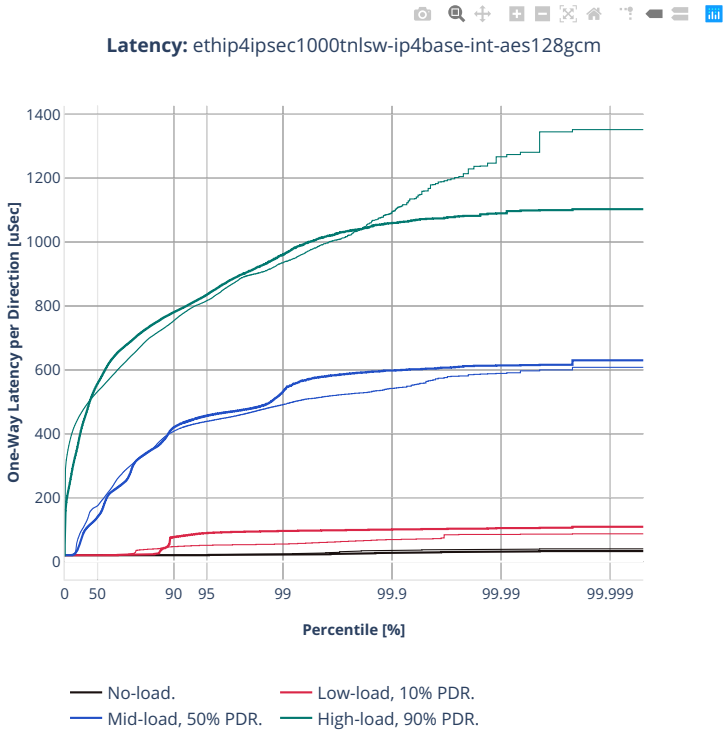
3n-alt-xl710

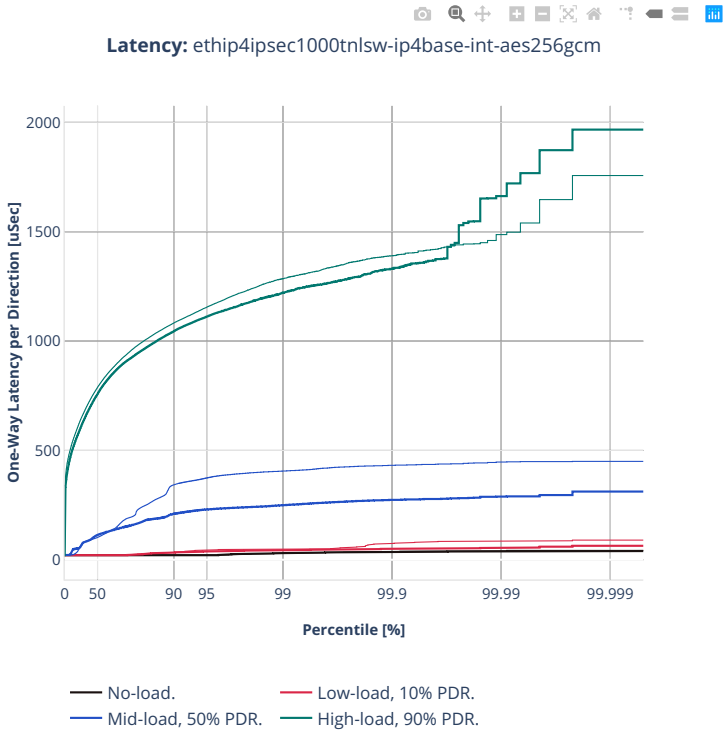
1518b-1t1c-ipsec-ip4routing-base-scale


Latency: ethip4ipsec10000tnlsw-ip4base-int-aes128gcm
One-Way Latency per Direction [uSec]
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

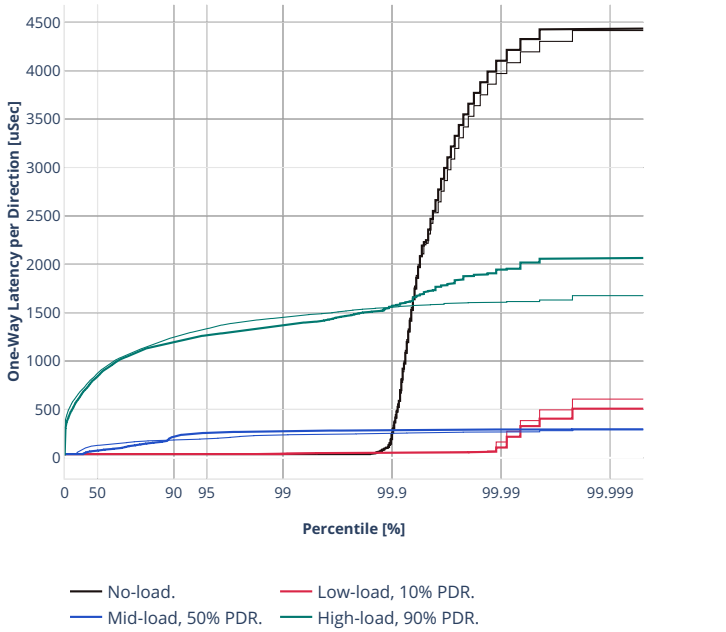

Latency: ethip4ipsec10000tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
3000
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

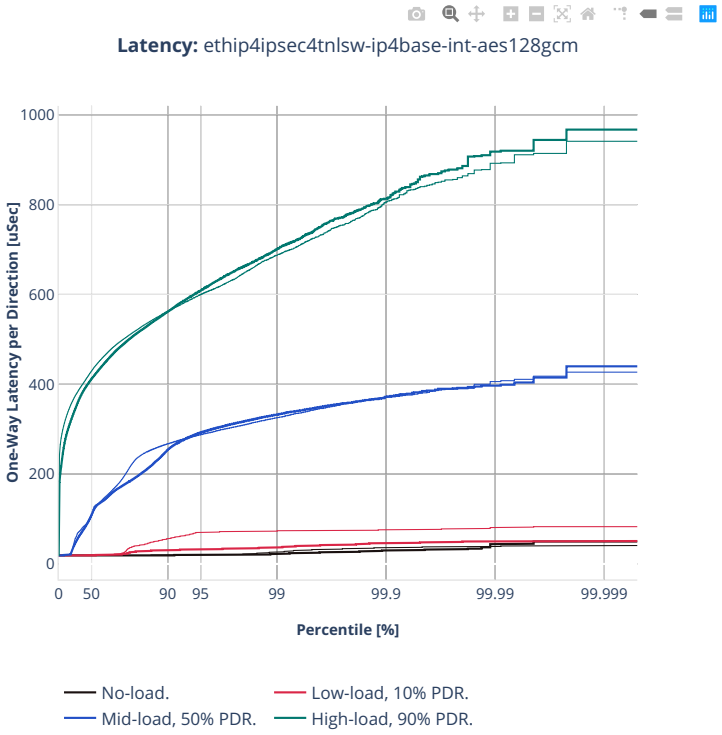

Latency: ethip4ipsec1000tnlsw-ip4base-int-aes128cbc-hmac512sha
One-Way Latency per Direction [uSec]
1800
1600
1400
1200
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

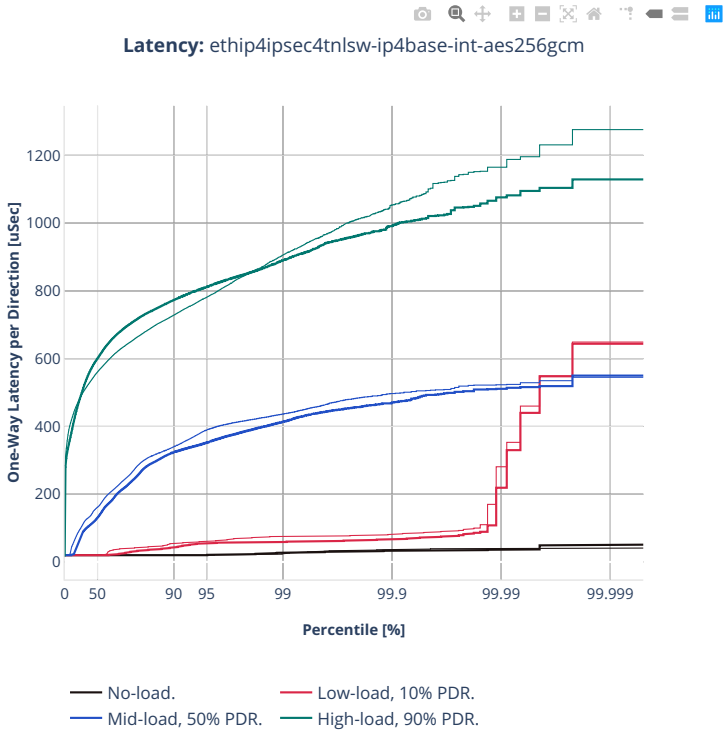

Latency: ethip4ipsec1000tnlsw-ip4base-int-aes128gcm
One-Way Latency per Direction [uSec]
1400
1200
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

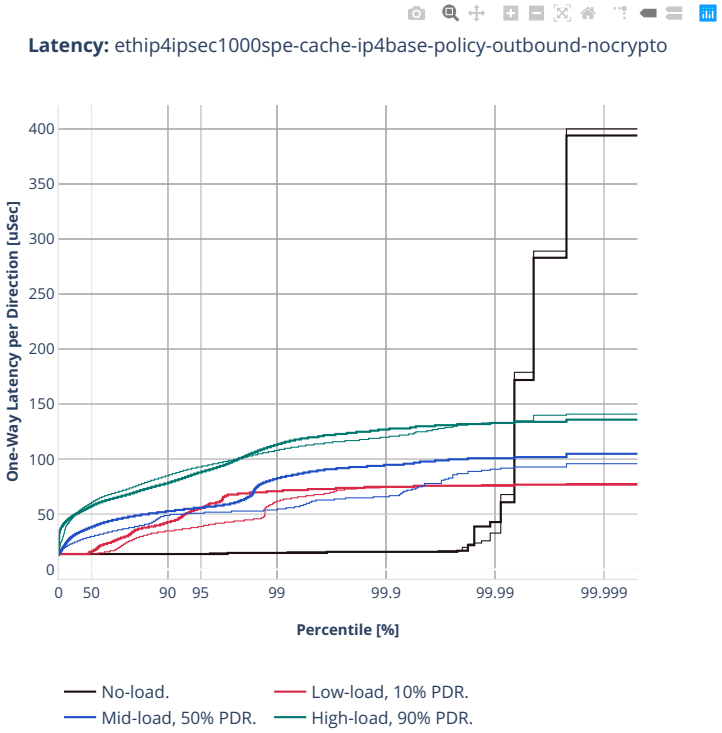
Latency: ethip4ipsec4tnlsw-ip4base-int-aes128cbc-hmac512sha

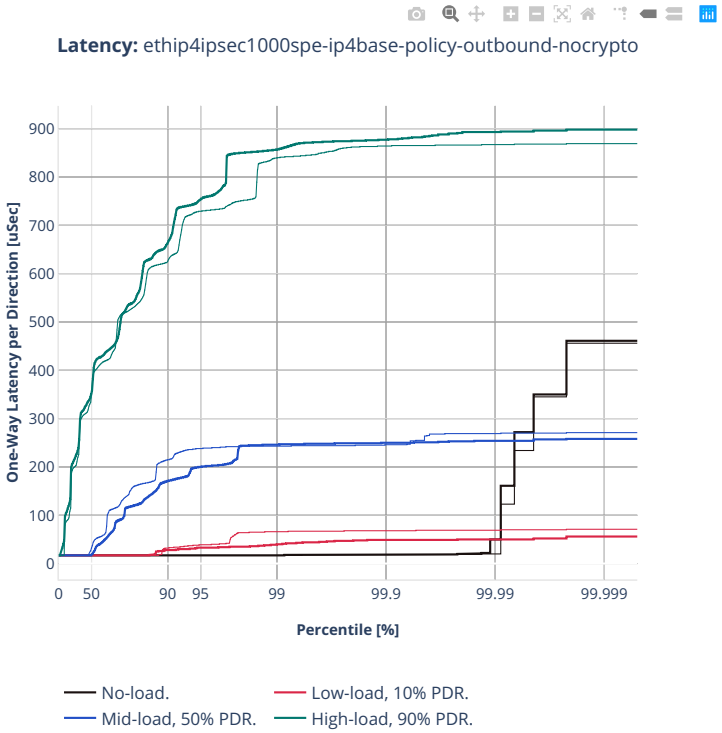



Latency: ethip4ipsec4tnlsw-ip4base-int-aes128gcm
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4ipsec4tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
1200
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

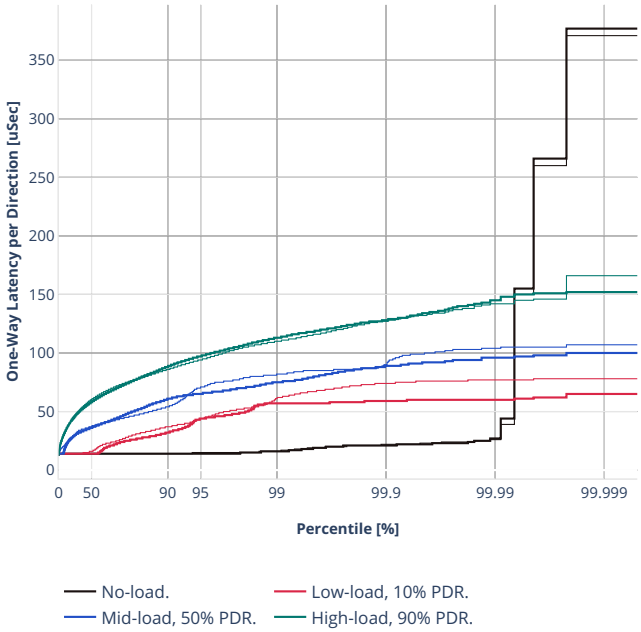
64b-1t1c-ipsec-ip4routing-base-scale

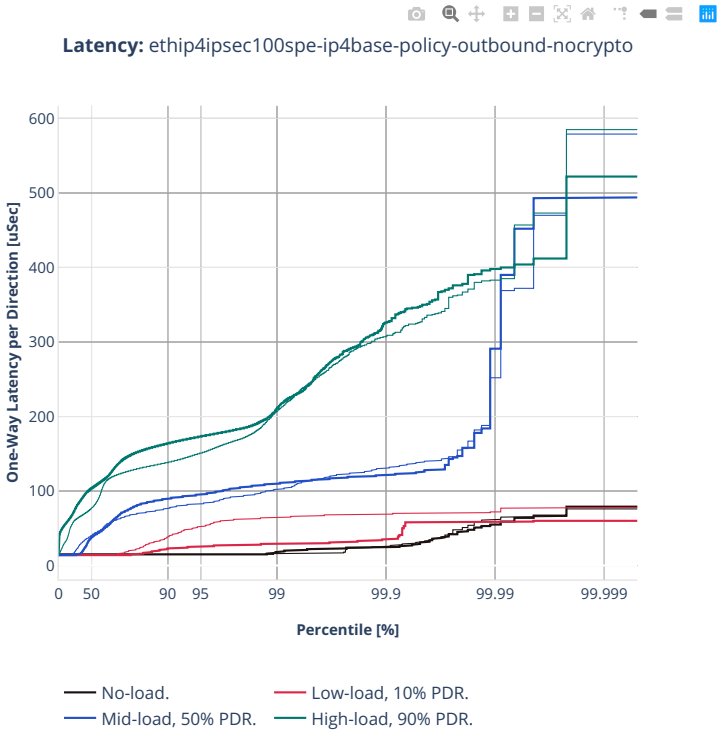


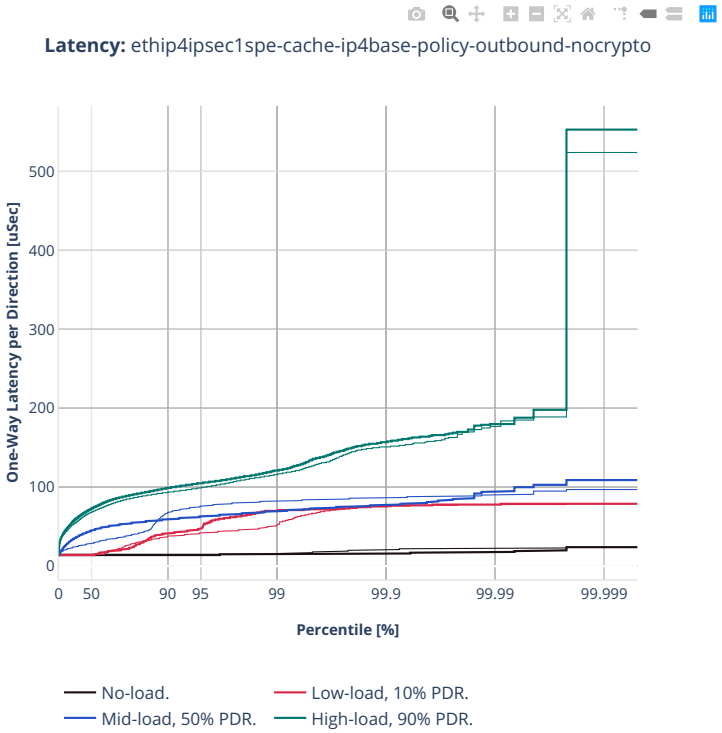

Latency: ethip4ipsec1000spe-ip4base-policy-outbound-nocrypto
One-Way Latency per Direction [uSec]
900
800
700
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

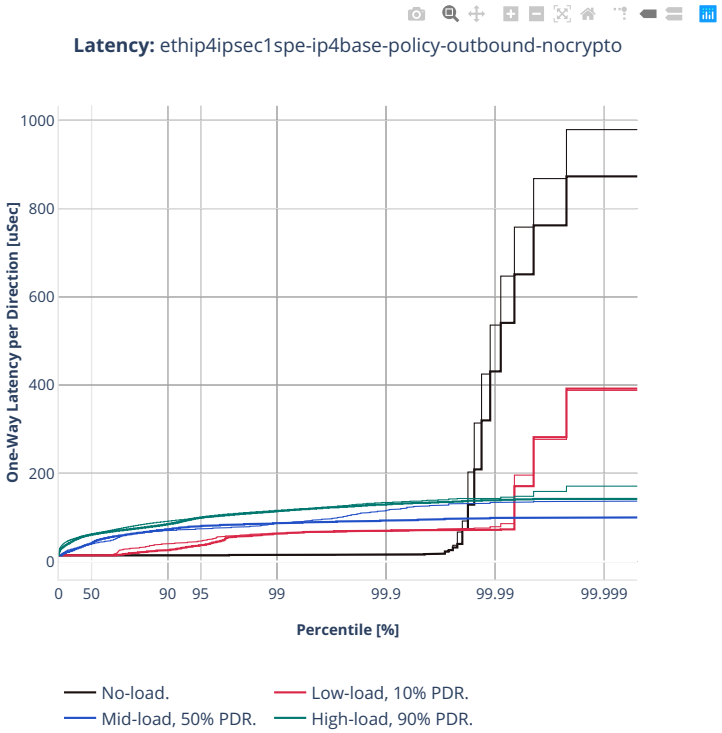

Latency: ethip4ipsec1000tnlsw-ip4base-policy-aes256gcm
One-Way Latency per Direction [uSec]
2500
2000
1500
1000
500
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

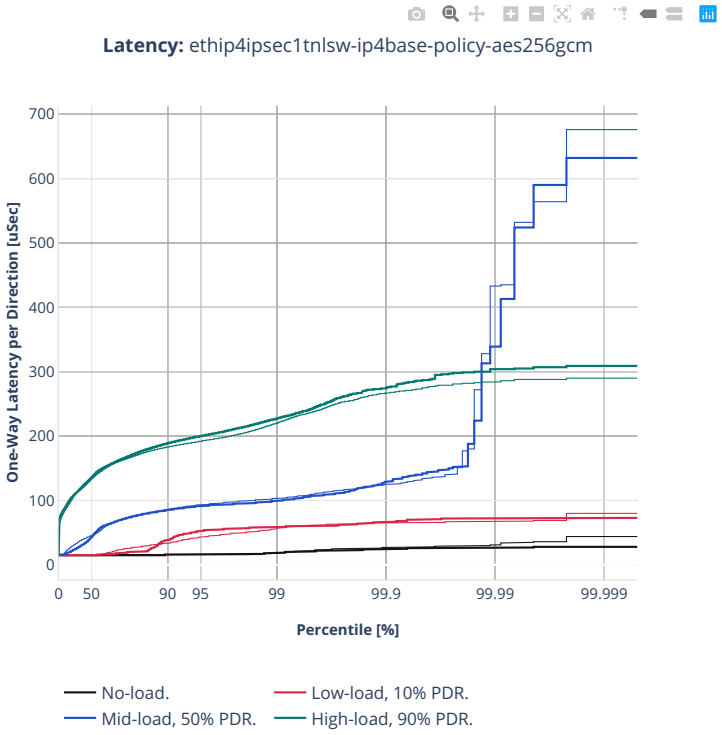
Latency: ethip4ipsec100spe-cache-ip4base-policy-outbound-nocrypto

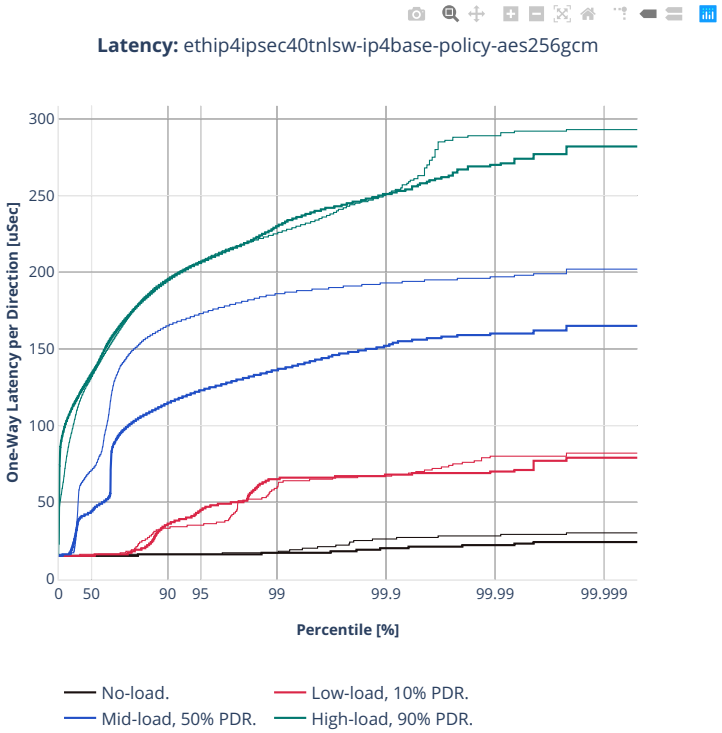



Latency: ethip4ipsec100spe-ip4base-policy-outbound-nocrypto
One-Way Latency per Direction [uSec]
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4ipsec1spe-cache-ip4base-policy-outbound-nocrypto
One-Way Latency per Direction [uSec]
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

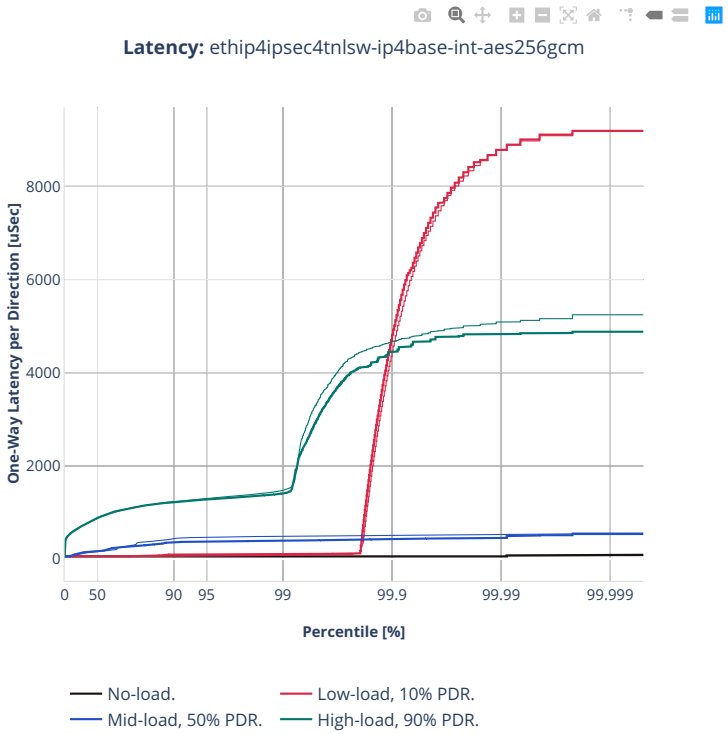

Latency: ethip4ipsec1spe-ip4base-policy-outbound-nocrypto
One-Way Latency per Direction [uSec]
1000
800
600
400
200
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

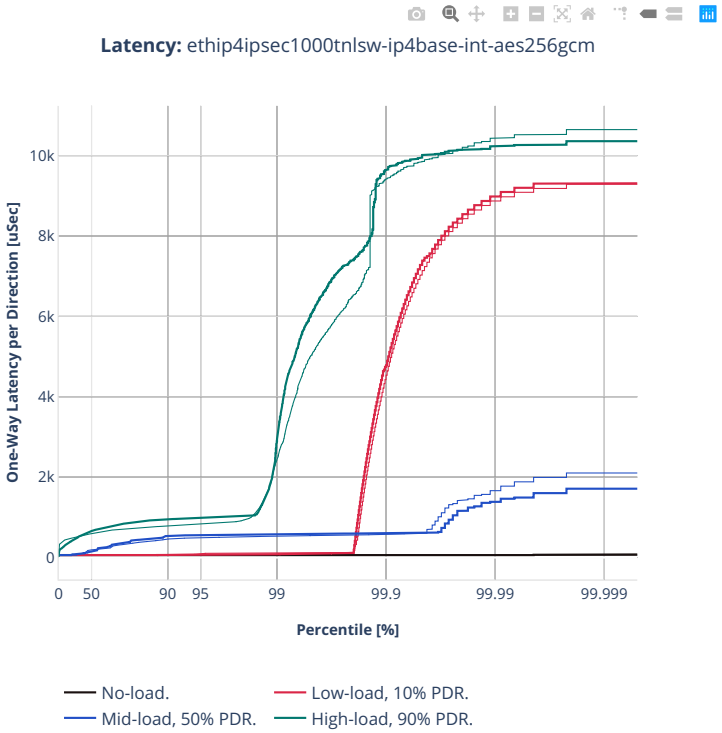

Latency: ethip4ipsec1tnlsw-ip4base-policy-aes256gcm
One-Way Latency per Direction [uSec]
700
600
500
400
300
200
100
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

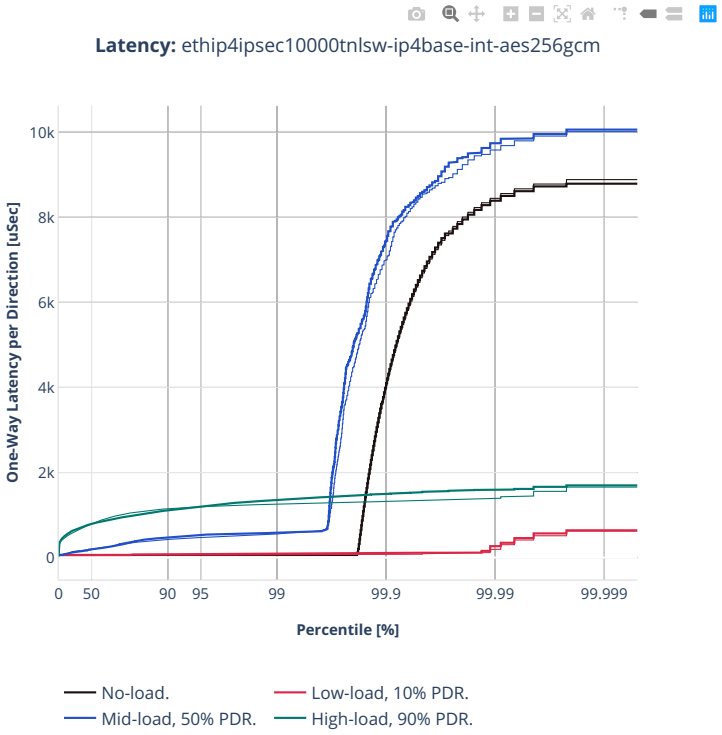

Latency: ethip4ipsec40tnlsw-ip4base-policy-aes256gcm
One-Way Latency per Direction [uSec]
300
250
200
150
100
50
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3n-tsh-x520

1518b-1t1c-ipsec-ip4routing-base-scale-sw-ixgbe




Latency: ethip4ipsec1000tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
10k
8k
6k
4k
2k
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.


Latency: ethip4ipsec10000tnlsw-ip4base-int-aes256gcm
One-Way Latency per Direction [uSec]
10k
8k
6k
4k
2k
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

2.6 Soak Tests

Long duration (30 minutes per test) soak tests are executed using *PLRsearch* (page 26) algorithm. As the tests take long time, only 12 test cases were executed, two runs each.

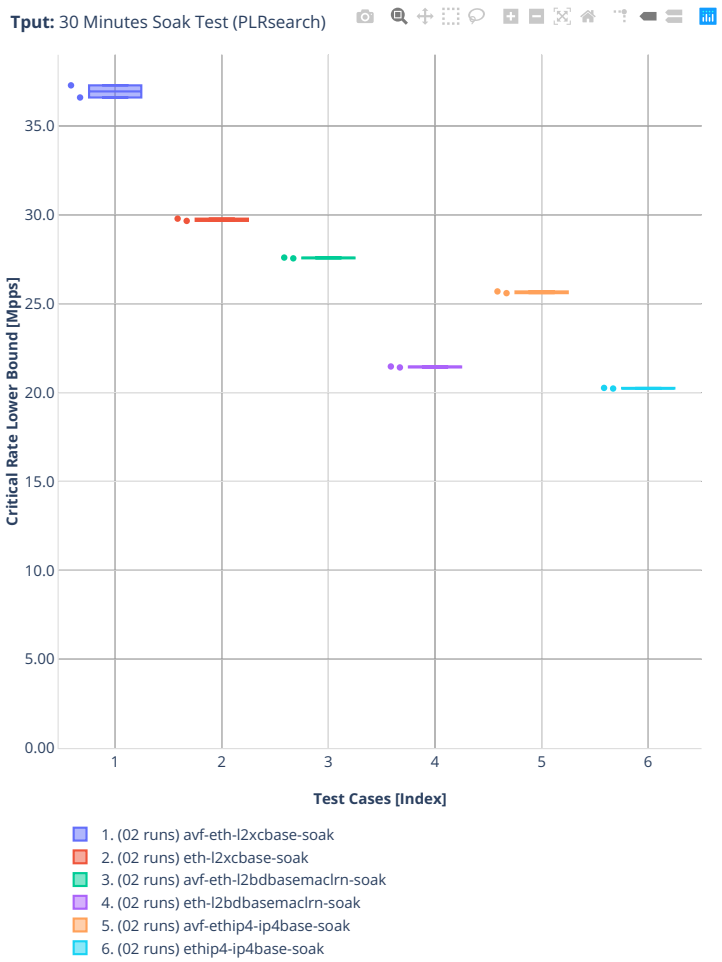
Additional information about graph data:

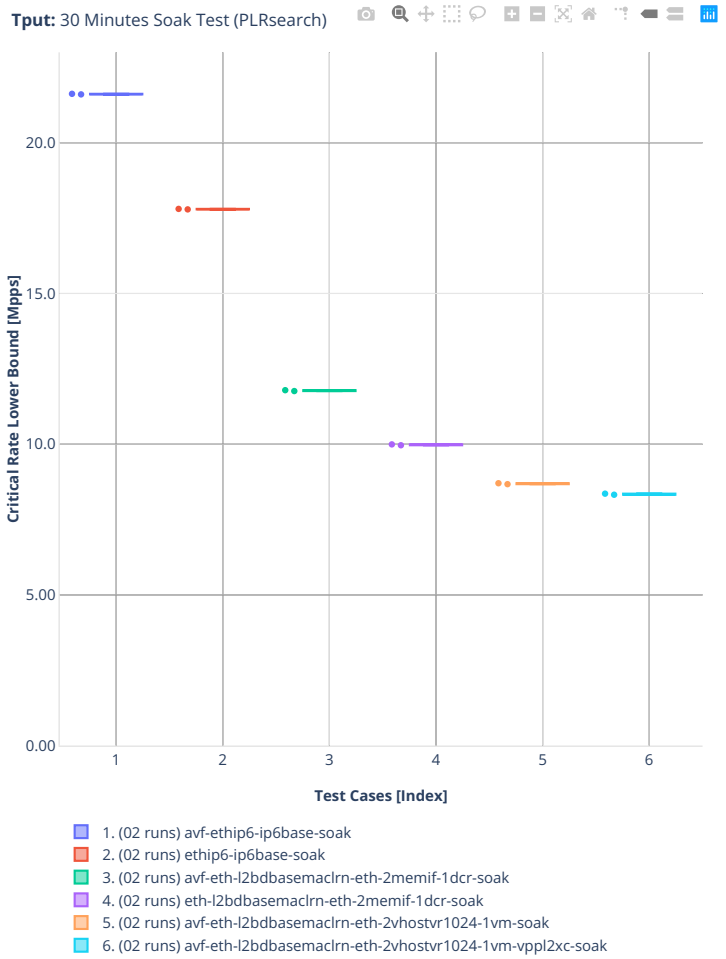
1. **Graph Title**: describes type of tests and soak test duration.
2. **X-axis Labels**: indices of test suites.
3. **Y-axis Labels**: estimated lower bounds for critical rate value in [Mpps].
4. **Graph Legend**: list of X-axis indices with CSIT test cases.
5. **Hover Information**: in general lists minimum, first quartile, median, third quartile, and maximum. As only two samples are used, minimum and maximum are not distinguished from quartiles.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[159], build logs from FD.io vpp performance job 2n-clx[160] with RF result files csit-vpp-perf-2210-*.zip archived here.

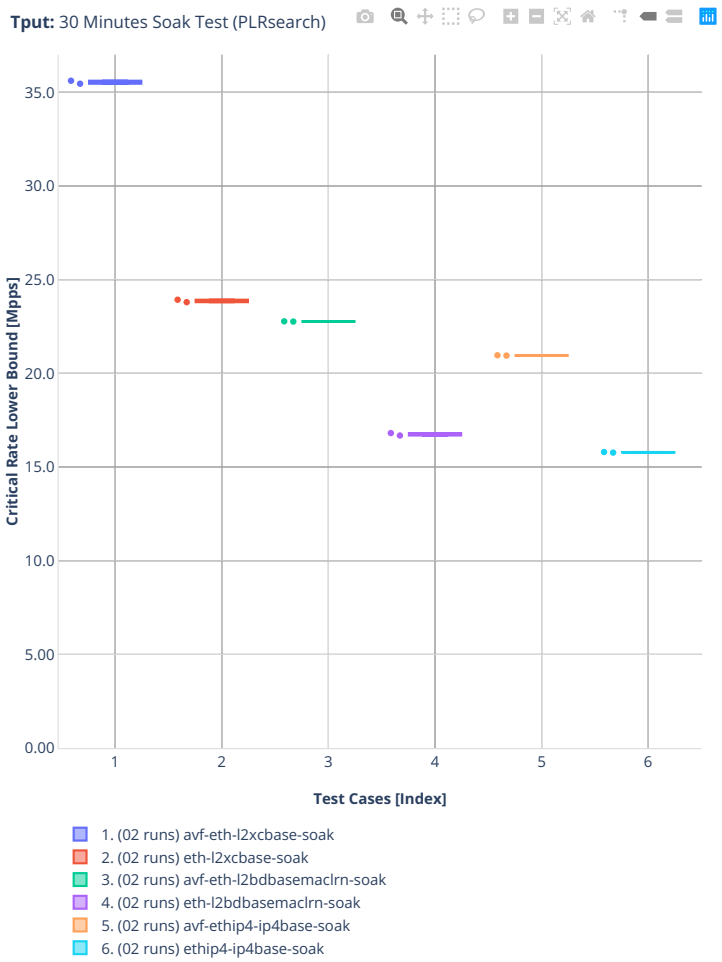
[159] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[160] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx

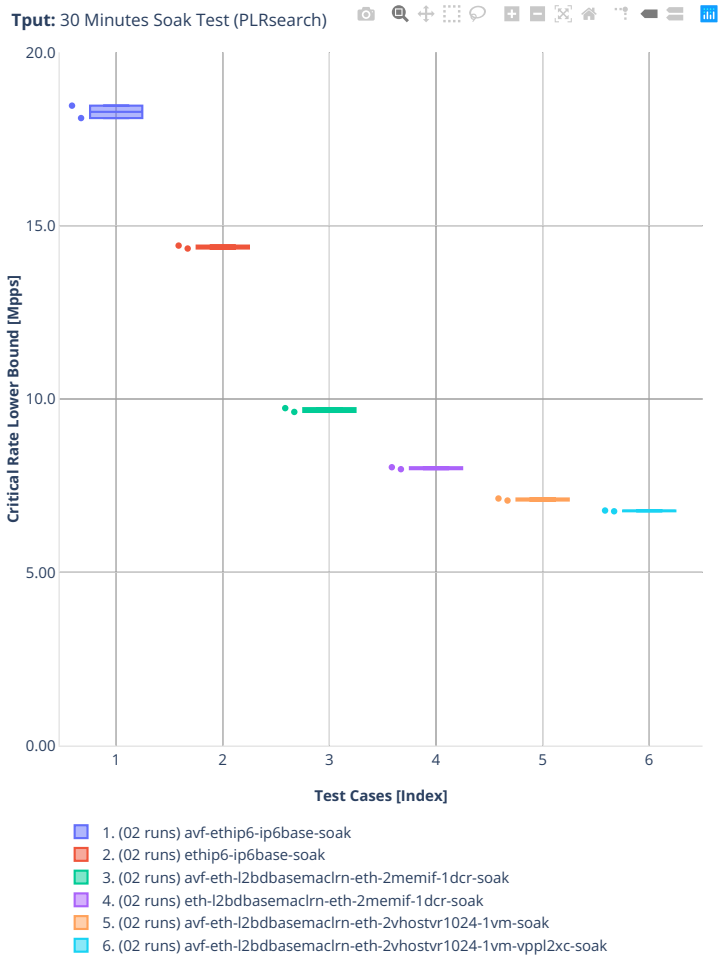
2.6.1 2n-icx


Tput: 30 Minutes Soak Test (PLRsearch)
Critical Rate Lower Bound [Mpps]
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1. (02 runs) avf-eth-l2xcbase-soak
2. (02 runs) eth-l2xcbase-soak
3. (02 runs) avf-eth-l2bdbasemaclrn-soak
4. (02 runs) eth-l2bdbasemaclrn-soak
5. (02 runs) avf-ethip4-ip4base-soak
6. (02 runs) ethip4-ip4base-soak


Tput: 30 Minutes Soak Test (PLRsearch)
Critical Rate Lower Bound [Mpps]
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1. (02 runs) avf-ethip6-ip6base-soak
2. (02 runs) ethip6-ip6base-soak
3. (02 runs) avf-eth-l2bdbasemaclrn-eth-2memif-1dcr-soak
4. (02 runs) eth-l2bdbasemaclrn-eth-2memif-1dcr-soak
5. (02 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-soak
6. (02 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc-soak

2.6.2 2n-clx


Tput: 30 Minutes Soak Test (PLRsearch)
Critical Rate Lower Bound [Mpps]
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
5
6
Test Cases [Index]
1. (02 runs) avf-eth-l2xcbase-soak
2. (02 runs) eth-l2xcbase-soak
3. (02 runs) avf-eth-l2bdbasemaclrn-soak
4. (02 runs) eth-l2bdbasemaclrn-soak
5. (02 runs) avf-ethip4-ip4base-soak
6. (02 runs) ethip4-ip4base-soak


Tput: 30 Minutes Soak Test (PLRsearch)
20.0
15.0
10.0
5.00
0.00
Critical Rate Lower Bound [Mpps]
1
2
3
4
5
6
Test Cases [Index]
1. (02 runs) avf-ethip6-ip6base-soak
2. (02 runs) ethip6-ip6base-soak
3. (02 runs) avf-eth-l2bdbasemaclrn-eth-2memif-1dcr-soak
4. (02 runs) eth-l2bdbasemaclrn-eth-2memif-1dcr-soak
5. (02 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-soak
6. (02 runs) avf-eth-l2bdbasemaclrn-eth-2vhostvr1024-1vm-vppl2xc-soak

2.7 Reconfiguration Tests

See *Reconfiguration Tests* (page 53) for methodology description of this test type.

2.7.1 VNF Service Chains

In each test, a single service chain is added, the re-configuration contains all the steps the initial chains got, except the last step (starting VMs) is skipped.

Additional information about graph data:

1. **Graph Title**: describes tested VPP packet path. Format:
   - wire encapsulation `dot1qip4vlxan`,
   - VPP forwarding mode `l2bd`,
   - total number {Y} of initial service chains `{Y}ch`,
   - total number of additional chains being reconfigured `1ach`,
   - total number of initial vhost-user interfaces forwarding packets on VPP with {Y} chains and {X} VMs per chain `{2XY}vh` (2 interfaces per {X} VMs per {Y} chains),
   - total number {XY} of (both initial and final) VNF VMs forwarding packets `{XY}vm` and finally
   - VNF workload in VM `testpmd`.
2. **X-axis Labels**: indices of individual test suites as listed in Graph Legend.
3. **Y-axis Labels**: measured Effective Blocked Time [s] values.
4. **Graph Legend**: lists X-axis indices with associated CSIT test suites executed to generate graphed test results and the average value of packet loss (measured in packets).
5. **Hover Information**: lists minimum, first quartile, median, third quartile, and maximum. If either type of outlier is present the whisker on the appropriate side is taken to 1.5×IQR from the quartile (the "inner fence") rather than the max or min, and individual outlying data points are displayed as unfilled circles (for suspected outliers) or filled circles (for outliers). (The "outer fence" is 3×IQR from the quartile.)

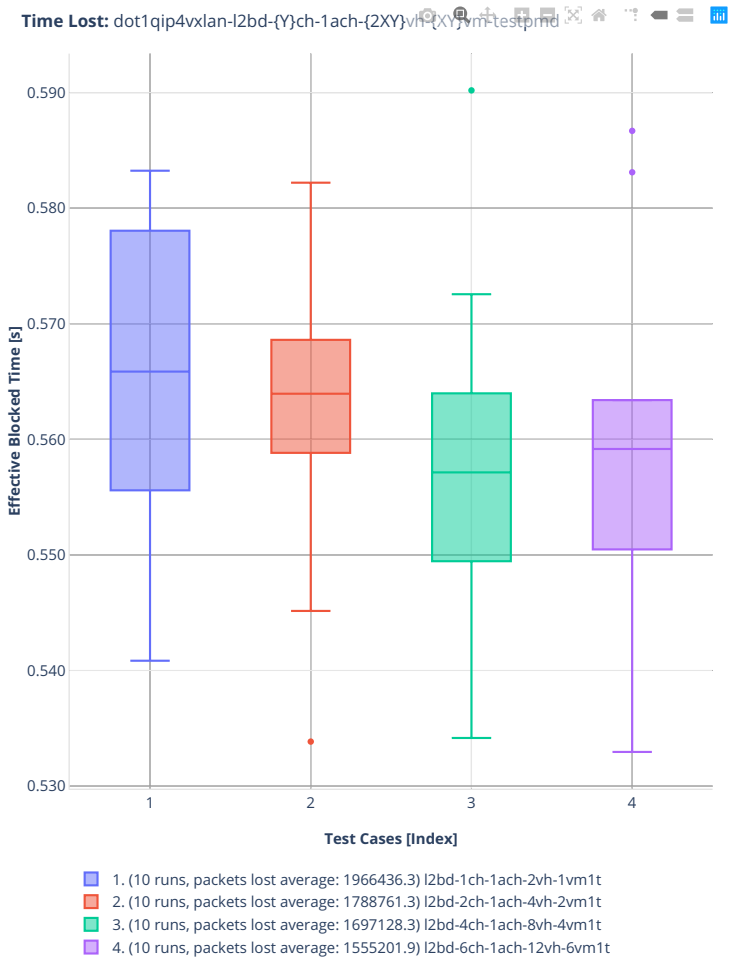
Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[161], build logs from FD.io vpp performance job 2n-clx[162] with RF result files csit-vpp-perf-2210-*.zip archived here.

---

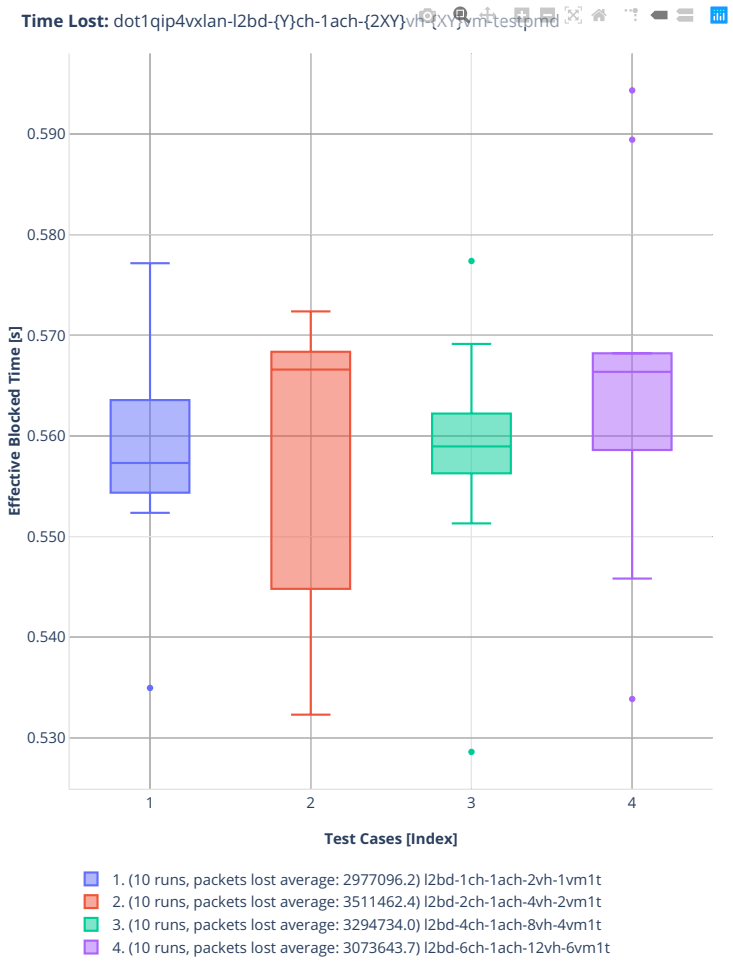
[161] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[162] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx

2n-icx-xxv710

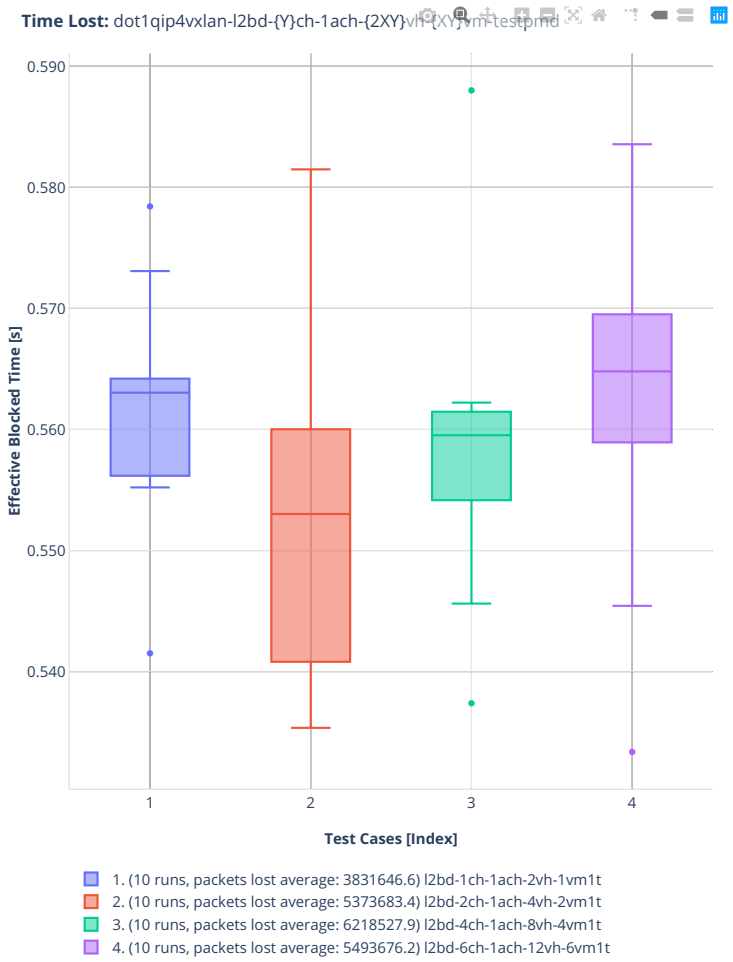
imix-2t1c-dot1qip4vxlan-l2bd



imix-4t2c-dot1qip4vxlan-l2bd

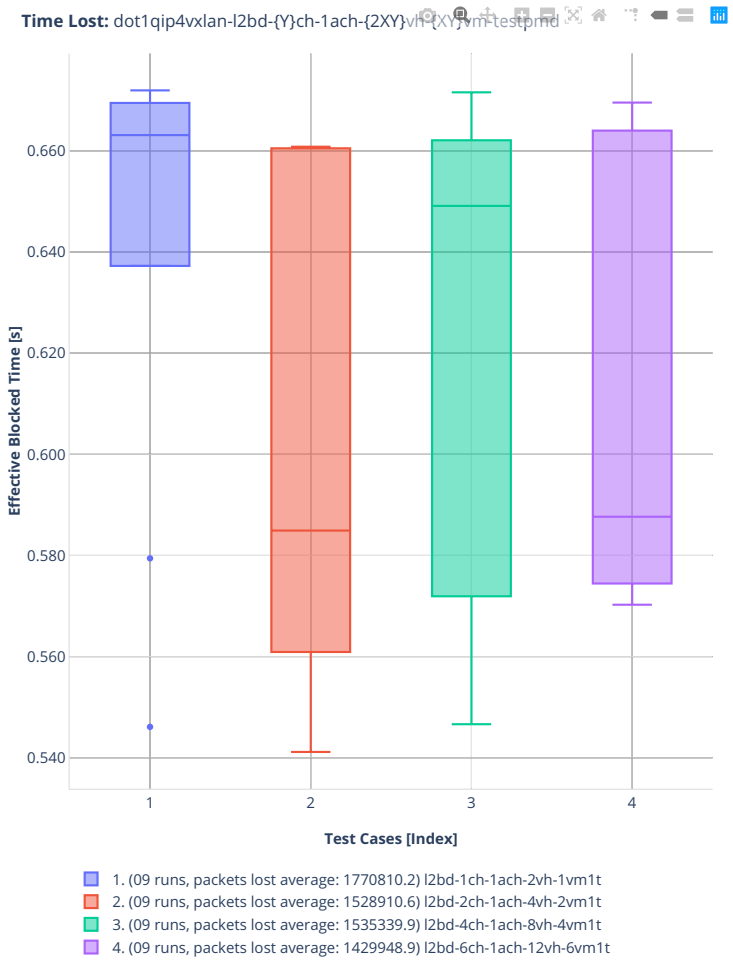


imix-8t4c-dot1qip4vxlan-l2bd

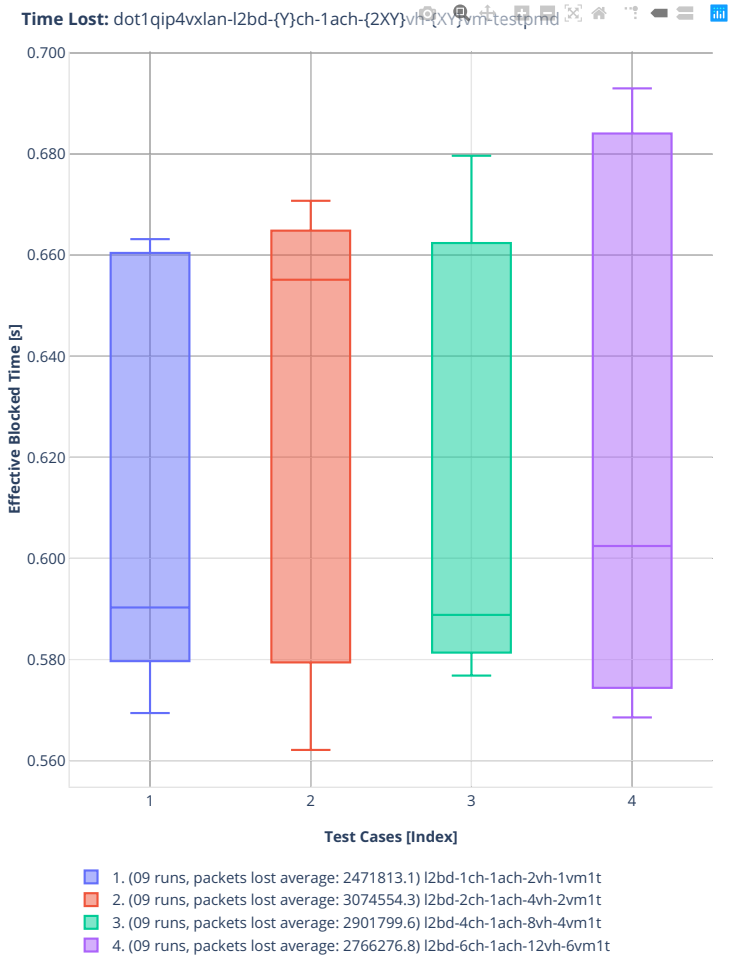


2n-clx-xxv710

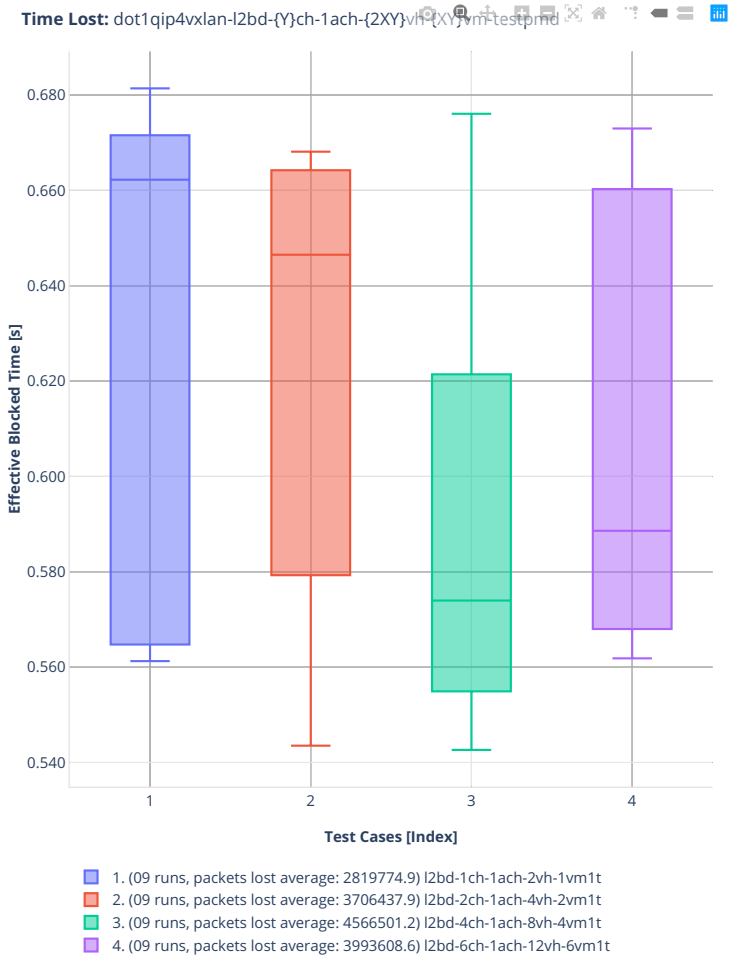
imix-2t1c-dot1qip4vxlan-l2bd



imix-4t2c-dot1qip4vxlan-l2bd



imix-8t4c-dot1qip4vxlan-l2bd



2.8 NFV Service Density

NFV Service Density is benchmarked in three distinct NF service configurations:

- VNF Service Chains Routing
- CNF Service Chains Routing
- CNF Service Pipelines Routing
- VNF Service Chains Tunnels

Each configuration is tested in a number of service density combinations [Number of Service Instances] x [Number of NFs per Service Instance]. The actual tested range is based on available CPU physical core resources.

2.8.1 VNF Service Chains Routing

Throughput graphs for VNF service chains are generated by multiple executions of tests covering a range of VNF service densities defined as [Number of Service Chains] x [Number of VNFs per Service Chain]. The results are presented in the service density graph. Each graph includes the results of both configurations: one NF per physical core and two NFs per physical core and their relative difference.

Additional information about graph data:

1. **Graph Title**: describes tested packet path including VNF workload running in each VM.
2. **X-axis Labels**: VNFs per service chain.
3. **Y-axis Labels**: number of service chains.
4. **Z-axis Color Scale**: lists 64B/IMIX Packet Throughput (mean MRR/NDR/PDR value) in Mpps or the Relative Difference.
5. **Hover Information**: specific test substring listing vhost-chain-vm combinations, number of runs executed, mean MRR/NDR/PDR throughput in Mpps, standard deviation for both configurations and their relative difference.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[163], build logs from FD.io vpp performance job 2n-clx[164] with RF result files csit-vpp-perf-2210-*.zip archived here.

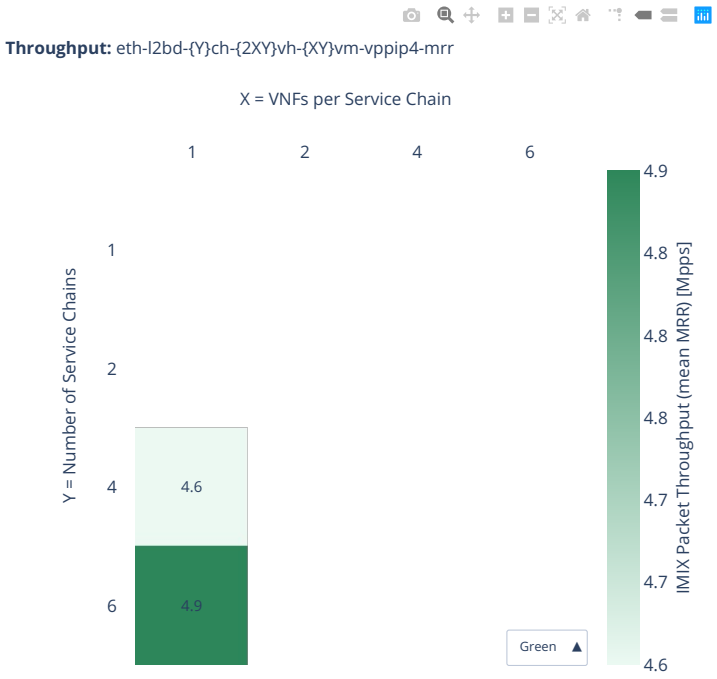
---

[163] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[164] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx

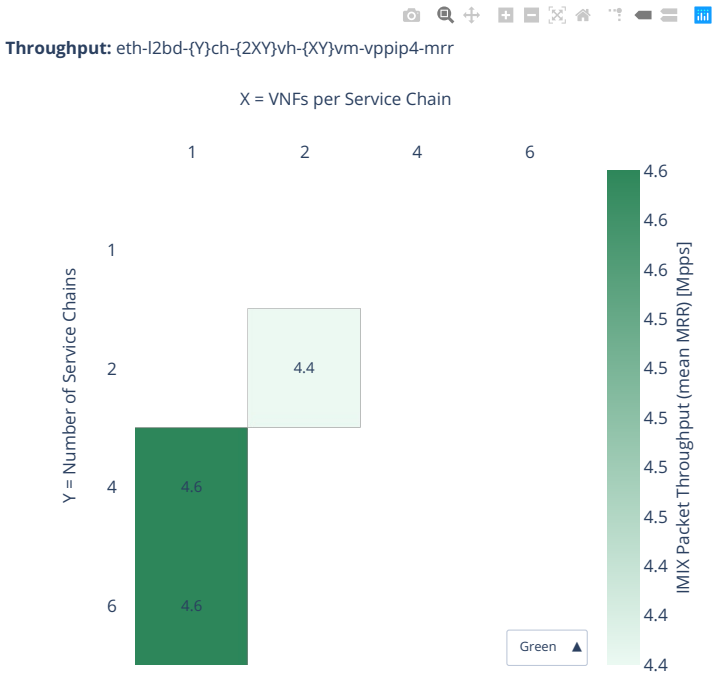
2n-icx-xxv710-mrr

imix-2t1c-eth-l2bd

imix-4t2c-eth-l2bd



imix-8t4c-eth-l2bd

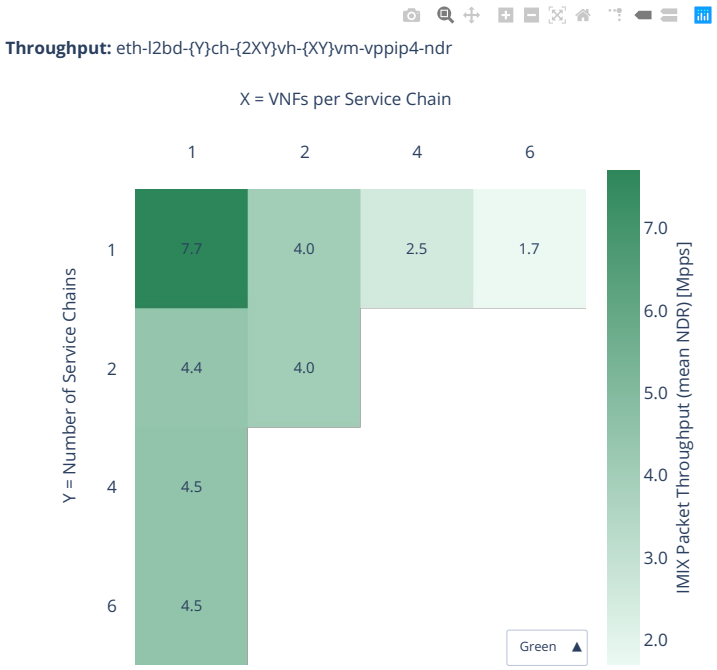


2n-icx-xxv710-ndr

imix-2t1c-eth-l2bd

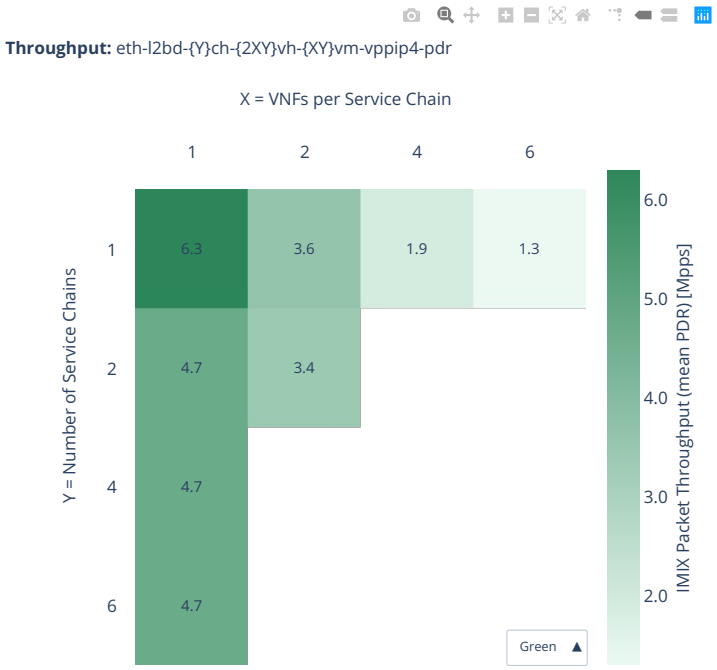
imix-4t2c-eth-l2bd

imix-8t4c-eth-l2bd

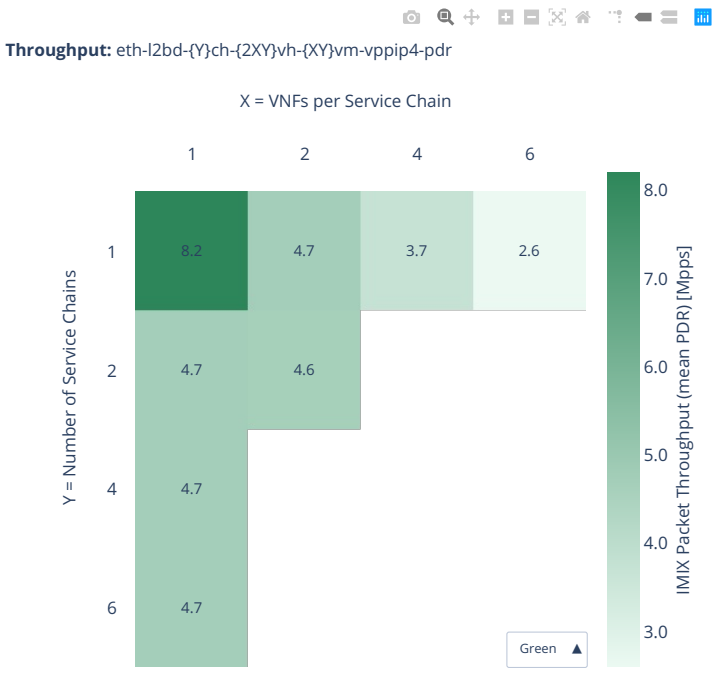


2n-icx-xxv710-pdr

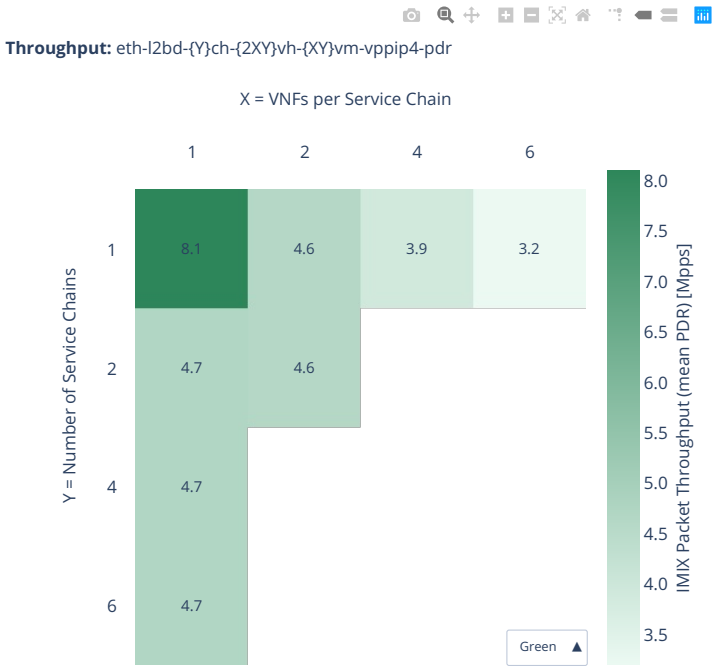
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

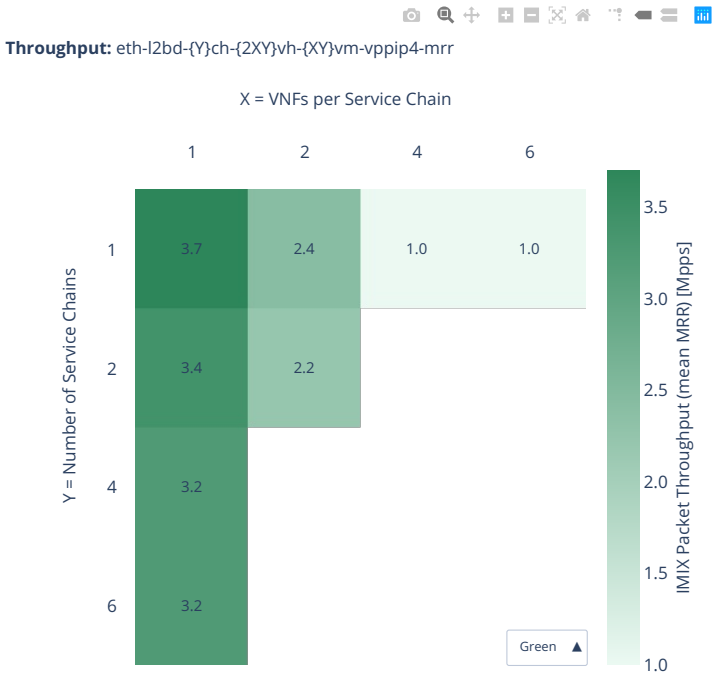


imix-8t4c-eth-l2bd

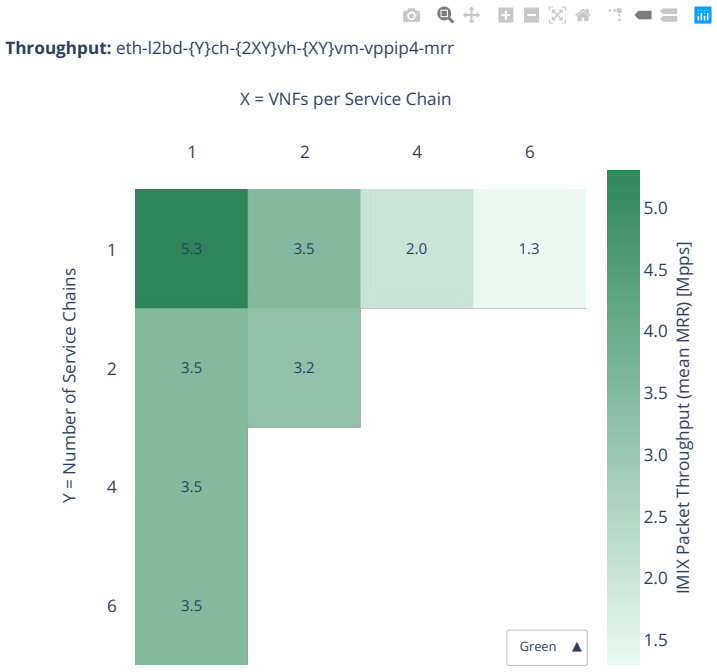


2n-clx-xxv710-mrr

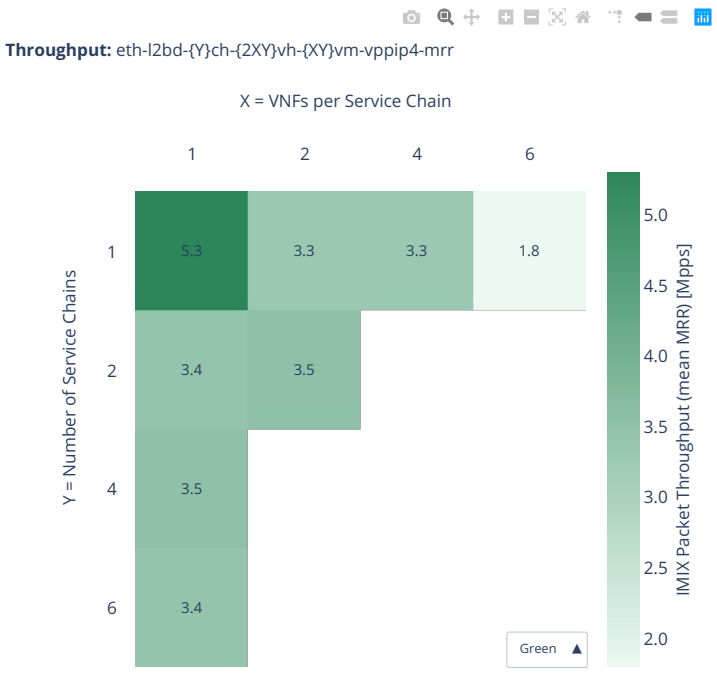
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

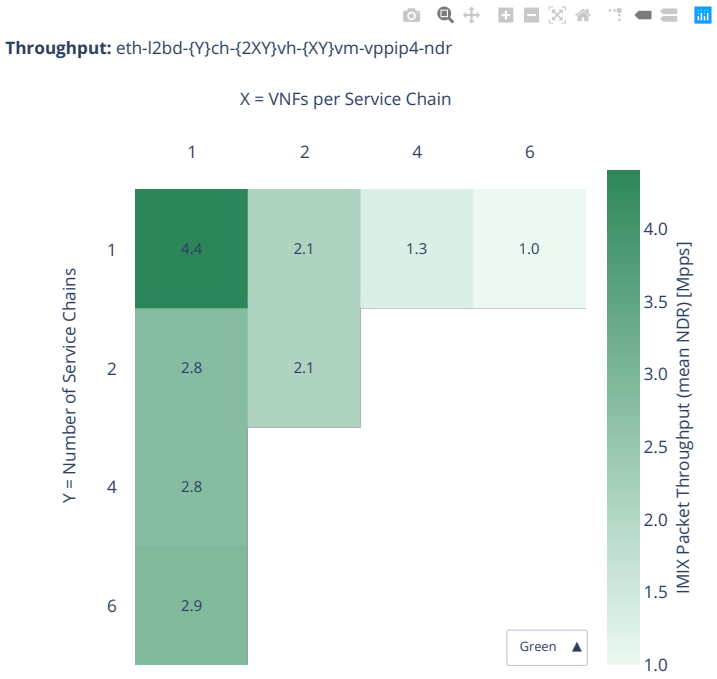


imix-8t4c-eth-l2bd

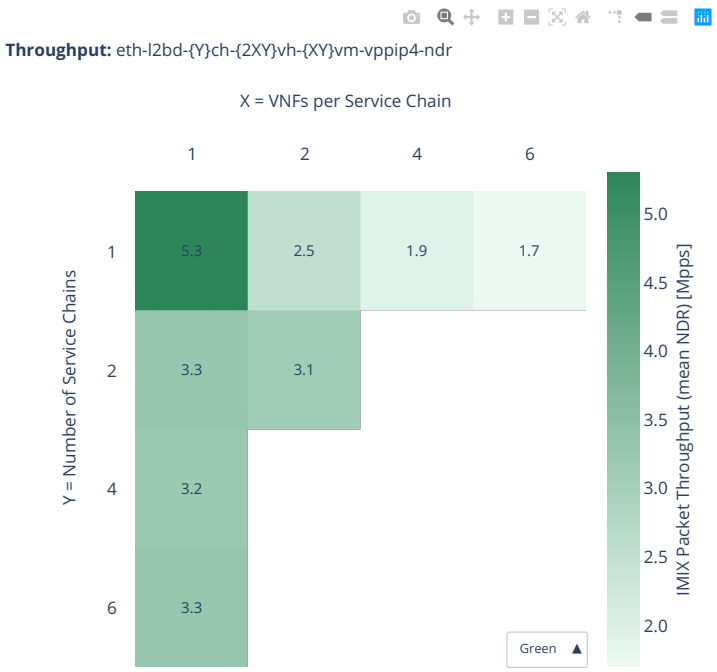


2n-clx-xxv710-ndr

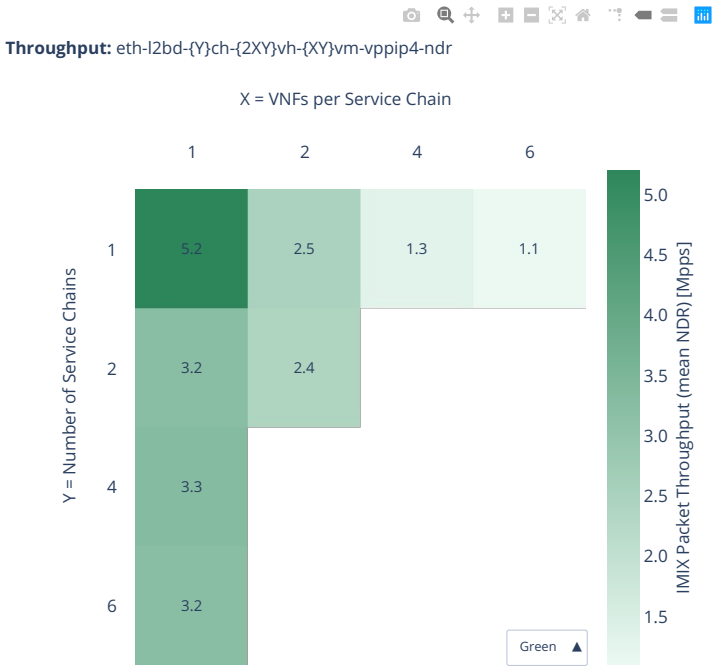
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

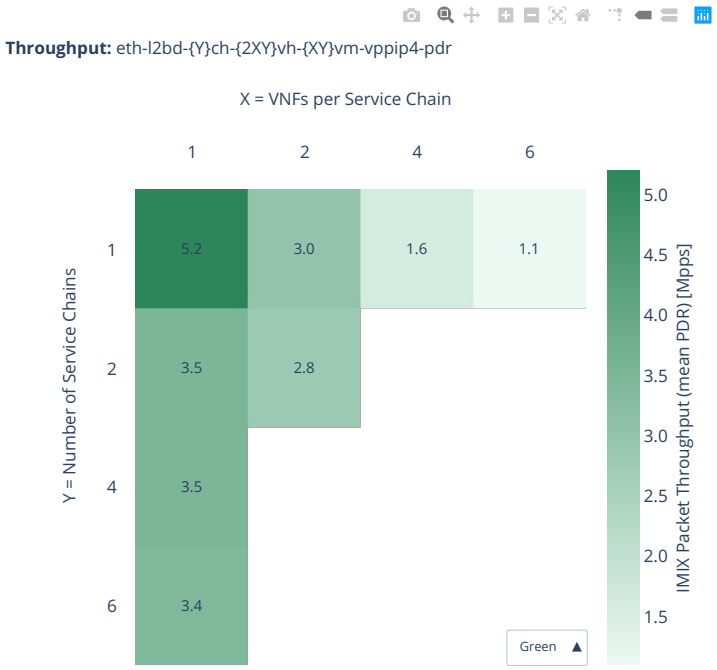


imix-8t4c-eth-l2bd

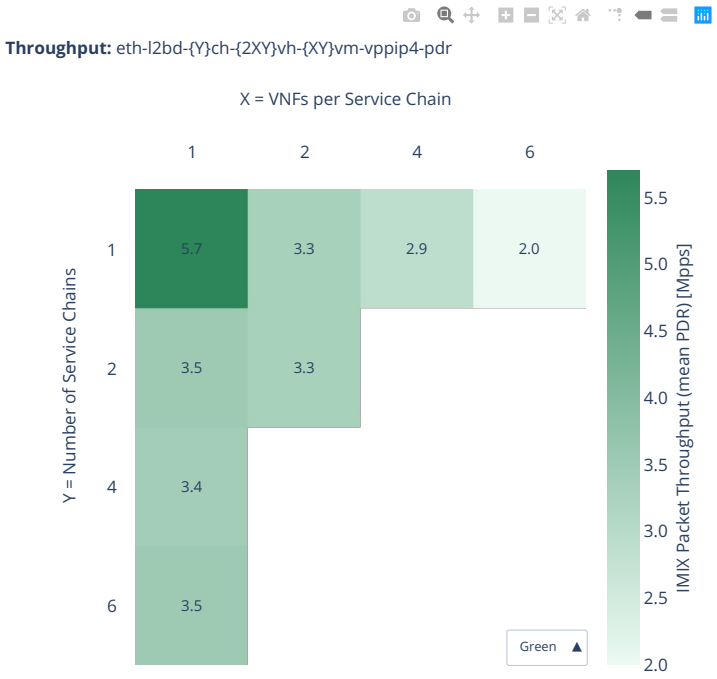


2n-clx-xxv710-pdr

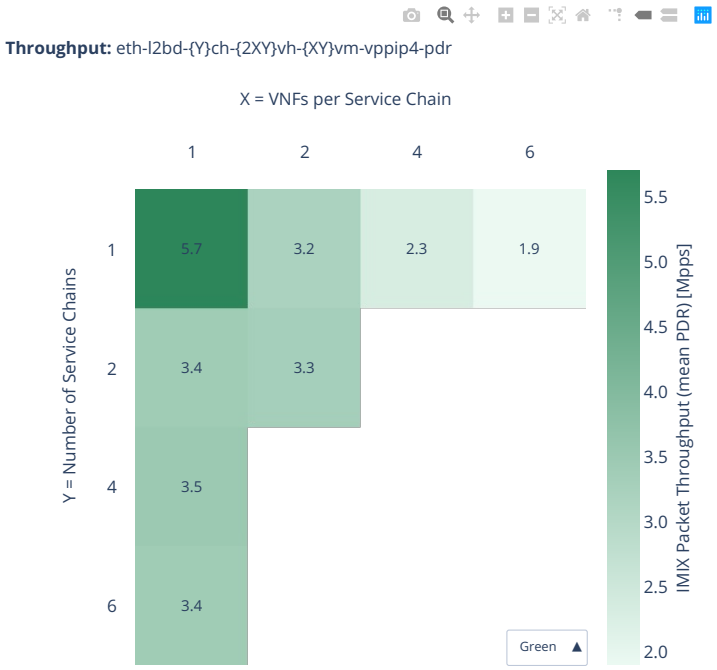
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd



imix-8t4c-eth-l2bd



2.8.2 CNF Service Chains Routing

Throughput graphs for CNF service chains are generated by multiple executions of tests covering a range of CNF service densities defined as [Number of Service Chains] x [Number of CNFs per Service Chain]. The results are presented in the service density graph. Each graph includes the results of both configurations: one NF per physical core and two NFs per physical core and their relative difference.

Additional information about graph data:

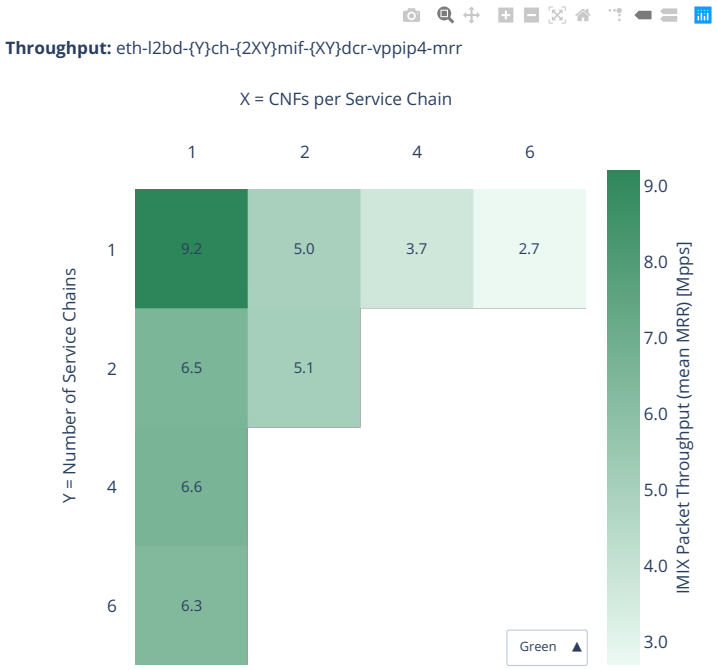
1. **Graph Title**: describes tested packet path including CNF workload running in each Docker Container.
2. **X-axis Labels**: CNFs per service chain.
3. **Y-axis Labels**: number of service chains.
4. **Z-axis Color Scale**: lists 64B/IMIX Packet Throughput (mean MRR/NDR/PDR value) in Mpps or the Relative Difference.
5. **Hover Information**: specific test substring listing memif-chain-docker_container combinations, number of runs executed, mean MRR/NDR/PDR throughput in Mpps, standard deviation for both configurations and their relative difference.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[165], build logs from FD.io vpp performance job 2n-clx[166] with RF result files csit-vpp-perf-2210-*.zip archived here.

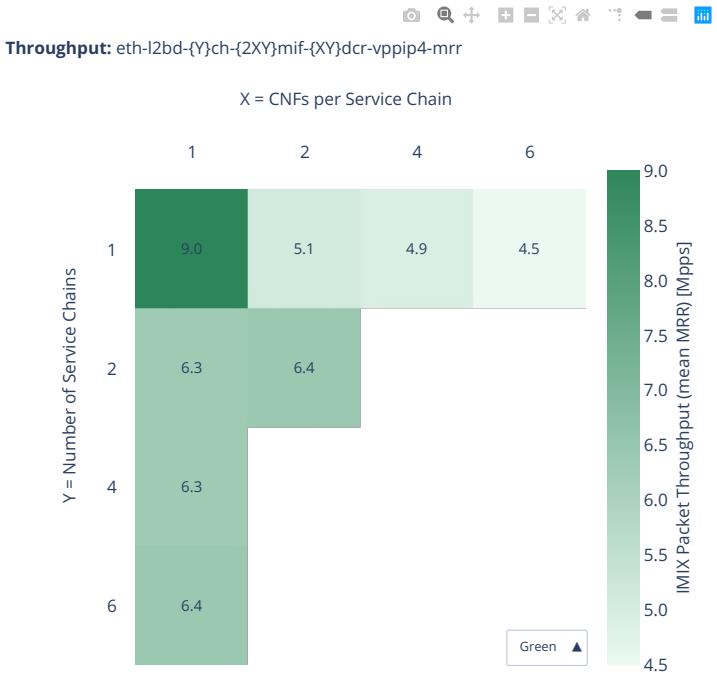
[165] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[166] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx

2n-icx-xxv710-mrr

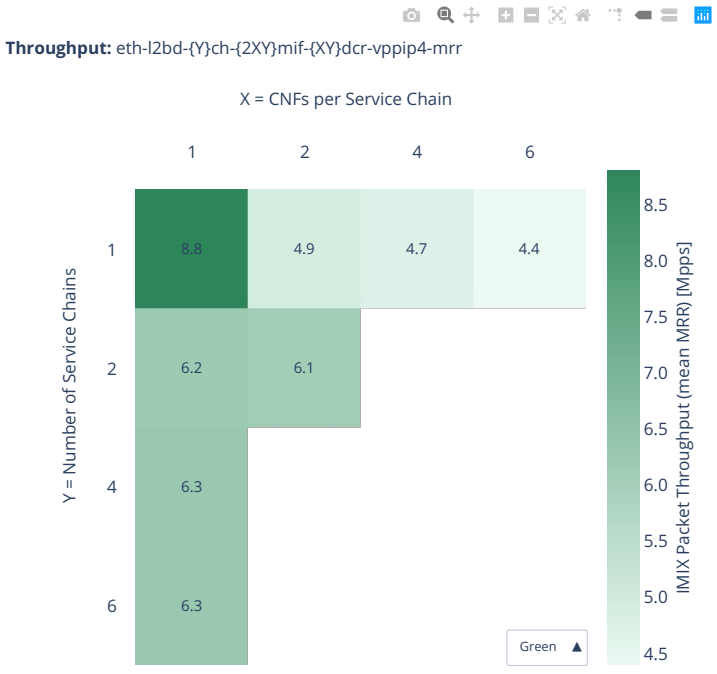
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

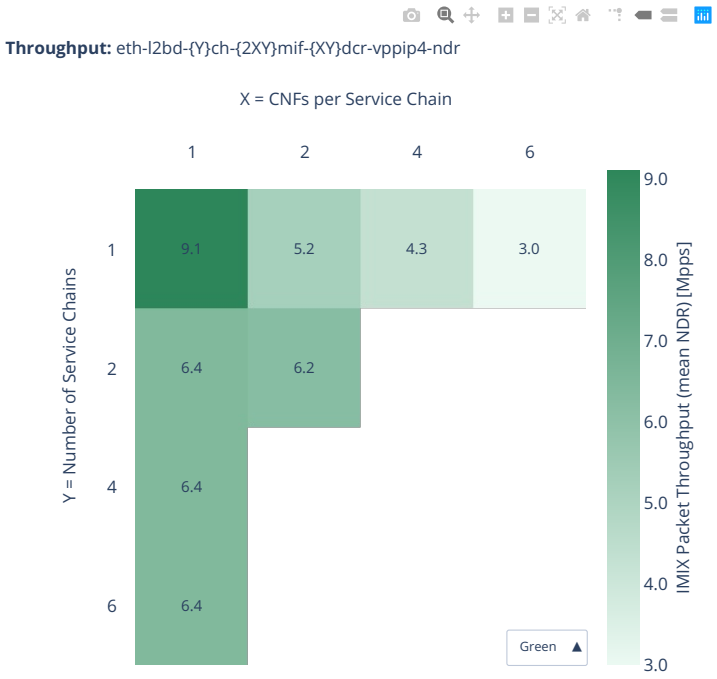


imix-8t4c-eth-l2bd

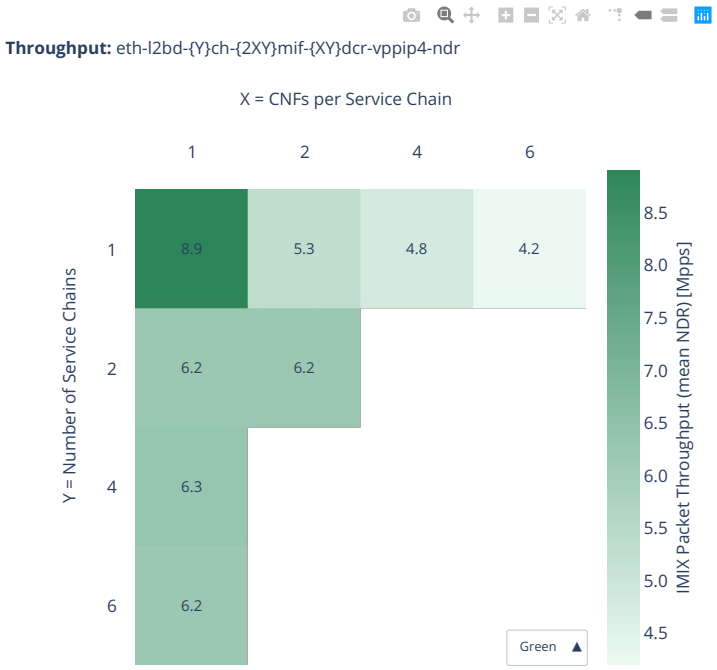


2n-icx-xxv710-ndr

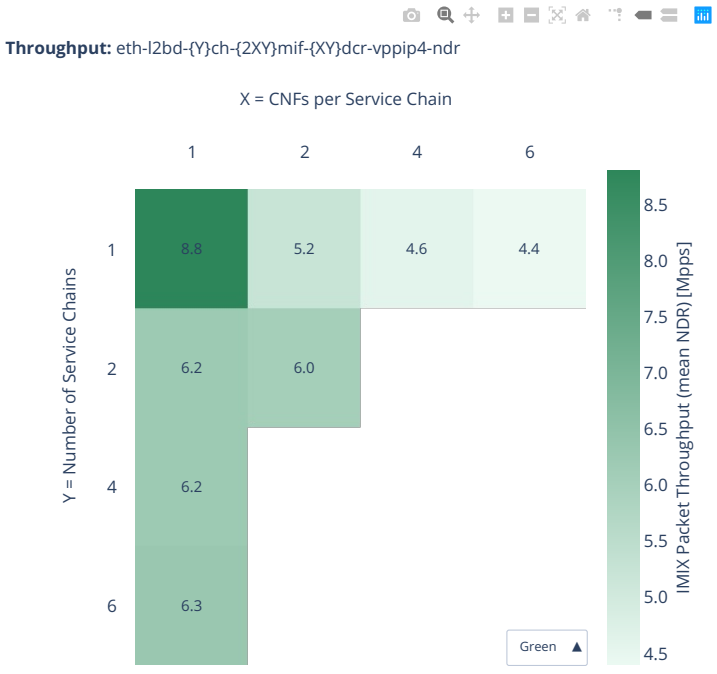
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

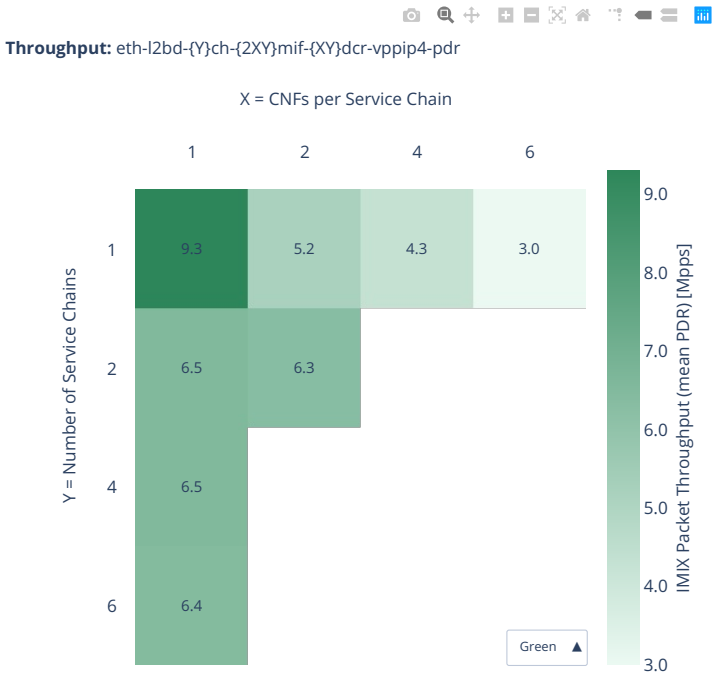


imix-8t4c-eth-l2bd

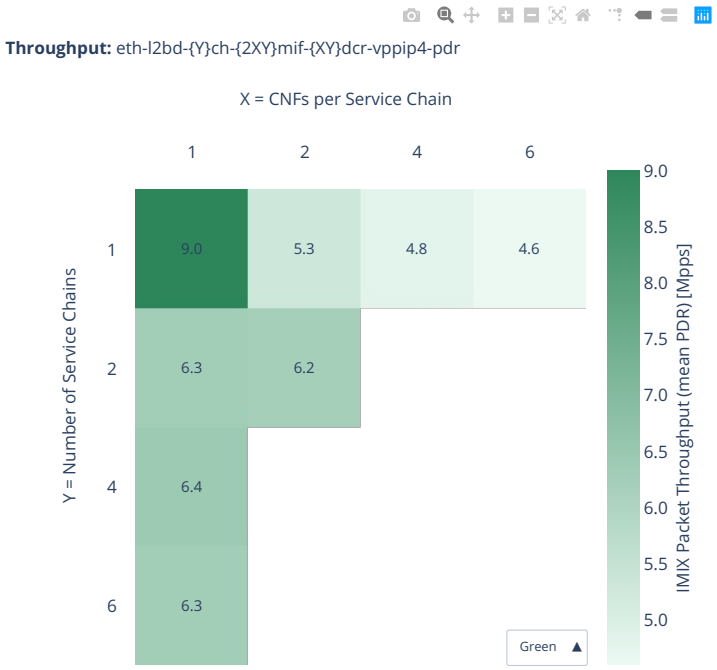


2n-icx-xxv710-pdr

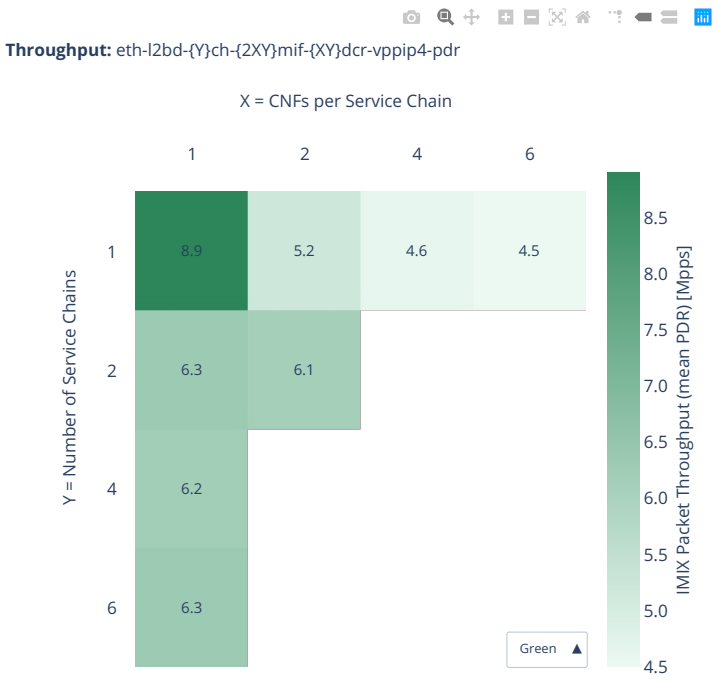
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

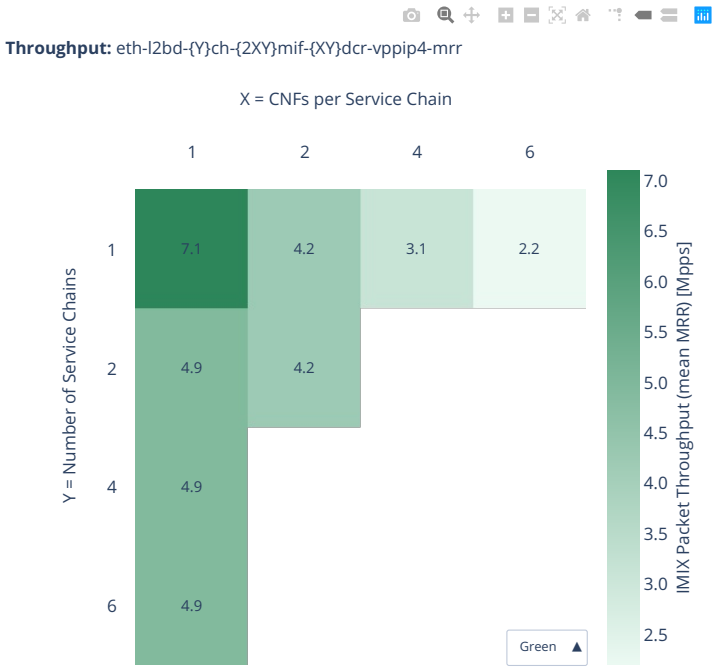


imix-8t4c-eth-l2bd

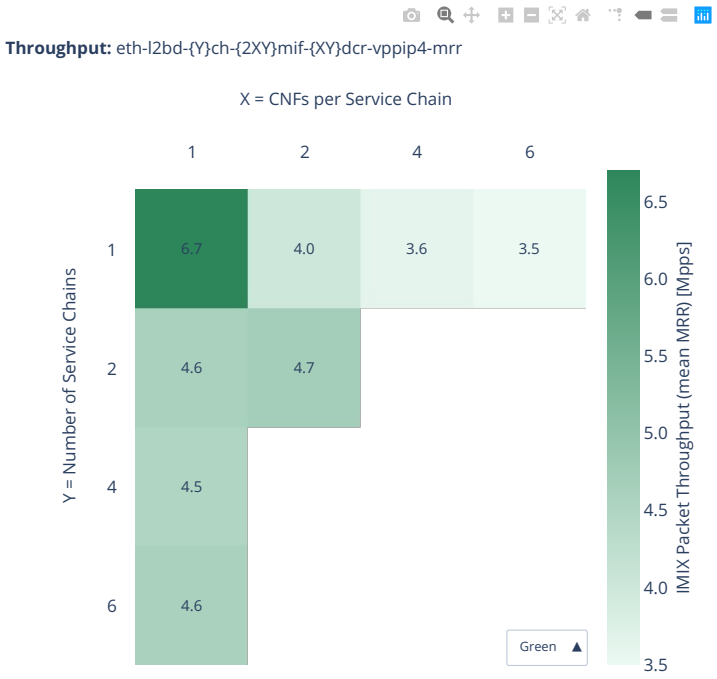


2n-clx-xxv710-mrr

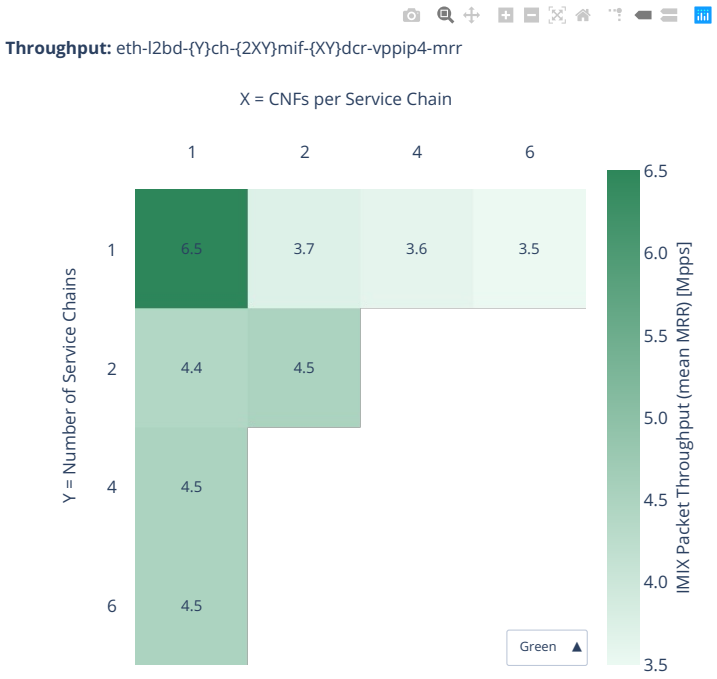
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

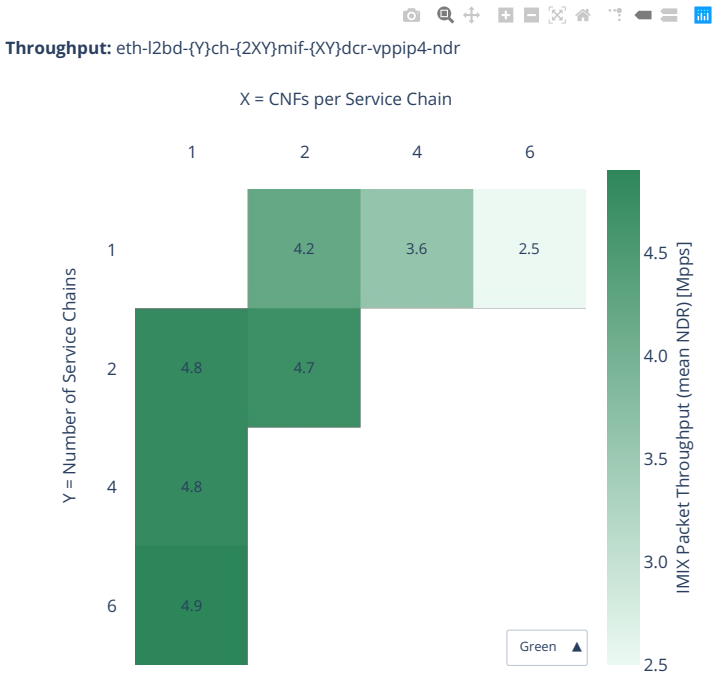


imix-8t4c-eth-l2bd

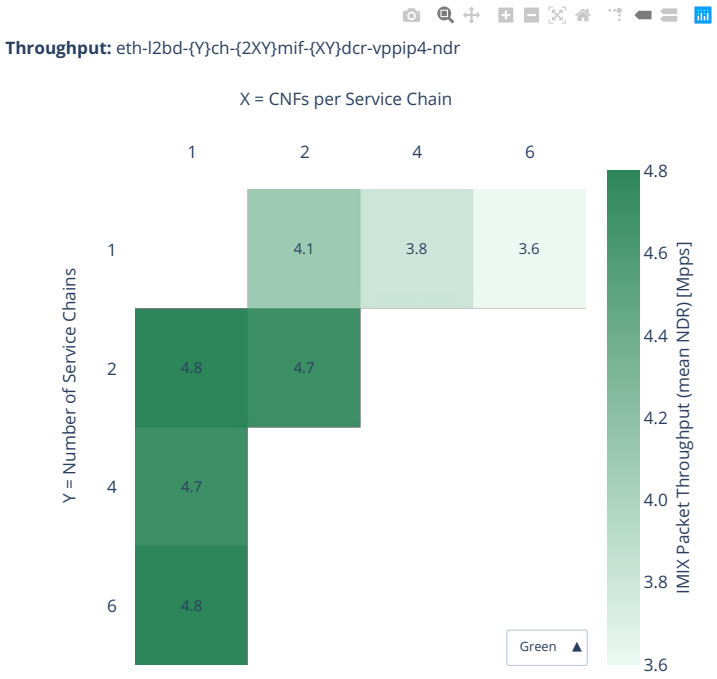


2n-clx-xxv710-ndr

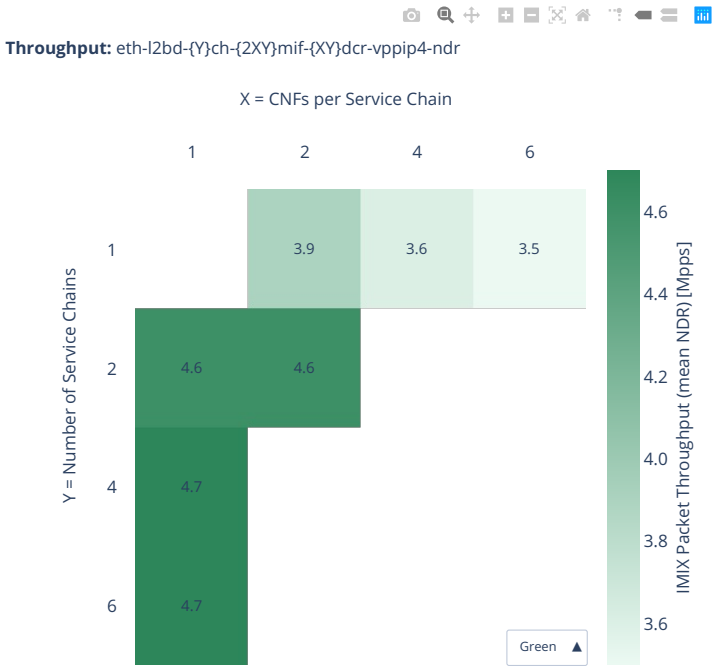
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

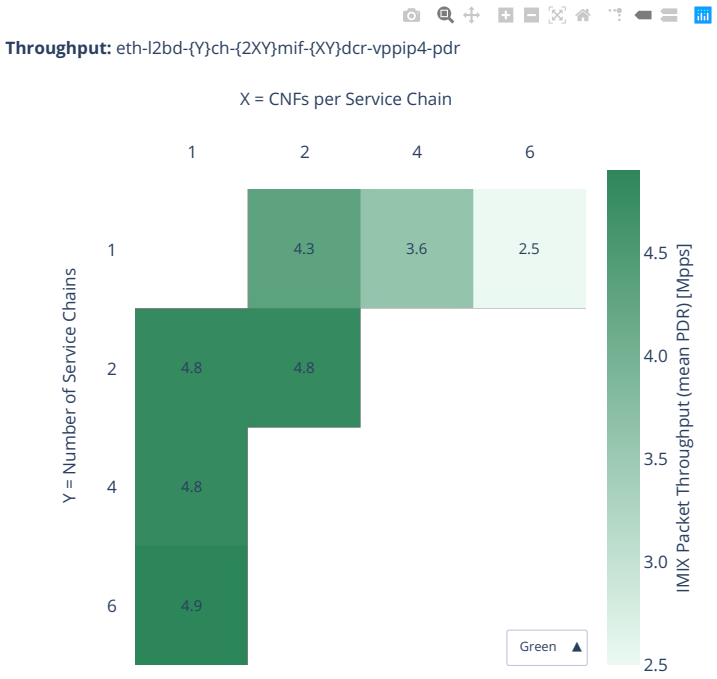


imix-8t4c-eth-l2bd

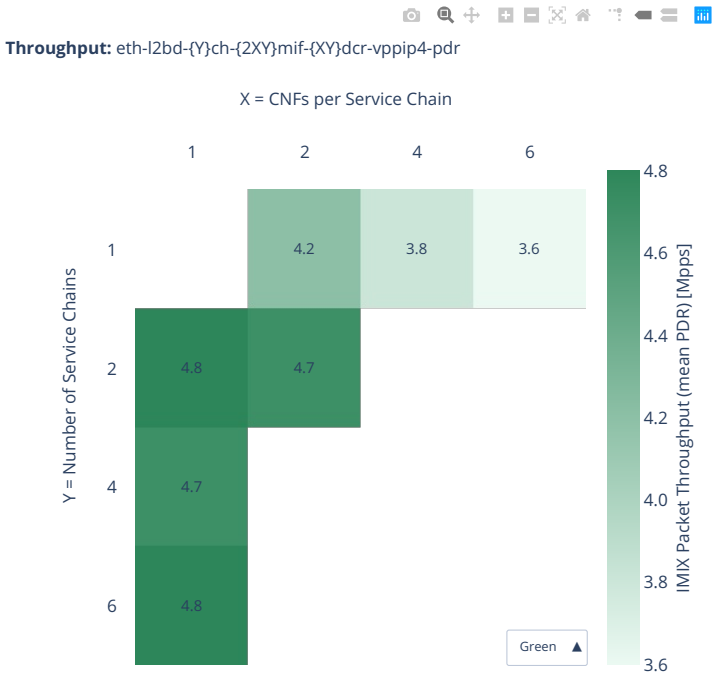


2n-clx-xxv710-pdr

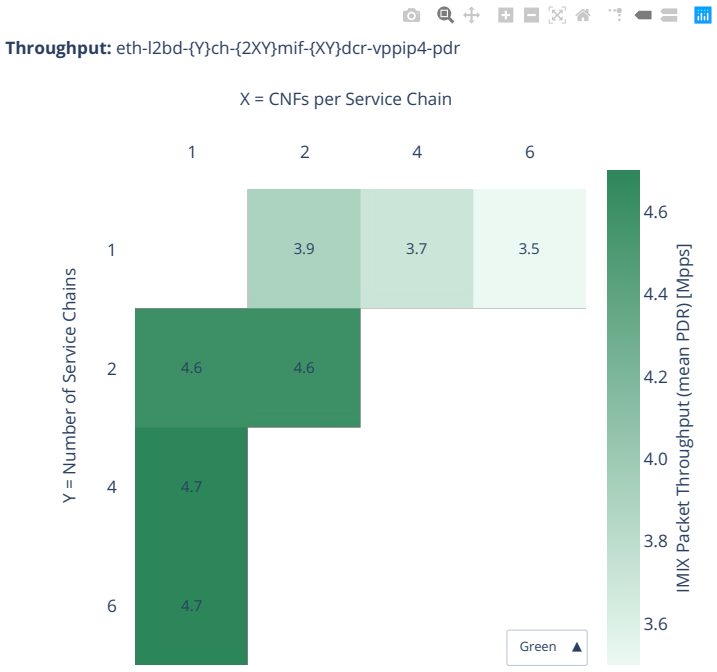
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd



imix-8t4c-eth-l2bd



2.8.3 CNF Service Pipelines Routing

Throughput graphs for CNF service pipelines are generated by multiple executions of tests covering a range of CNF service densities defined as [Number of Service Pipelines] x [Number of CNFs per Service Pipeline]. The results are presented in the service density graph. Each graph includes the results of both configurations: one NF per physical core and two NFs per physical core and their relative difference.

Additional information about graph data:

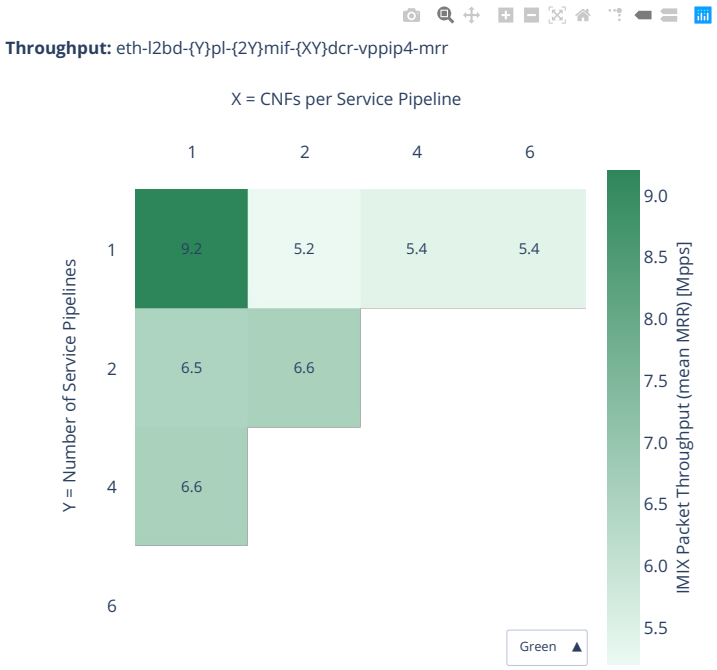
1. **Graph Title**: describes tested packet path including CNF workload running in each Docker Container.
2. **X-axis Labels**: CNFs per service pipeline.
3. **Y-axis Labels**: number of service pipelines.
4. **Z-axis Color Scale**: lists 64B/IMIX Packet Throughput (mean MRR/NDR/PDR value) in Mpps or the Relative Difference.
5. **Hover Information**: specific test substring listing memif-pipeline-docker_container combinations, number of runs executed, mean MRR/NDR/PDR throughput in Mpps, standard deviation for both configurations and their relative difference.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[167], build logs from FD.io vpp performance job 2n-clx[168] with RF result files csit-vpp-perf-2210-*.zip archived here.

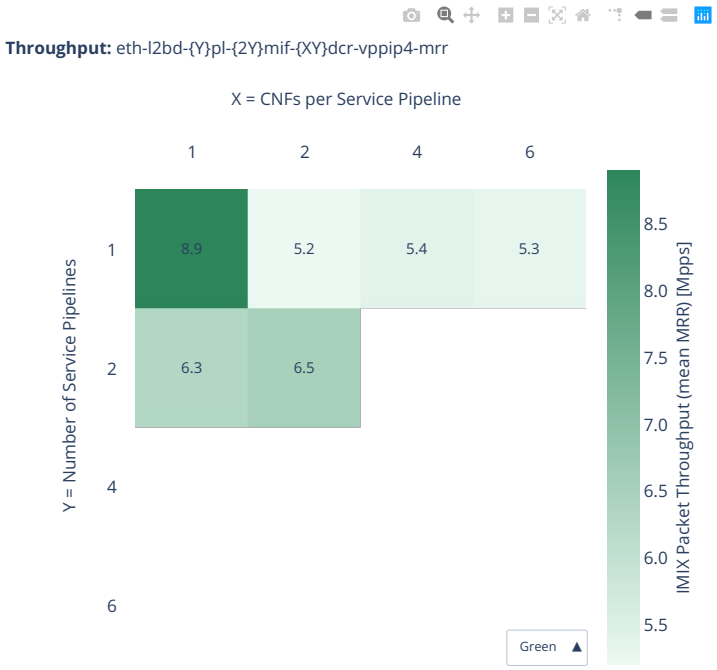
[167] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[168] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx

2n-icx-xxv710-mrr

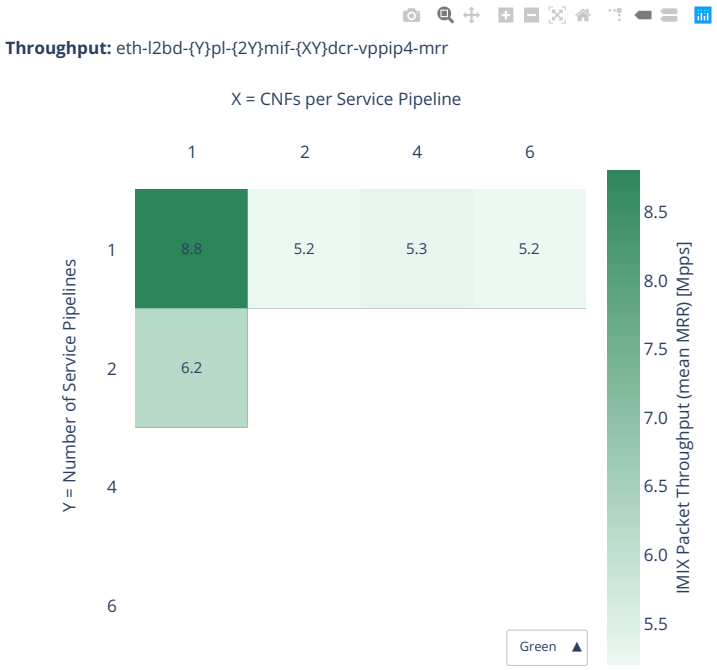
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

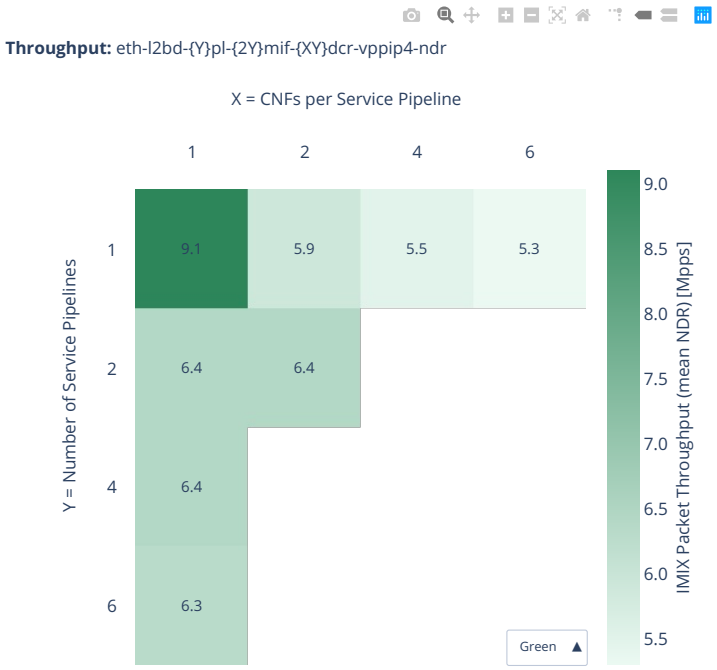


imix-8t4c-eth-l2bd

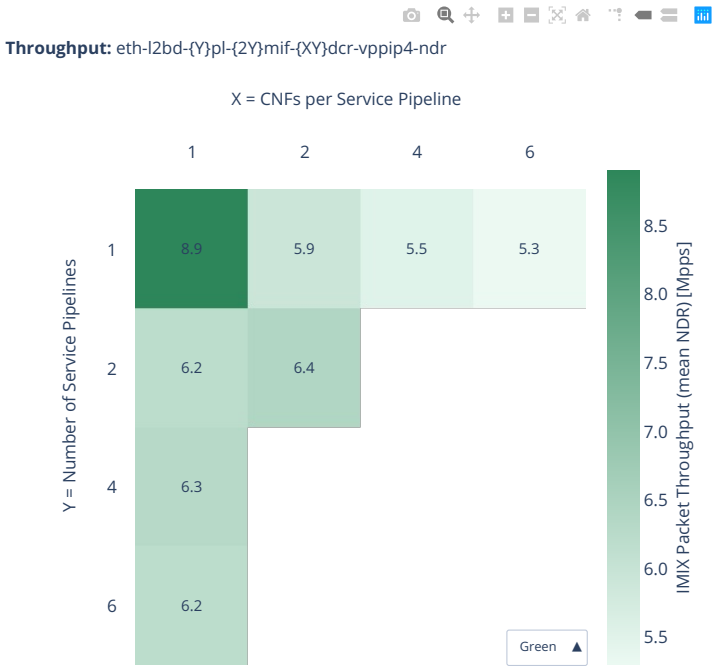


2n-icx-xxv710-ndr

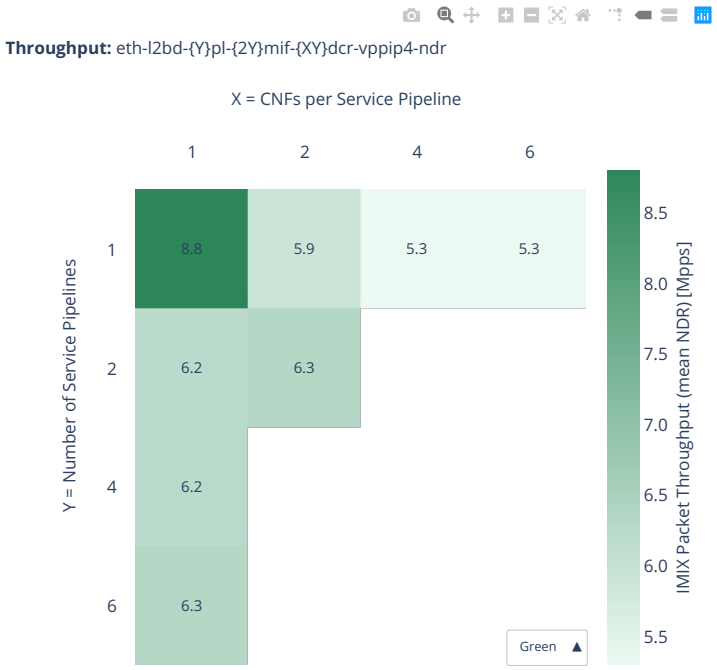
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

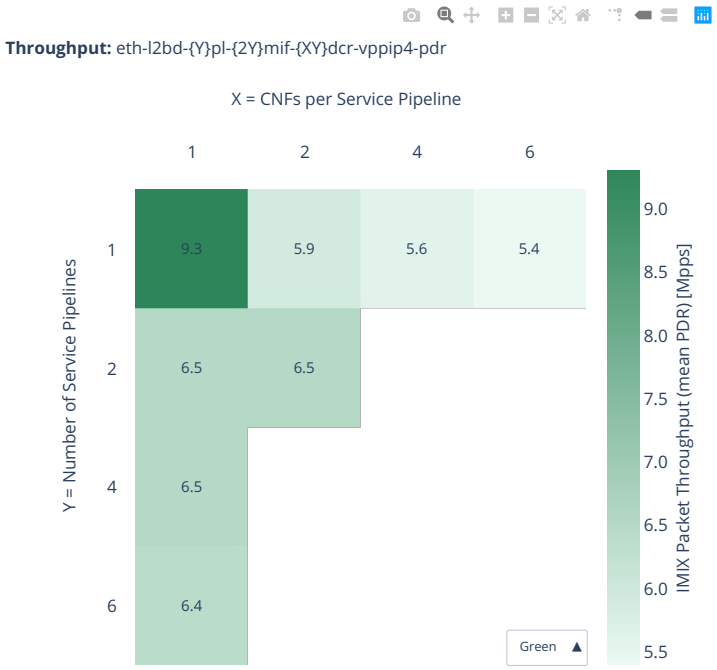


imix-8t4c-eth-l2bd

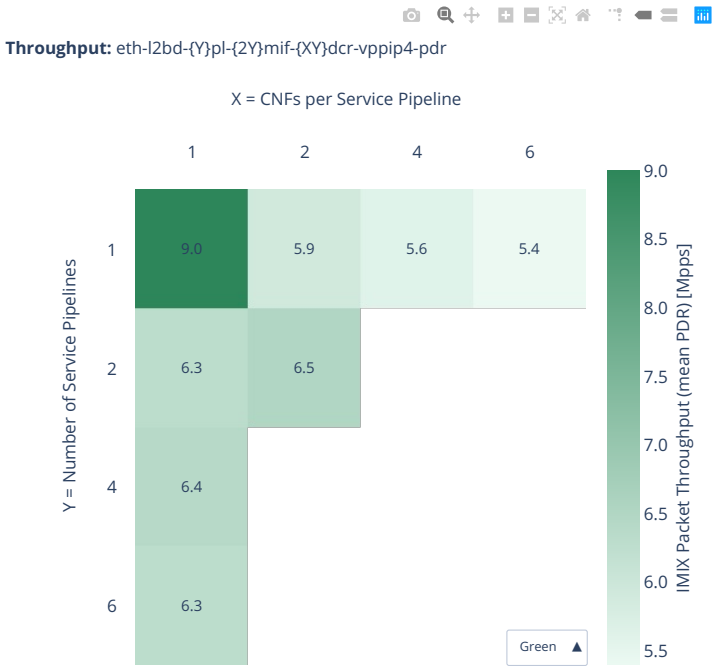


2n-icx-xxv710-pdr

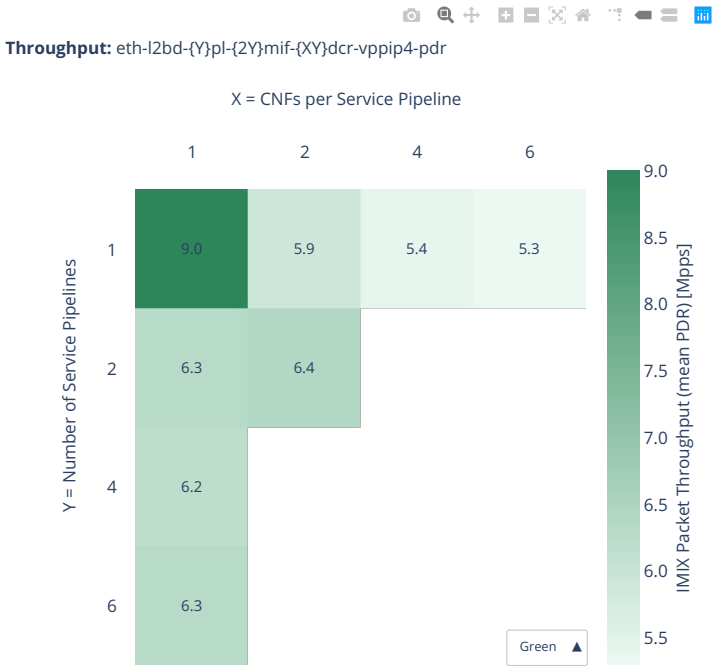
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

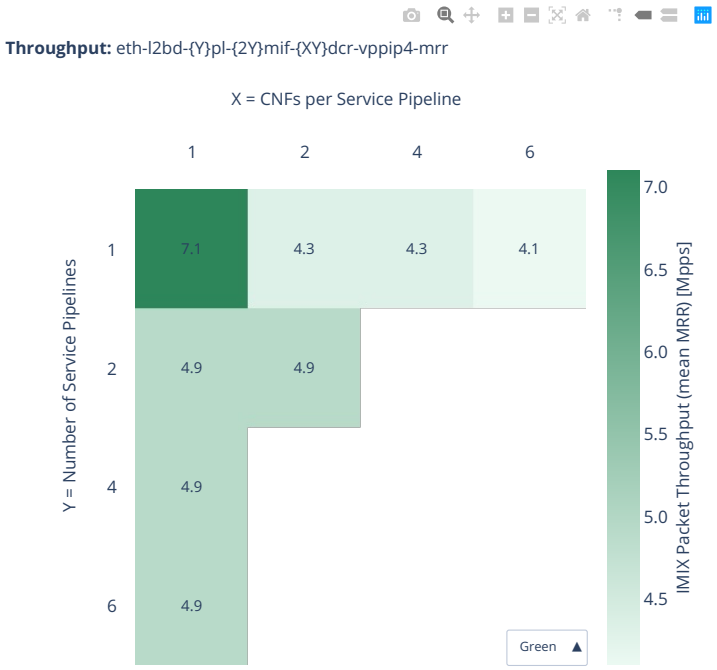


imix-8t4c-eth-l2bd

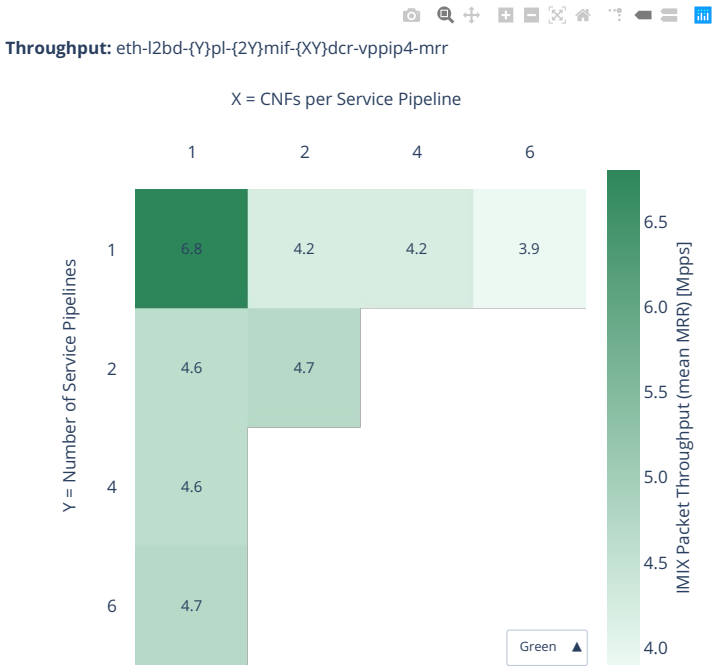


2n-clx-xxv710-mrr

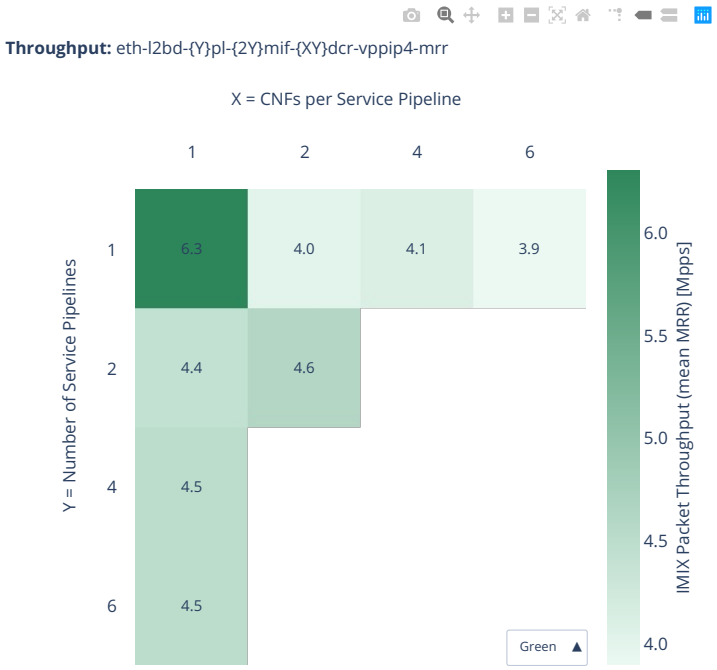
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

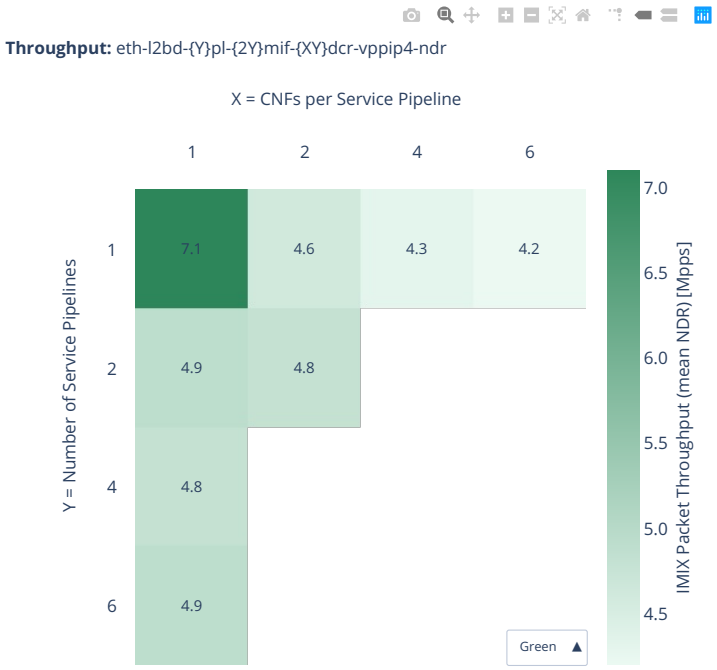


imix-8t4c-eth-l2bd

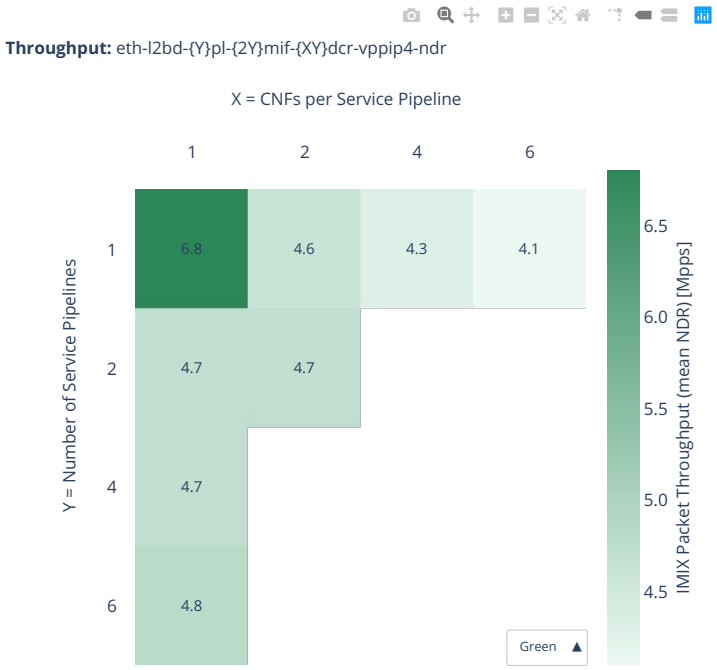


2n-clx-xxv710-ndr

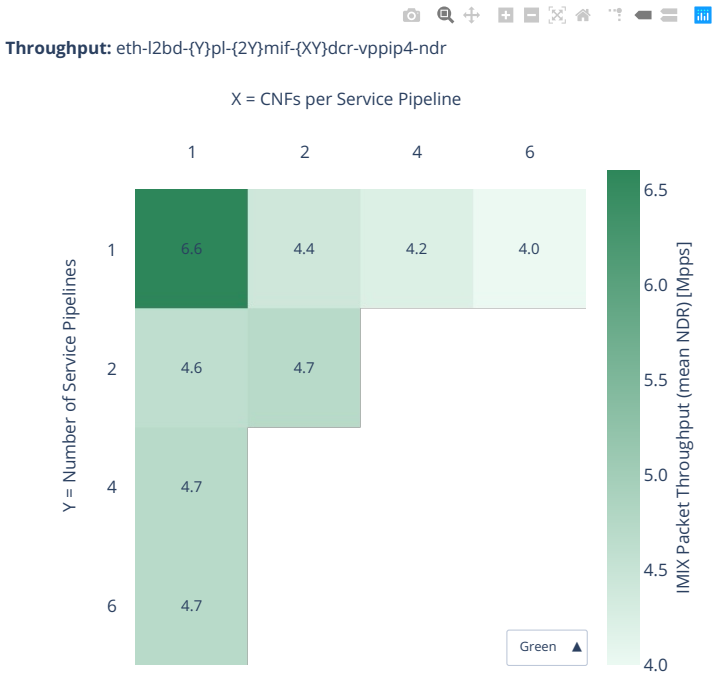
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

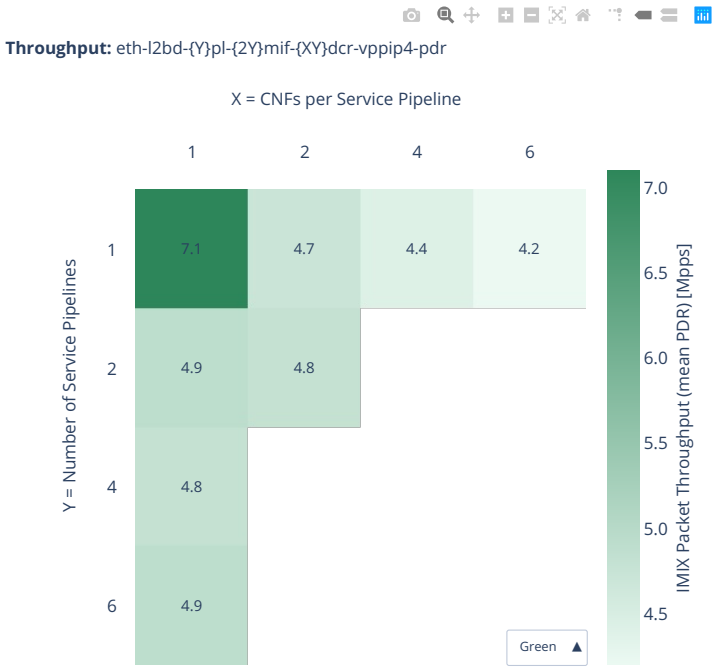


imix-8t4c-eth-l2bd

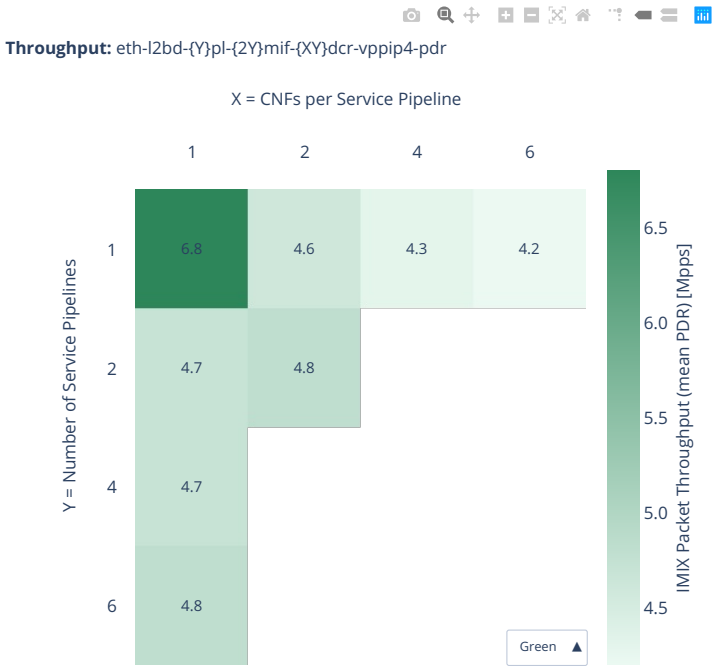


2n-clx-xxv710-pdr

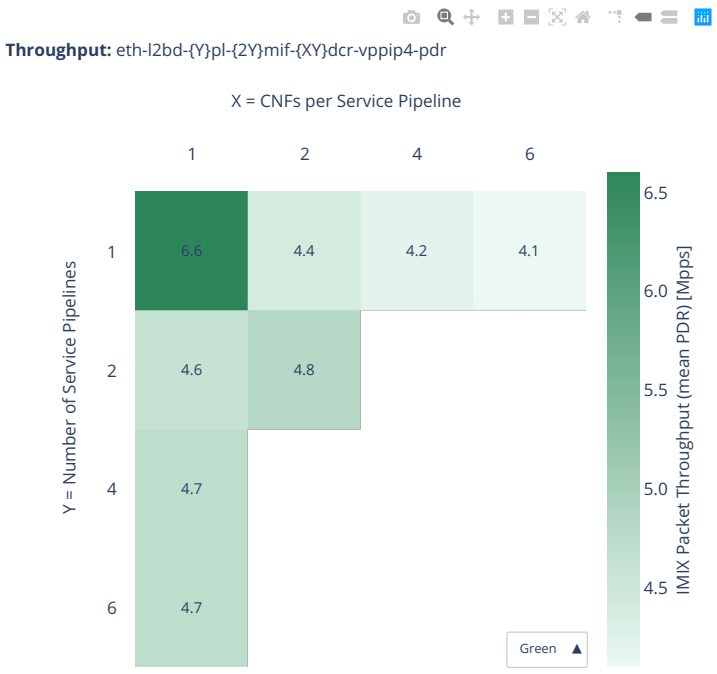
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd



imix-8t4c-eth-l2bd



2.8.4 VNF Service Chains Tunnels

Additional information about graph data:

1. **Graph Title**: describes tested packet path including VNF workload running in each VM.
2. **X-axis Labels**: VNFs per service chain.
3. **Y-axis Labels**: number of service chains.
4. **Z-axis Color Scale**: lists 64B/IMIX Packet Throughput (mean MRR/NDR/PDR value) in Mpps or the Relative Difference.
5. **Hover Information**: specific test substring listing vhost-chain-vm combinations, number of runs executed, mean MRR/NDR/PDR throughput in Mpps, standard deviation for both configurations and their relative difference.

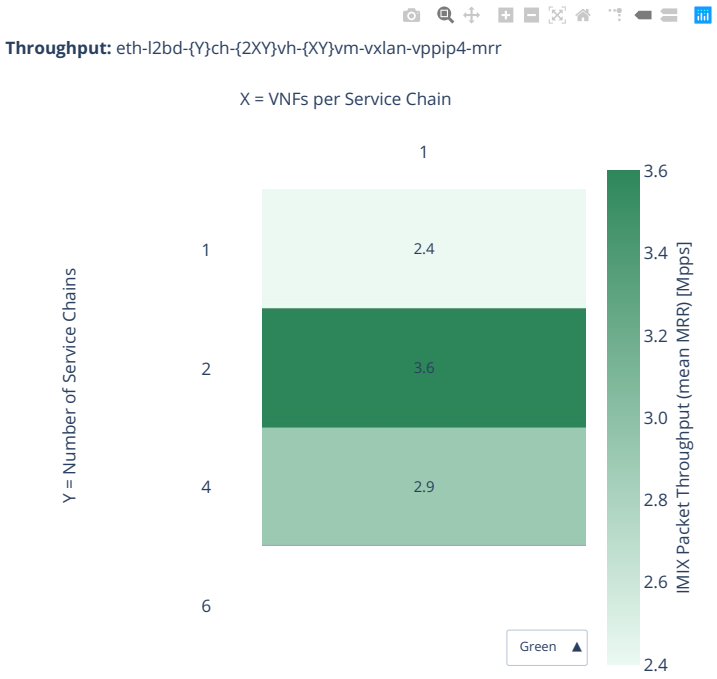
Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[169], build logs from FD.io vpp performance job 2n-clx[170] with RF result files csit-vpp-perf-2210-*.zip archived here.

---

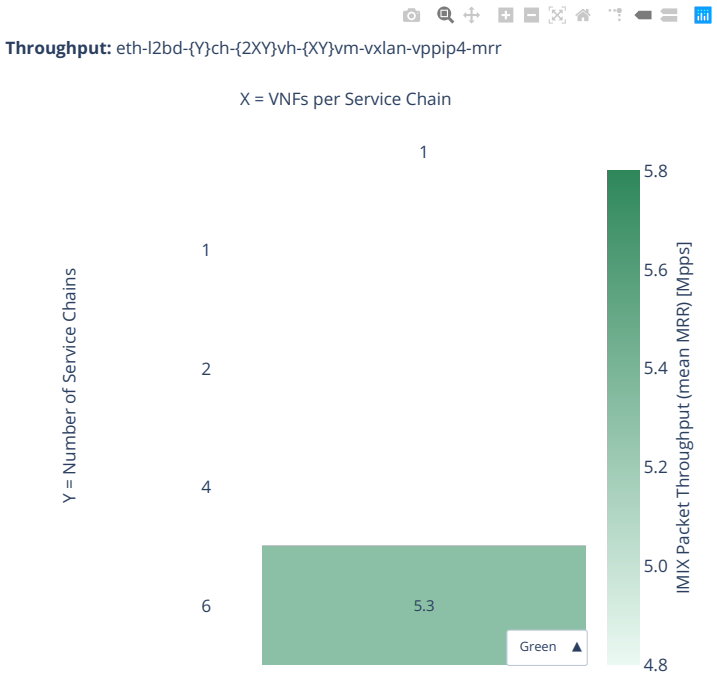
[169] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[170] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx

2n-icx-xxv710-mrr

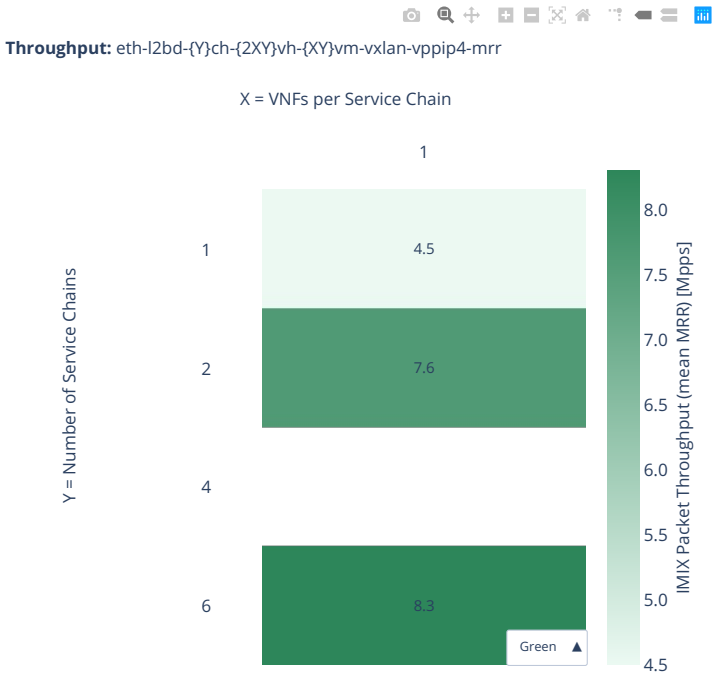
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

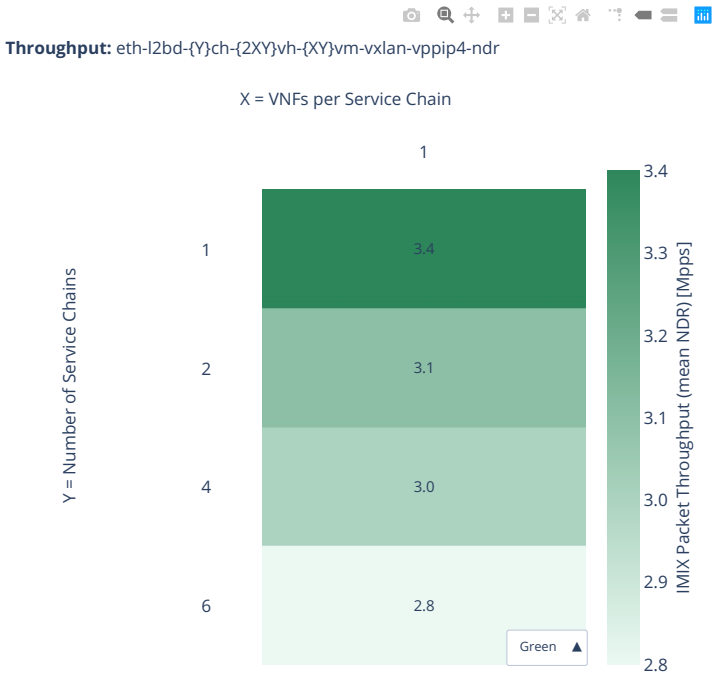


imix-8t4c-eth-l2bd

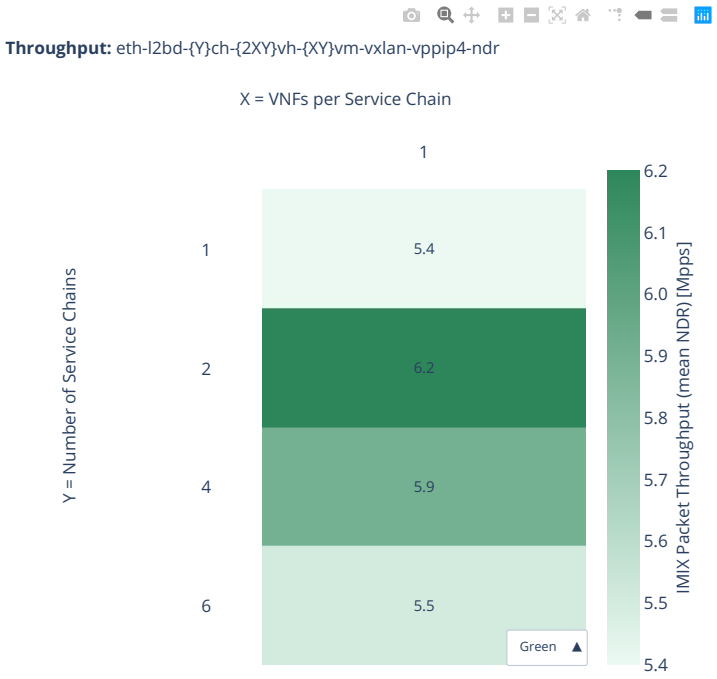


2n-icx-xxv710-ndr

imix-2t1c-eth-l2bd



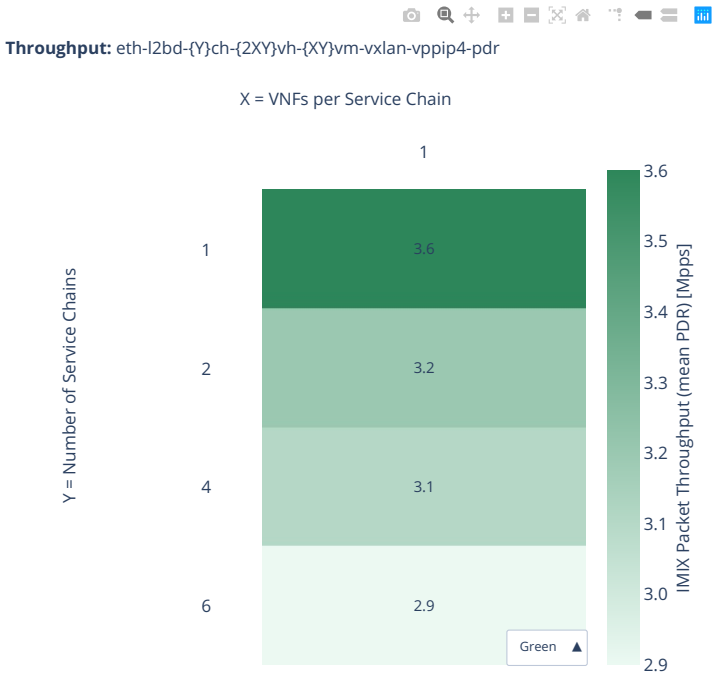
imix-4t2c-eth-l2bd



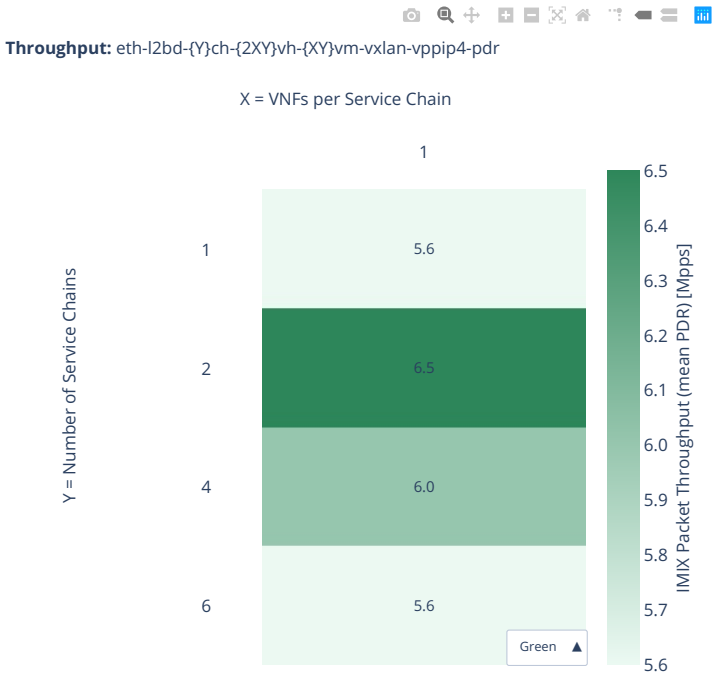
imix-8t4c-eth-l2bd

2n-icx-xxv710-pdr

imix-2t1c-eth-l2bd



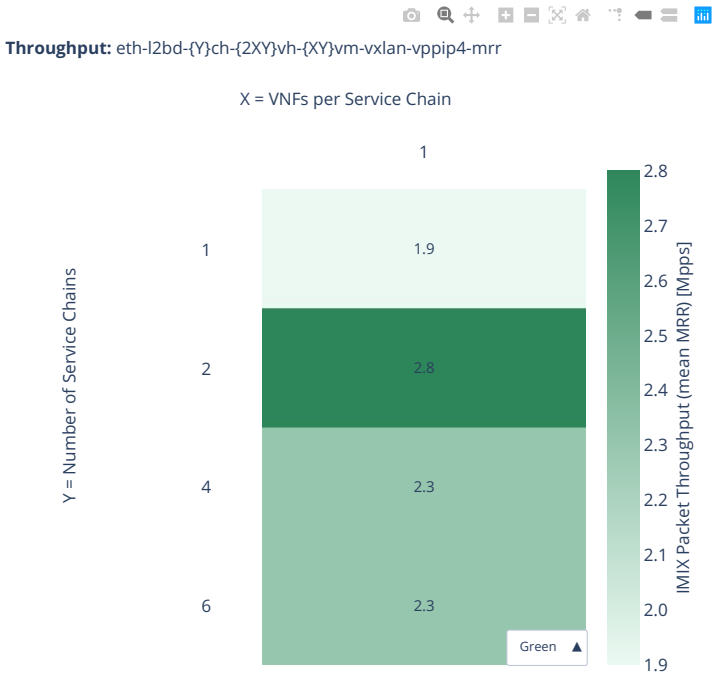
imix-4t2c-eth-l2bd



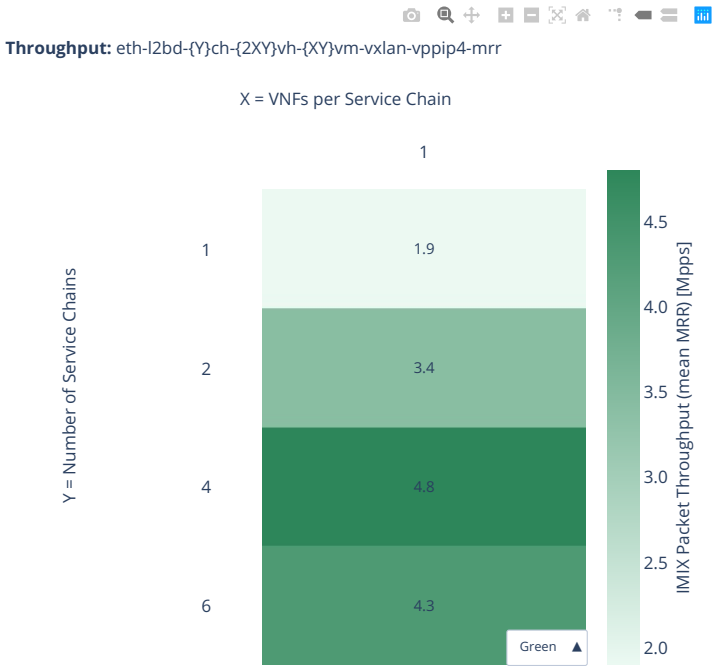
imix-8t4c-eth-l2bd

2n-clx-xxv710-mrr

imix-2t1c-eth-l2bd



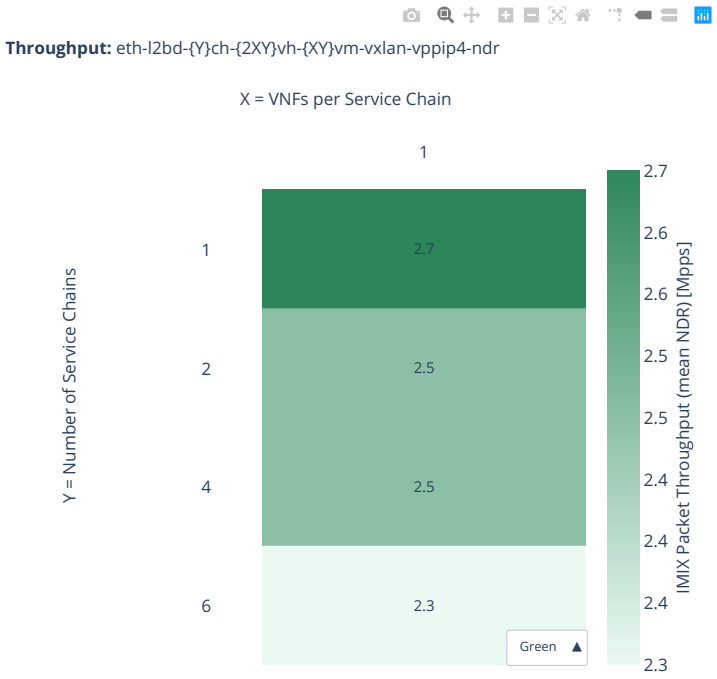
imix-4t2c-eth-l2bd



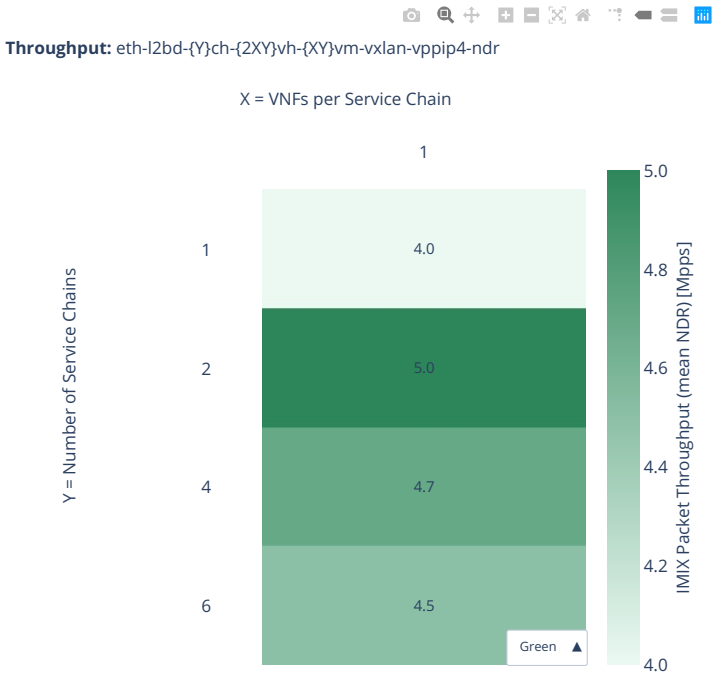
imix-8t4c-eth-l2bd

2n-clx-xxv710-ndr

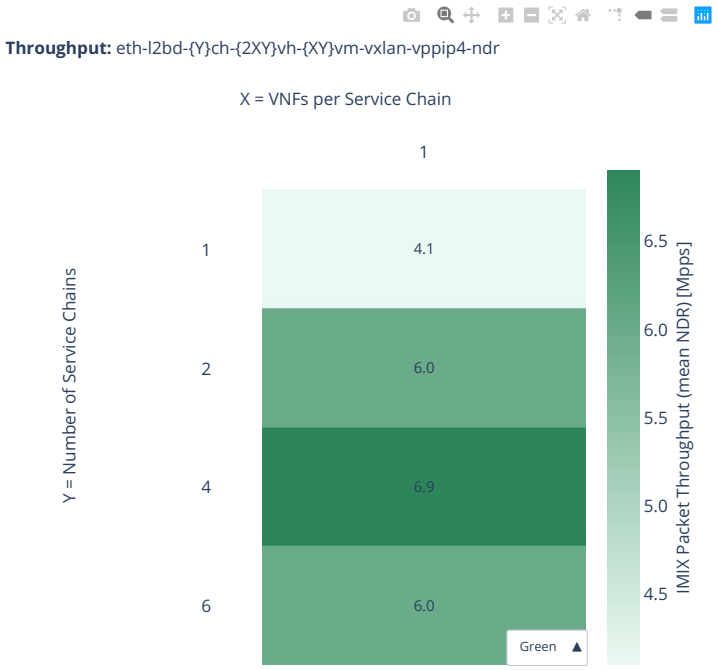
imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

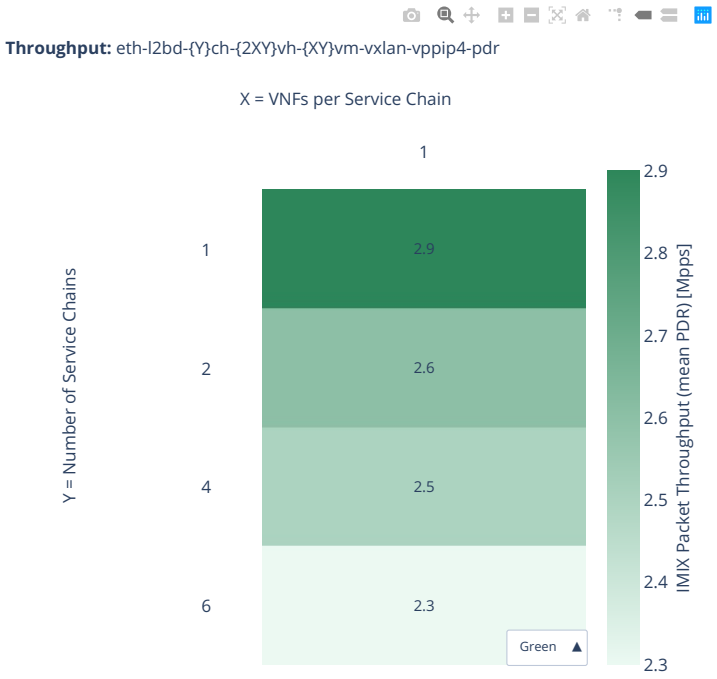


imix-8t4c-eth-l2bd

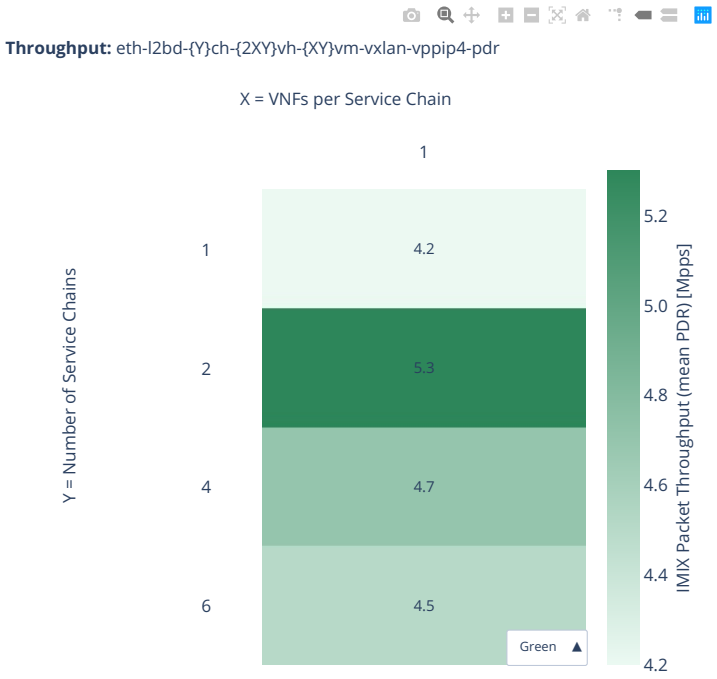


2n-clx-xxv710-pdr

imix-2t1c-eth-l2bd



imix-4t2c-eth-l2bd

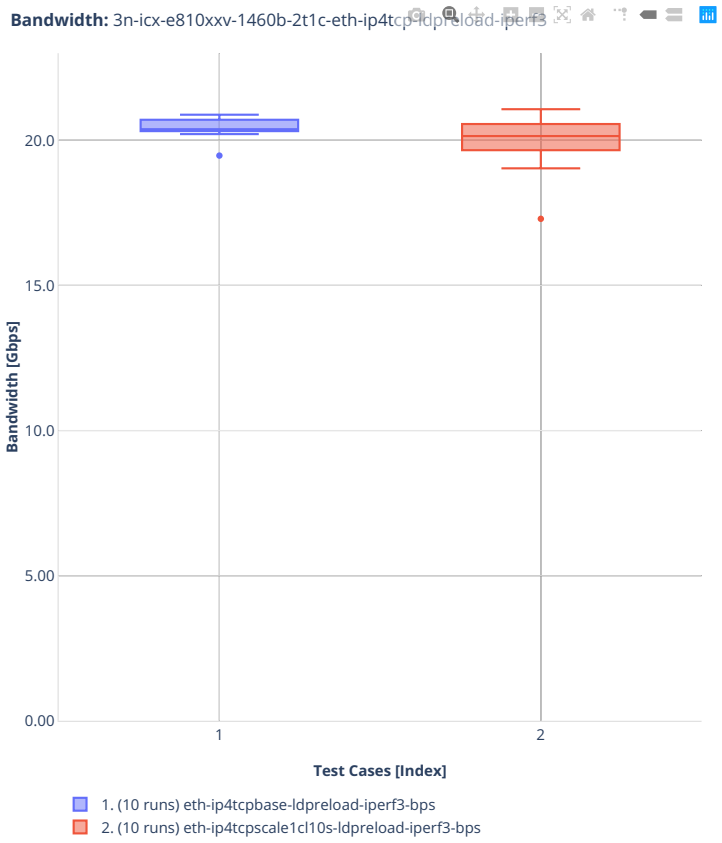


imix-8t4c-eth-l2bd

2.9 Hoststack Testing

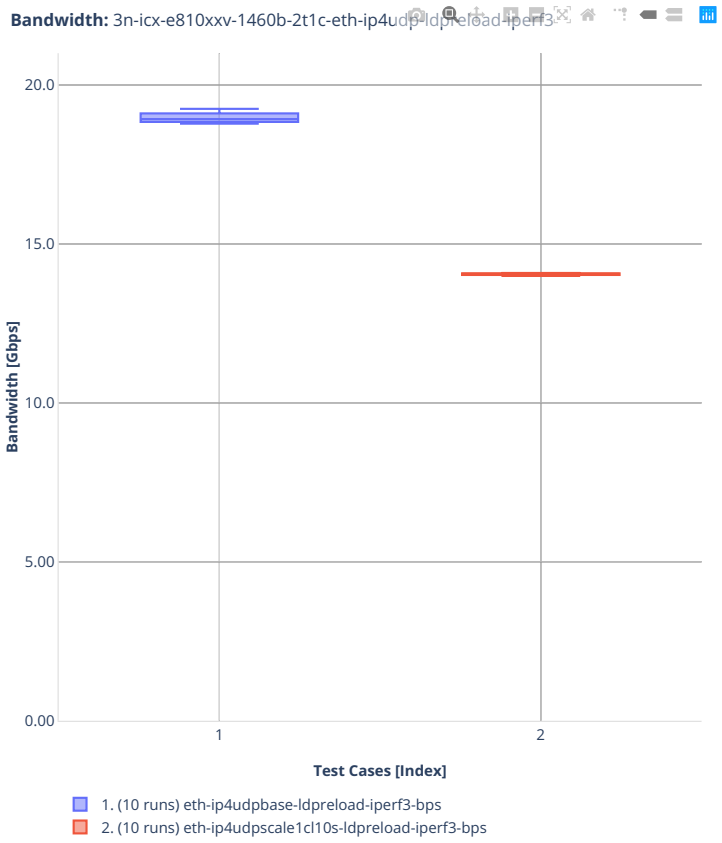
2.9.1 TCP/IP with iperf3

1460b-2t1c-e810xxv-ip4tcp-base-scale



2.9.2 UDP/IP with iperf3

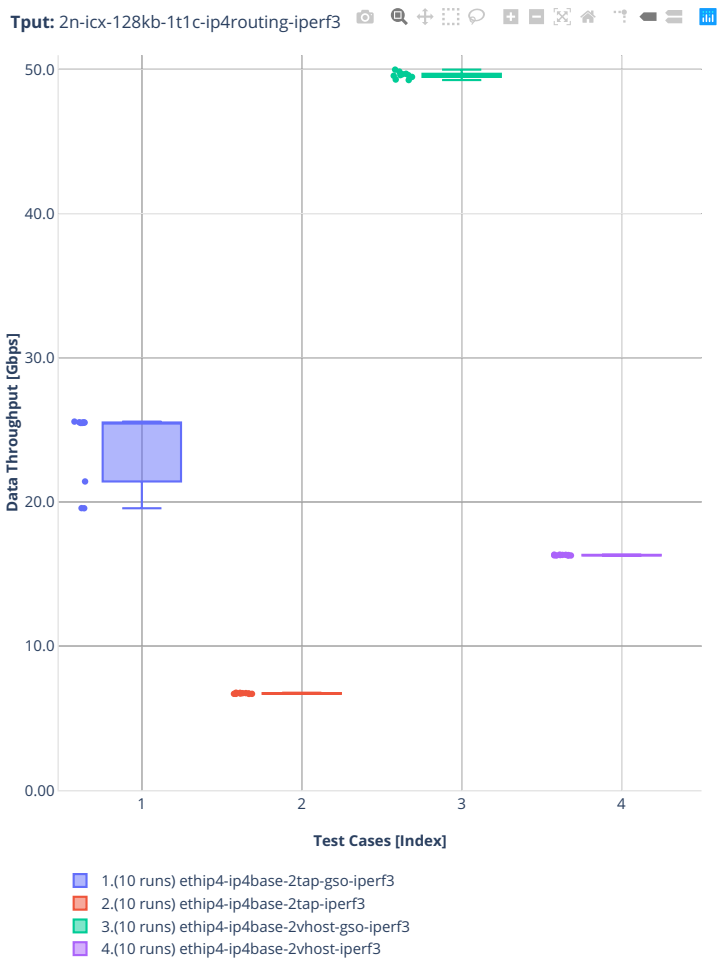
1460b-2t1c-e810xxv-ip4udp-base-scale



2.10 GSO Testing

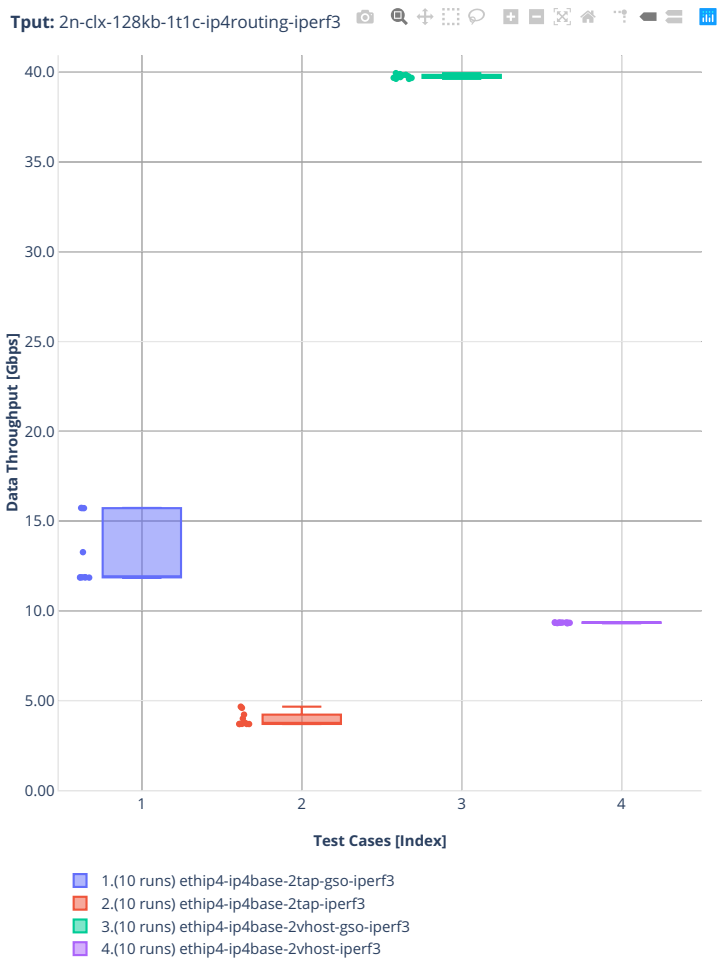
2.10.1 2n-icx

1t1c


Tput: 2n-icx-128kb-1t1c-ip4routing-iperf3
50.0
40.0
30.0
20.0
10.0
0.00
Data Throughput [Gbps]
1
2
3
4
Test Cases [Index]
1.(10 runs) ethip4-ip4base-2tap-gso-iperf3
2.(10 runs) ethip4-ip4base-2tap-iperf3
3.(10 runs) ethip4-ip4base-2vhost-gso-iperf3
4.(10 runs) ethip4-ip4base-2vhost-iperf3

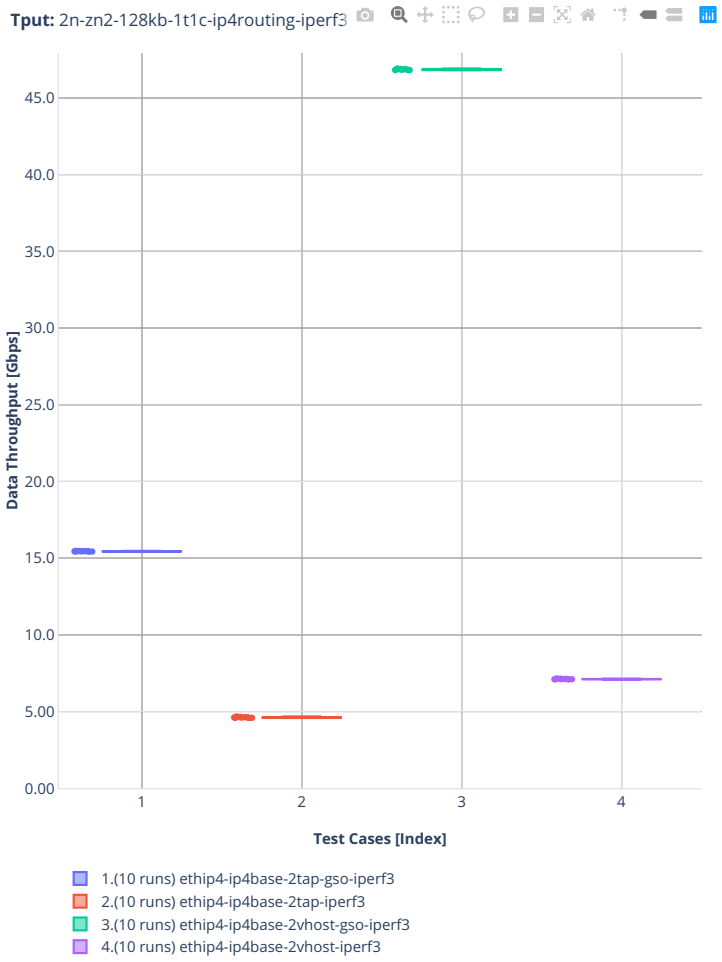
2.10.2 2n-clx

1t1c


Tput: 2n-clx-128kb-1t1c-ip4routing-iperf3
Data Throughput [Gbps]
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) ethip4-ip4base-2tap-gso-iperf3
2.(10 runs) ethip4-ip4base-2tap-iperf3
3.(10 runs) ethip4-ip4base-2vhost-gso-iperf3
4.(10 runs) ethip4-ip4base-2vhost-iperf3

2.10.3 2n-zn2

1t1c


Tput: 2n-zn2-128kb-1t1c-ip4routing-iperf3
Data Throughput [Gbps]
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
1
2
3
4
Test Cases [Index]
1.(10 runs) ethip4-ip4base-2tap-gso-iperf3
2.(10 runs) ethip4-ip4base-2tap-iperf3
3.(10 runs) ethip4-ip4base-2vhost-gso-iperf3
4.(10 runs) ethip4-ip4base-2vhost-iperf3

2.11 Comparisons

2.11.1 Current vs Previous Release

Relative comparison of VPP packet throughput (NDR, PDR and MRR) between VPP-22.10 release and VPP-22.06 release (measured for CSIT-2210 and CSIT-2206 respectively) is calculated from results of tests running on 2-node Intel Atom Denverton (2n-dnv), 3-node Intel Atom Denverton (3n-dnv), 3-node Arm TaiShan (3n-tsh) testbeds, in 1-core, 2-core and 4-core (MRR only) configurations.

Listed mean and standard deviation values are computed based on a series of the same tests executed against respective VPP releases to verify test results repeatability, with percentage change calculated for mean values. Note that the standard deviation is quite high for a small number of packet throughput tests, what indicates poor test results repeatability and makes the relative change of mean throughput value not fully representative for these tests. The root causes behind poor results repeatability vary between the test cases.

Note: Test results are stored in

- build logs from FD.io vpp performance job 2n-icx[171],
- build logs from FD.io vpp performance job 3n-icx[172],
- build logs from FD.io vpp performance job 2n-clx[173],
- build logs from FD.io vpp performance job 2n-zn2[174],
- build logs from FD.io vpp performance job 2n-dnv[175],
- build logs from FD.io vpp performance job 3n-dnv[176],
- build logs from FD.io vpp performance job 3n-tsh[177],
- build logs from FD.io vpp performance job 2n-tx2[178],
- build logs from FD.io vpp performance job 2n-aws[179],

with RF result files csit-vpp-perf-2210-*.zip archived here.

2n-icx-xxv710

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c NDR comparison
- HTML 4t2c NDR comparison
- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

[171] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[172] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-icx
[173] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx
[174] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-zn2
[175] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-dnv
[176] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-dnv
[177] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-3n-tsh
[178] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-tx2
[179] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-aws

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR comparison
- HTML 4t2c PDR comparison
- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c MRR comparison
- HTML 4t2c MRR comparison
- HTML 8t4c MRR comparison
- ASCII 2t1c MRR comparison
- ASCII 4t2c MRR comparison
- ASCII 8t4c MRR comparison
- CSV 2t1c MRR comparison
- CSV 4t2c MRR comparison
- CSV 8t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR50, direction1, average value comparison
- HTML 2t1c PDR90, direction1, average value comparison
- HTML 2t1c PDR90, direction1, max value comparison
- ASCII 2t1c PDR50, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, max value comparison
- CSV 2t1c PDR50, direction1, average value comparison
- CSV 2t1c PDR90, direction1, average value comparison
- CSV 2t1c PDR90, direction1, max value comparison

3n-icx-xxv710

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c NDR comparison
- HTML 4t2c NDR comparison
- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR comparison
- HTML 4t2c PDR comparison
- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c MRR comparison
- HTML 4t2c MRR comparison
- HTML 8t4c MRR comparison
- ASCII 2t1c MRR comparison
- ASCII 4t2c MRR comparison
- ASCII 8t4c MRR comparison
- CSV 2t1c MRR comparison
- CSV 4t2c MRR comparison
- CSV 8t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR50, direction1, average value comparison
- HTML 2t1c PDR90, direction1, average value comparison
- HTML 2t1c PDR90, direction1, max value comparison
- ASCII 2t1c PDR50, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, max value comparison
- CSV 2t1c PDR50, direction1, average value comparison
- CSV 2t1c PDR90, direction1, average value comparison
- CSV 2t1c PDR90, direction1, max value comparison

2n-clx-xxv710

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c NDR comparison
- HTML 4t2c NDR comparison
- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR comparison
- HTML 4t2c PDR comparison
- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c MRR comparison
- HTML 4t2c MRR comparison
- HTML 8t4c MRR comparison
- ASCII 2t1c MRR comparison
- ASCII 4t2c MRR comparison
- ASCII 8t4c MRR comparison
- CSV 2t1c MRR comparison
- CSV 4t2c MRR comparison
- CSV 8t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR50, direction1, average value comparison
- HTML 2t1c PDR90, direction1, average value comparison
- HTML 2t1c PDR90, direction1, max value comparison
- ASCII 2t1c PDR50, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, max value comparison
- CSV 2t1c PDR50, direction1, average value comparison
- CSV 2t1c PDR90, direction1, average value comparison
- CSV 2t1c PDR90, direction1, max value comparison

2n-clx-cx556a

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c NDR comparison
- HTML 4t2c NDR comparison
- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR comparison
- HTML 4t2c PDR comparison
- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c MRR comparison
- HTML 4t2c MRR comparison
- HTML 8t4c MRR comparison
- ASCII 2t1c MRR comparison
- ASCII 4t2c MRR comparison
- ASCII 8t4c MRR comparison
- CSV 2t1c MRR comparison
- CSV 4t2c MRR comparison
- CSV 8t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR50, direction1, average value comparison
- HTML 2t1c PDR90, direction1, average value comparison
- HTML 2t1c PDR90, direction1, max value comparison
- ASCII 2t1c PDR50, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, max value comparison
- CSV 2t1c PDR50, direction1, average value comparison
- CSV 2t1c PDR90, direction1, average value comparison
- CSV 2t1c PDR90, direction1, max value comparison

2n-zn2-xxv710

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c NDR comparison
- HTML 4t2c NDR comparison
- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR comparison
- HTML 4t2c PDR comparison
- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c MRR comparison
- HTML 4t2c MRR comparison
- HTML 8t4c MRR comparison
- ASCII 2t1c MRR comparison
- ASCII 4t2c MRR comparison
- ASCII 8t4c MRR comparison
- CSV 2t1c MRR comparison
- CSV 4t2c MRR comparison
- CSV 8t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR50, direction1, average value comparison
- HTML 2t1c PDR90, direction1, average value comparison
- HTML 2t1c PDR90, direction1, max value comparison
- ASCII 2t1c PDR50, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, average value comparison
- ASCII 2t1c PDR90, direction1, max value comparison
- CSV 2t1c PDR50, direction1, average value comparison
- CSV 2t1c PDR90, direction1, average value comparison
- CSV 2t1c PDR90, direction1, max value comparison

2n-dnv-x553

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c NDR comparison
- HTML 2t2c NDR comparison
- ASCII 1t1c NDR comparison
- ASCII 2t2c NDR comparison
- CSV 1t1c NDR comparison
- CSV 2t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR comparison
- HTML 2t2c PDR comparison
- ASCII 1t1c PDR comparison
- ASCII 2t2c PDR comparison
- CSV 1t1c PDR comparison
- CSV 2t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c MRR comparison
- HTML 2t2c MRR comparison
- HTML 4t4c MRR comparison
- ASCII 1t1c MRR comparison
- ASCII 2t2c MRR comparison
- ASCII 4t4c MRR comparison
- CSV 1t1c MRR comparison
- CSV 2t2c MRR comparison
- CSV 4t4c MRR comparison

3n-dnv-x553

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c NDR comparison
- HTML 2t2c NDR comparison
- ASCII 1t1c NDR comparison
- ASCII 2t2c NDR comparison
- CSV 1t1c NDR comparison
- CSV 2t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR comparison
- HTML 2t2c PDR comparison
- ASCII 1t1c PDR comparison
- ASCII 2t2c PDR comparison
- CSV 1t1c PDR comparison
- CSV 2t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c MRR comparison
- HTML 2t2c MRR comparison
- HTML 4t4c MRR comparison
- ASCII 1t1c MRR comparison
- ASCII 2t2c MRR comparison
- ASCII 4t4c MRR comparison
- CSV 1t1c MRR comparison
- CSV 2t2c MRR comparison
- CSV 4t4c MRR comparison

3n-tsh-x520

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c NDR comparison
- HTML 2t2c NDR comparison
- ASCII 1t1c NDR comparison
- ASCII 2t2c NDR comparison
- CSV 1t1c NDR comparison
- CSV 2t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR comparison
- HTML 2t2c PDR comparison
- ASCII 1t1c PDR comparison
- ASCII 2t2c PDR comparison
- CSV 1t1c PDR comparison
- CSV 2t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c MRR comparison
- HTML 2t2c MRR comparison
- HTML 4t4c MRR comparison
- ASCII 1t1c MRR comparison
- ASCII 2t2c MRR comparison
- ASCII 4t4c MRR comparison
- CSV 1t1c MRR comparison
- CSV 2t2c MRR comparison
- CSV 4t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR50, direction1, average value comparison
- HTML 1t1c PDR90, direction1, average value comparison
- HTML 1t1c PDR90, direction1, max value comparison
- ASCII 1t1c PDR50, direction1, average value comparison
- ASCII 1t1c PDR90, direction1, average value comparison
- ASCII 1t1c PDR90, direction1, max value comparison
- CSV 1t1c PDR50, direction1, average value comparison
- CSV 1t1c PDR90, direction1, average value comparison
- CSV 1t1c PDR90, direction1, max value comparison

3n-alt-xl710

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c NDR comparison
- HTML 2t2c NDR comparison
- ASCII 1t1c NDR comparison
- ASCII 2t2c NDR comparison
- CSV 1t1c NDR comparison
- CSV 2t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR comparison
- HTML 2t2c PDR comparison
- ASCII 1t1c PDR comparison
- ASCII 2t2c PDR comparison
- CSV 1t1c PDR comparison
- CSV 2t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c MRR comparison
- HTML 2t2c MRR comparison
- HTML 4t4c MRR comparison
- ASCII 1t1c MRR comparison
- ASCII 2t2c MRR comparison
- ASCII 4t4c MRR comparison
- CSV 1t1c MRR comparison
- CSV 2t2c MRR comparison
- CSV 4t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR50, direction1, average value comparison
- HTML 1t1c PDR90, direction1, average value comparison
- HTML 1t1c PDR90, direction1, max value comparison
- ASCII 1t1c PDR50, direction1, average value comparison
- ASCII 1t1c PDR90, direction1, average value comparison
- ASCII 1t1c PDR90, direction1, max value comparison
- CSV 1t1c PDR50, direction1, average value comparison
- CSV 1t1c PDR90, direction1, average value comparison
- CSV 1t1c PDR90, direction1, max value comparison

2n-tx2-xl710

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c NDR comparison
- HTML 2t2c NDR comparison
- ASCII 1t1c NDR comparison
- ASCII 2t2c NDR comparison
- CSV 1t1c NDR comparison
- CSV 2t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR comparison
- HTML 2t2c PDR comparison
- ASCII 1t1c PDR comparison
- ASCII 2t2c PDR comparison
- CSV 1t1c PDR comparison
- CSV 2t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c MRR comparison
- HTML 2t2c MRR comparison
- HTML 4t4c MRR comparison
- ASCII 1t1c MRR comparison
- ASCII 2t2c MRR comparison
- ASCII 4t4c MRR comparison
- CSV 1t1c MRR comparison
- CSV 2t2c MRR comparison
- CSV 4t4c MRR comparison

Latency Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1t1c PDR50, direction1, average value comparison
- HTML 1t1c PDR90, direction1, average value comparison
- HTML 1t1c PDR90, direction1, max value comparison
- ASCII 1t1c PDR50, direction1, average value comparison
- ASCII 1t1c PDR90, direction1, average value comparison
- ASCII 1t1c PDR90, direction1, max value comparison
- CSV 1t1c PDR50, direction1, average value comparison
- CSV 1t1c PDR90, direction1, average value comparison
- CSV 1t1c PDR90, direction1, max value comparison

2n-aws-nitro50g

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c NDR comparison
- HTML 4t2c NDR comparison
- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c PDR comparison
- HTML 4t2c PDR comparison
- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 2t1c MRR comparison
- HTML 4t2c MRR comparison
- ASCII 2t1c MRR comparison
- ASCII 4t2c MRR comparison
- CSV 2t1c MRR comparison
- CSV 4t2c MRR comparison

2.11.2 2n-Icx vs 2n-Clx Testbeds

Relative comparison of VPP-22.10 release packet throughput (NDR, PDR and MRR) is calculated for the same tests executed on 2-Node Cascadelake (2n- clx) and 2-Node Icelake (2n-icx) physical testbed types, in 1-core, 2-core and 4-core configurations.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[180] and build logs from FD.io vpp performance job 2n-clx[181] with RF result files csit-vpp-perf-2210-*.zip archived here.

NDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1c NDR comparison
- HTML 2c NDR comparison
- ASCII 1c NDR comparison
- ASCII 2c NDR comparison
- CSV 1c NDR comparison
- CSV 2c NDR comparison

PDR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1c PDR comparison
- HTML 2c PDR comparison
- ASCII 1c PDR comparison
- ASCII 2c PDR comparison
- CSV 1c PDR comparison
- CSV 2c PDR comparison

[180] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[181] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx

MRR Comparison

Comparison tables in HTML, ASCII and CSV formats:

- HTML 1c MRR comparison
- HTML 2c MRR comparison
- HTML 4c MRR comparison
- ASCII 1c MRR comparison
- ASCII 2c MRR comparison
- ASCII 4c MRR comparison
- CSV 1c MRR comparison
- CSV 2c MRR comparison
- CSV 4c MRR comparison

2.11.3 Soak Tests vs NDR Tests

Relative comparison of VPP-22.10 release Soak PLRSearch vs NDR packet throughput is calculated for the tests executed on 2-Node Skylake physical testbed types, in 1-core configurations.

Note: Test results are stored in build logs from FD.io vpp performance job 2n-icx[182], build logs from FD.io vpp performance job 2n-clx[183] with RF result files csit-vpp-perf-2210-*.zip archived here.

Comparison tables in ASCII and CSV formats:

2n-icx

- ASCII Soak vs NDR comparison
- CSV Soak vs NDR comparison

2n-clx

- ASCII Soak vs NDR comparison
- CSV Soak vs NDR comparison

2.12 Throughput Trending

In addition to reporting throughput comparison between VPP releases, CSIT provides continuous performance trending for VPP master branch: C-Dash[184]

[182] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-icx
[183] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-report-iterative-2210-2n-clx
[184] http://csit.fd.io/trending/

2.13 Test Environment

2.13.1 Environment Versioning

CSIT test environment versioning has been introduced to track modifications of the test environment.

Any benchmark anomalies (progressions, regressions) between releases of a DUT application (e.g. VPP, DPDK), are determined by testing it in the same test environment, to avoid test environment changes clouding the picture. To beter distinguish impact of test environment changes, we also execute tests without any SUT (just with TRex TG sending packets over a link looping back to TG).

A mirror approach is introduced to determine benchmarking anomalies due to the test environment change. This is achieved by testing the same DUT application version between releases of CSIT test system. This works under the assumption that the behaviour of the DUT is deterministic under the test conditions.

CSIT test environment versioning scheme ensures integrity of all the test system components, including their HW revisions, compiled SW code versions and SW source code, within a specific CSIT version. Components included in the CSIT environment versioning include:

- **HW** Server hardware firmware and BIOS (motherboard, processsor, NIC(s), accelerator card(s)), tracked in CSIT branch.
- **Linux** Server Linux OS version and configuration, tracked in CSIT Reports.
- **TRex** TRex Traffic Generator version, drivers and configuration tracked in TG Settings.
- **CSIT** CSIT framework code tracked in CSIT release branches.

Following is the list of CSIT versions to date:

- Ver. 1 associated with CSIT rls1908 branch (HW[185], Linux[186], TRex[187], CSIT[188]).
- Ver. 2 associated with CSIT rls2001 branch (HW[189], Linux[190], TRex[191], CSIT[192]).
- Ver. 4 associated with CSIT rls2005 branch (HW[193], Linux[194], TRex[195], CSIT[196]).
- Ver. 5 associated with CSIT rls2009 branch (HW[197], Linux[198], TRex[199], CSIT[200]).
  - The main change is TRex data-plane core resource adjustments: increase from 7 to 8 cores and pinning cores to interfaces[201] for better TRex performance with symmetric traffic profiles.
- Ver. 6 associated with CSIT rls2101 branch (HW[202], Linux[203], TRex[204], CSIT[205]).
  - The main change is TRex version upgrade: increase from 2.82 to 2.86.

[185] https://git.fd.io/csit/tree/docs/lab?h=rls1908
[186] https://docs.fd.io/csit/rls1908/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[187] https://docs.fd.io/csit/rls1908/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[188] https://git.fd.io/csit/tree/?h=rls1908
[189] https://git.fd.io/csit/tree/docs/lab?h=rls2001
[190] https://docs.fd.io/csit/rls2001/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[191] https://docs.fd.io/csit/rls2001/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[192] https://git.fd.io/csit/tree/?h=rls2001
[193] https://git.fd.io/csit/tree/docs/lab?h=rls2005
[194] https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[195] https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[196] https://git.fd.io/csit/tree/?h=rls2005
[197] https://git.fd.io/csit/tree/docs/lab?h=rls2009
[198] https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[199] https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[200] https://git.fd.io/csit/tree/?h=rls2009
[201] https://gerrit.fd.io/r/c/csit/+/28184
[202] https://git.fd.io/csit/tree/docs/lab?h=rls2101
[203] https://docs.fd.io/csit/rls2101/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[204] https://docs.fd.io/csit/rls2101/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[205] https://git.fd.io/csit/tree/?h=rls2101

- Ver. 7 associated with CSIT rls2106 branch (HW[206], Linux[207], TRex[208], CSIT[209]).
  - TRex version upgrade: increase from 2.86 to 2.88.
  - Ubuntu upgrade from 18.04 LTS to 20.04.2 LTS.
- Ver. 8 associated with CSIT rls2110 branch (HW[210], Linux[211], TRex[212], CSIT[213]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
- Ver. 9 associated with CSIT rls2202 branch (HW[214], Linux[215], TRex[216], CSIT[217]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
- Ver. 10 associated with CSIT rls2206 branch (HW[218], Linux[219], TRex[220], CSIT[221]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
  - Mellanox 556A series firmware upgrade based on DPDK compatibility matrix.
  - Intel IceLake all core turbo frequency turned off. Current base frequency is 2.6GHz.
  - TRex version upgrade: increase from 2.88 to 2.97.
- Ver. 11 associated with CSIT rls2210 branch (HW[222], Linux[223], TRex[224], CSIT[225]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
  - Mellanox 556A series firmware upgrade based on DPDK compatibility matrix.
  - Ubuntu upgrade from 20.04.2 LTS to 22.04.1 LTS. (2n-dnv and 3n-dnv keeps the Ubuntu 20.04.2LTS as a part of decomission).
  - TRex version upgrade: increase from 2.97 to 3.00.

To identify performance changes due to VPP code development between previous and current VPP release version, both have been tested in CSIT environment of latest version and compared against each other. All substantial progressions and regressions have been marked up with RCA analysis. See *Comparisons* (page 1190) and *Known Issues* (page 87).

[206] https://git.fd.io/csit/tree/docs/lab?h=rls2106
[207] https://s3-docs.fd.io/csit/rls2106/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[208] https://s3-docs.fd.io/csit/rls2106/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[209] https://git.fd.io/csit/tree/?h=rls2106
[210] https://git.fd.io/csit/tree/docs/lab?h=rls2110
[211] https://s3-docs.fd.io/csit/rls2110/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[212] https://s3-docs.fd.io/csit/rls2110/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[213] https://git.fd.io/csit/tree/?h=rls2110
[214] https://git.fd.io/csit/tree/docs/lab?h=rls2202
[215] https://s3-docs.fd.io/csit/rls2202/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[216] https://s3-docs.fd.io/csit/rls2202/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[217] https://git.fd.io/csit/tree/?h=rls2202
[218] https://git.fd.io/csit/tree/docs/lab?h=rls2206
[219] https://s3-docs.fd.io/csit/rls2206/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[220] https://s3-docs.fd.io/csit/rls2206/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[221] https://git.fd.io/csit/tree/?h=rls2206
[222] https://git.fd.io/csit/tree/docs/lab?h=rls2210
[223] https://s3-docs.fd.io/csit/rls2210/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[224] https://s3-docs.fd.io/csit/rls2210/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[225] https://git.fd.io/csit/tree/?h=rls2210

2.13.2 Physical Testbeds

FD.io CSIT performance tests are executed in physical testbeds hosted by LF for FD.io project. Two physical testbed topology types are used:

- **3-Node Topology**: Consisting of two servers acting as SUTs (Systems Under Test) and one server as TG (Traffic Generator), all connected in ring topology.
- **2-Node Topology**: Consisting of one server acting as SUTs and one server as TG both connected in ring topology.

Tested SUT servers are based on a range of processors. More detailed description is provided in *Performance Physical Testbeds* (page 4). Tested logical topologies are described in *Logical Topologies* (page 79).

2.13.3 Server Specifications

Complete technical specifications of compute servers used in CSIT physical testbeds are maintained in FD.io CSIT repository: FD.io CSIT testbeds - Xeon Cascade Lake[226], FD.io CSIT testbeds - Xeon Ice Lake[227], FD.io CSIT testbeds - EPYC Zen2[228], FD.io CSIT testbeds - Atom Denverton[229], FD.io CSIT testbeds - Atom Snowridge[230].

2.13.4 SUT Settings - Linux

System provisioning is done by combination of PXE boot unattented install and Ansible[231] described in CSIT Testbed Setup[232].

Linux Boot Parameters

- **isolcpus=<cpu number>-<cpu number>** used for all cpu cores apart from first core of each socket used for running VPP worker threads and Qemu/LXC processes https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt
- **intel_pstate=disable** - [X86] Do not enable intel_pstate as the default scaling driver for the supported processors. Intel P-State driver decide what P-state (CPU core power state) to use based on requesting policy from the cpufreq core. [X86 - Either 32-bit or 64-bit x86] https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt
- **nohz_full=<cpu number>-<cpu number>** - [KNL,BOOT] In kernels built with CONFIG_NO_HZ_FULL=y, set the specified list of CPUs whose tick will be stopped whenever possible. The boot CPU will be forced outside the range to maintain the timekeeping. The CPUs in this range must also be included in the rcu_nocbs= set. Specifies the adaptive-ticks CPU cores, causing kernel to avoid sending scheduling-clock interrupts to listed cores as long as they have a single runnable task. [KNL - Is a kernel start-up parameter, SMP - The kernel is an SMP kernel]. https://www.kernel.org/doc/Documentation/timers/NO_HZ.txt
- **rcu_nocbs** - [KNL] In kernels built with CONFIG_RCU_NOCB_CPU=y, set the specified list of CPUs to be no-callback CPUs, that never queue RCU callbacks (read-copy update). https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt
- **numa_balancing=disable** - [KNL,X86] Disable automatic NUMA balancing.
- **intel_iommu=enable** - [DMAR] Enable Intel IOMMU driver (DMAR) option.

[226] https://git.fd.io/csit/tree/docs/lab/testbeds_sm_clx_hw_bios_cfg.md?h=rls2210
[227] https://git.fd.io/csit/tree/docs/lab/testbeds_sm_icx_hw_bios_cfg.md?h=rls2210
[228] https://git.fd.io/csit/tree/docs/lab/testbeds_sm_zn2_hw_bios_cfg.md?h=rls2210
[229] https://git.fd.io/csit/tree/docs/lab/testbeds_sm_dnv_hw_bios_cfg.md?h=rls2210
[230] https://git.fd.io/csit/tree/docs/lab/testbeds_sm_snr_hw_bios_cfg.md?h=rls2210
[231] https://www.ansible.com
[232] https://git.fd.io/csit/tree/fdio.infra.ansible?h=rls2210

- **iommu=on, iommu=pt** - [x86, IA-64] Disable IOMMU bypass, using IOMMU for PCI devices.
- **nmi_watchdog=0** - [KNL,BUGS=X86] Debugging features for SMP kernels. Turn hardlockup detector in nmi_watchdog off.
- **nosoftlockup** - [KNL] Disable the soft-lockup detector.
- **tsc=reliable** - Disable clocksource stability checks for TSC. [x86] reliable: mark tsc clocksource as reliable, this disables clocksource verification at runtime, as well as the stability checks done at bootup. Used to enable high-resolution timer mode on older hardware, and in virtualized environment.
- **hpet=disable** - [X86-32,HPET] Disable HPET and use PIT instead.

Hugepages Configuration

Huge pages are managed via sysctl configuration located in */etc/sysctl.d/90-csit.conf* on each testbed. Default huge page size is 2M. The exact amount of huge pages depends on testbed. All the values are defined in *Ansible inventory - hosts* files.

2.13.5 DUT Settings - VPP

VPP Version

VPP-22.10 release

VPP Compile Parameters

FD.io VPP compile job[233]

VPP Install Parameters

```
$ dpkg -i --force-all *vpp*
```

VPP Startup Configuration

VPP startup configuration vary per test case, with different settings for *$$CORELIST_WORKERS*, *$$NUM_RX_QUEUES*, *$$UIO_DRIVER*, and *$$NO_MULTI_SEG* parameter. List of plugins to enable is driven by test requirements. Default template is provided below:

```
ip
{
  heap-size 4G
}
statseg
{
  size 4G
  per-node-counters on
}
unix
{
  cli-listen /run/vpp/cli.sock
  log /tmp/vpe.log
```

(continues on next page)

[233] https://jenkins.fd.io/view/vpp/job/vpp-merge-2210-ubuntu2004-x86_64/

(continued from previous page)

```
  nodaemon
  full-coredump
}
socksvr {
  socket-name /run/vpp/api.sock
}
ip6
{
  heap-size 4G
  hash-buckets 2000000
}
heapsize 4G
plugins
{
  plugin default
  {
    disable
  }
  plugin <$$test_requirement>_plugin.so
  {
    enable
  }
}
cpu
{
  corelist-workers $$CORELIST_WORKERS
  main-core 1
}
buffers
{
  buffers-per-numa 215040
}

# Below: in case of dpdk based drivers (vfio-pci) only
dpdk
{
  uio-driver $$UIO_DRIVER
  $$NO_MULTI_SEG
  log-level debug
  dev default
  {
    num-rx-queues $$NUM_RX_QUEUES
  }
  no-tx-checksum-offload
  dev $$DEV_1
  dev $$DEV_2
}
```

Description of VPP startup settings used in CSIT is provided in *Test Methodology* (page 18).

2.13.6 TG Settings - TRex

TG Version

TRex v3.00

DPDK Version

DPDK v21.02

TG Installation

T-Rex installation is managed via Ansible role.

TG Startup Configuration

```
$ sudo -E -S sh -c 'cat << EOF > /etc/trex_cfg.yaml
- version: 2
  c: 8
  limit_memory: 8192
  interfaces: ["${pci1}","${pci2}"]
  port_info:
      - dest_mac: [${dest_mac1}]
        src_mac: [${src_mac1}]
      - dest_mac: [${dest_mac2}]
        src_mac: [${src_mac2}]
  platform :
      master_thread_id: 0
      latency_thread_id: 9
      dual_if:
          - socket: 0
            threads: [1, 2, 3, 4, 5, 6, 7, 8]
EOF'
```

TG Startup Command (Stateless Mode)

```
$ sudo -E -S sh -c "cd '${trex_install_dir}/scripts/' && \
  nohup ./t-rex-64 -i --prefix $(hostname) --hdrh --no-scapy-server \
  --mbuf-factor 32 > /tmp/trex.log 2>&1 &" > /dev/null
```

Also, Python client is now starting traffic with:

```
core_mask=STLClient.CORE_MASK_PIN
```

TG Startup Command (Stateful Mode)

```
$ sudo -E -S sh -c "cd '${trex_install_dir}/scripts/' && \
  nohup ./t-rex-64 -i --prefix $(hostname) --astf --hdrh --no-scapy-server \
  --mbuf-factor 32 > /tmp/trex.log 2>&1 &" > /dev/null
```

TG API Driver

TRex driver[234]

2.13.7 Pre-Test Server Calibration

Number of SUT server sub-system runtime parameters have been identified as impacting data plane performance tests. Calibrating those parameters is part of FD.io CSIT pre-test activities, and includes measuring and reporting following:

1. System level core jitter - measure duration of core interrupts by Linux in clock cycles and how often interrupts happen. Using CPU core jitter tool[235].
2. Memory bandwidth - measure bandwidth with Intel MLC tool[236].
3. Memory latency - measure memory latency with Intel MLC tool.
4. Cache latency at all levels (L1, L2, and Last Level Cache) - measure cache latency with Intel MLC tool.

Measured values of listed parameters are especially important for repeatable zero packet loss throughput measurements across multiple system instances. Generally they come useful as a background data for comparing data plane performance results across disparate servers.

Following sections include measured calibration data for testbeds.

Ice Lake

Following sections include sample calibration data measured on server running in one of the Intel Xeon Ice Lake testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=6ff26c8a-8c65-4025-a6e7-d97dee6025d0␣
↪ro audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_
↪pstate=disable iommu=pt isolcpus=1-31,33-63,65-95,97-127 mce=off nmi_watchdog=0 nohz_
↪full=1-31,33-63,65-95,97-127 nosoftlockup numa_balancing=disable processor.max_cstate=1␣
↪rcu_nocbs=1-31,33-63,65-95,97-127 tsc=reliable console=ttyS0,115200n8 quiet
```

[234] https://git.fd.io/csit/tree/GPL/tools/trex/trex_stl_profile.py?h=rls2210
[235] https://git.fd.io/pma_tools/tree/jitter
[236] https://software.intel.com/en-us/articles/intelr-memory-latency-checker

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 3 /home/testuser/pma_tools/jitter/jitter -i 30
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
160022,167912,7890,160034,160022,167912,854327296,3203987030,1
160022,168114,8092,160042,160022,168114,4234936320,3204004240,2
160022,168386,8364,160040,160022,168386,3320578048,3204007496,3
160022,169432,9410,160028,160022,169432,2406219776,3204213462,4
160022,168050,8028,160040,160022,169432,1491861504,3203982428,5
160022,166384,6362,160040,160022,169432,577503232,3203969006,6
160022,168962,8940,160042,160022,169432,3958112256,3204002514,7
160020,169248,9228,160038,160020,169432,3043753984,3204208318,8
160022,168854,8832,160038,160020,169432,2129395712,3203987894,9
160022,166754,6732,160042,160020,169432,1215037440,3203984104,10
160022,168208,8186,160040,160020,169432,300679168,3203980640,11
160022,172450,12428,160040,160020,172450,3681288192,3204208216,12
160022,168244,8222,160042,160020,172450,2766929920,3204037074,13
160022,166894,6872,160040,160020,172450,1852571648,3203979376,14
160022,169068,9046,160038,160020,172450,938213376,3204009714,15
160020,168528,8508,160036,160020,172450,23855104,3204028382,16
160022,169458,9436,160042,160020,172450,3404464128,3204179220,17
160020,167056,7036,160040,160020,172450,2490105856,3203990218,18
160022,167038,7016,160038,160020,172450,1575747584,3203976712,19
160022,168610,8588,160040,160020,172450,661389312,3204025230,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several speculative execution CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[237].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (Intel SSBD)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: YES (L1D flush feature bit)
  * Microarchitectural Data Sampling
    * VERW instruction is available: YES (MD_CLEAR feature bit)
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: YES
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: YES
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): YES
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): YES
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): YES
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): YES
    * TSX_CTRL MSR indicates TSX RTM is disabled: YES
    * TSX_CTRL MSR indicates TSX CPUID bit is cleared: YES
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): YES
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x6a␣
↪stepping 0x6 ucode 0xd000280 cpuid 0x606a6)
  * CPU microcode is the latest known available version: NO (latest version is 0xd000331␣
↪dated 2021/12/03 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
```

(continues on next page)

[237] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): YES
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): YES
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): YES
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): YES
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Enhanced IBRS, IBPB:␣
↪conditional, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your kernel reported your CPU model as not affected)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Cascade Lake

Following sections include sample calibration data measured on server running in one of the Intel Xeon Skylake testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=2d6f4d44-76b1-4343-bc73-c066a3e95b32␣
↪ro audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_
↪pstate=disable iommu=pt isolcpus=1-23,25-47,49-71,73-95 mce=off nmi_watchdog=0 nohz_
↪full=1-23,25-47,49-71,73-95 nosoftlockup numa_balancing=disable processor.max_cstate=1␣
↪rcu_nocbs=1-23,25-47,49-71,73-95 tsc=reliable console=ttyS0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 3 /home/testuser/pma_tools/jitter/jitter -i 30
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
160026,167568,7542,160032,160026,167568,183238656,3204033176,1
160026,171174,11148,160028,160026,171174,3563847680,3204142488,2
160024,170002,9978,160032,160024,171174,2649489408,3204224288,3
160026,169124,9098,160032,160024,171174,1735131136,3204142126,4
160026,169096,9070,160030,160024,171174,820772864,3204069082,5
160026,168788,8762,160028,160024,171174,4201381888,3204056954,6
160024,169196,9172,160030,160024,171174,3287023616,3204364824,7
160026,168176,8150,160028,160024,171174,2372665344,3204073670,8
160026,169466,9440,160032,160024,171174,1458307072,3204068092,9
160026,168858,8832,160032,160024,171174,543948800,3204109862,10
160026,169418,9392,160028,160024,171174,3924557824,3204289508,11
160026,167776,7750,160032,160024,171174,3010199552,3204089538,12
160024,170538,10514,160032,160024,171174,2095841280,3204109170,13
160026,169320,9294,160034,160024,171174,1181483008,3204108772,14
160026,169976,9950,160034,160024,171174,267124736,3204259754,15
160026,166826,6800,160030,160024,171174,3647733760,3204058488,16
160026,168314,8288,160032,160024,171174,2733375488,3204110518,17
160026,170176,10150,160028,160024,171174,1819017216,3204283146,18
160024,168698,8674,160030,160024,171174,904658944,3204162904,19
160026,168234,8208,160034,160024,171174,4285267968,3204059562,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several speculative execution CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[238].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is Intel(R) Xeon(R) Gold 6252N CPU @ 2.30GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (Intel SSBD)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: YES (L1D flush feature bit)
  * Microarchitectural Data Sampling
    * VERW instruction is available: YES (MD_CLEAR feature bit)
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: YES
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: YES
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): YES
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): NO
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): NO
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): YES
    * TSX_CTRL MSR indicates TSX RTM is disabled: YES
    * TSX_CTRL MSR indicates TSX CPUID bit is cleared: YES
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x55␣
↪stepping 0x7 ucode 0x500002c cpuid 0x50657)
  * CPU microcode is the latest known available version: NO (latest version is 0x500320a␣
↪dated 2021/08/13 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
```

(continues on next page)

[238] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): YES
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): YES
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): YES
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Enhanced IBRS, IBPB:␣
↪conditional, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB+RSB filling, is needed to mitigate the␣
↪vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your kernel reported your CPU model as not affected)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Mitigation: TSX disabled)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (KVM: Mitigation: VMX disabled)
> STATUS: NOT VULNERABLE (KVM: Mitigation: VMX disabled)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

EPYC Zen2

Following sections include sample calibration data measured on server running in one of the AMD EPYC testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=cac1254f-9426-4ea6-a8db-2554f075db99␣
↪ro amd_iommu=on audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 hpet=disable iommu=pt isolcpus=1-15,17-31,33-47,49-63 nmi_watchdog=0␣
↪nohz_full=off nosoftlockup numa_balancing=disable processor.max_cstate=0 rcu_nocbs=1-15,
↪17-31,33-47,49-63 tsc=reliable console=ttyS0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux s60-t210-sut1 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64␣
↪x86_64 x86_64 GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 3 /home/testuser/pma_tools/jitter/jitter -i 30
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
116400,145848,29448,116400,116400,145848,2076377088,2375383296,1
116400,145848,29448,116400,116400,145848,388169728,2363555544,2
116400,145848,29448,116400,116400,145848,2994929664,2359881480,3
116400,145848,29448,116400,116400,145848,1306722304,2367487104,4
116400,145848,29448,116400,116400,145848,3913482240,2357721768,5
116400,145848,29448,116400,116400,145848,2225274880,2381723112,6
116400,145848,29448,116424,116400,145848,537067520,2373138432,7
116400,145848,29448,116424,116400,145848,3143827456,2372221464,8
116400,145848,29448,116400,116400,145848,1455620096,2365450272,9
116400,145848,29448,116400,116400,145848,4062380032,2364814440,10
116400,145848,29448,116400,116400,145848,2374172672,2375992608,11
116400,145848,29448,116400,116400,145848,685965312,2362608552,12
116400,145848,29448,116400,116400,145848,3292725248,2362597944,13
```

(continues on next page)

(continued from previous page)

```
116400,145848,29448,145512,116400,145848,1604517888,2370049344,14
116400,145848,29448,116400,116400,145848,4211277824,2366291784,15
116400,145848,29448,116400,116400,145848,2523070464,2349077352,16
116400,145848,29448,116400,116400,145848,834863104,2375406360,17
116400,145848,29448,116400,116400,145848,3441623040,2373272976,18
116400,145848,29448,116400,116400,145848,1753415680,2382267192,19
116400,145848,29448,116400,116400,145848,65208320,2359406040,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several speculative execution CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[239].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is AMD EPYC 7532 32-Core Processor

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (IBRS_SUPPORT feature bit)
    * CPU indicates preferring IBRS always-on: NO
    * CPU indicates preferring IBRS over retpoline: YES
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (IBPB_SUPPORT feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (AMD STIBP feature bit)
    * CPU indicates preferring STIBP always-on: NO
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (AMD SSBD in SPEC_CTRL)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: NO
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x17 model 0x31␣
↪stepping 0x0 ucode 0x8301038 cpuid 0x830f10)
  * CPU microcode is the latest known available version: NO (latest version is 0x8301052␣
↪dated 2021/11/11 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): NO
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
```

(continues on next page)

[239] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Retpolines, IBPB:␣
↪conditional, IBRS_FW, STIBP: always-on, RSB filling)
> STATUS: VULNERABLE (retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
```

(continues on next page)

(continued from previous page)

```
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Denverton

Following sections include sample calibration data measured on server running in one of the Intel Atom Denverton testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.0-65-generic root=UUID=26ca7b0f-904a-462d-a1c6-98c420c29515␣
↪ro audit=0 hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_pstate=disable␣
↪iommu=pt isolcpus=1-5 mce=off nmi_watchdog=0 nohz_full=1-5 nosoftlockup numa_
↪balancing=disable processor.max_cstate=1 rcu_nocbs=1-5 tsc=reliable console=tty0␣
↪console=ttyS0,115200n8
```

Linux uname

```
$ uname -a
Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 2 /home/testuser/pma_tools/jitter/jitter -c 2 -i 20
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number
Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
177008,217292,40284,177552,177008,217292,80543744,3555521762,1
167862,222370,54508,177552,167862,222370,191692800,3555482758,2
172576,251932,79356,177538,167862,251932,302841856,3556013278,3
177368,215300,37932,177552,167862,251932,413990912,3555428816,4
167914,215066,47152,177552,167862,251932,525139968,3555415700,5
177494,241748,64254,177552,167862,251932,636289024,3555835494,6
177038,210186,33148,177552,167862,251932,747438080,3555398164,7
170956,211022,40066,177552,167862,251932,858587136,3555435464,8
174130,237428,63298,177552,167862,251932,969736192,3555771752,9
174726,205252,30526,177552,167862,251932,1080885248,3555426516,10
177104,234502,57398,177554,167862,251932,1192034304,3555785760,11
175304,240416,65112,177550,167862,251932,1303183360,3555908234,12
166674,216176,49502,177552,166674,251932,1414332416,3555468016,13
177532,205792,28260,177552,166674,251932,1525481472,3555440968,14
177516,235032,57516,177550,166674,251932,1636630528,3555832414,15
177522,207292,29770,177552,166674,251932,1747779584,3555495058,16
177532,205174,27642,177552,166674,251932,1858928640,3555458754,17
177528,234230,56702,177552,166674,251932,1970077696,3555837046,18
177530,209364,31834,177552,166674,251932,2081226752,3555469590,19
177530,205002,27472,177552,166674,251932,2192375808,3555397840,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[240].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
```

(continues on next page)

[240] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
Kernel is Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64
CPU is Intel(R) Atom(TM) CPU C3858 @ 2.00GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: NO
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: NO
  * Microarchitectural Data Sampling
    * VERW instruction is available: NO
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: NO
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: NO
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): NO
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): NO
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): NO
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): NO
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x5f␣
↪stepping 0x1 ucode 0x20 cpuid 0x506f1)
  * CPU microcode is the latest known available version: NO (latest version is 0x36 dated␣
↪2021/05/10 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
```

(continues on next page)

(continued from previous page)

```
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Full generic retpoline,␣
↪IBPB: conditional, IBRS_FW, STIBP: disabled, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: VULNERABLE (an up-to-date CPU microcode is needed to mitigate this␣
↪vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: NO (Vulnerable)
> STATUS: VULNERABLE (Neither your CPU nor your kernel support SSBD)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)
```

(continues on next page)

(continued from previous page)

```
CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:KO CVE-2018-
↪3639:KO CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Snowridge

Following sections include sample calibration data measured on server running in one of the Intel Atom Snowridge testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-5.15.0-46-generic root=/dev/mapper/ubuntu--vg-ubuntu--lv ro audit=0␣
↪default_hugepagesz=2M hugepagesz=1G hugepages=2 hugepagesz=2M hugepages=4096␣
↪hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_pstate=disable iommu=pt␣
↪isolcpus=1-23 mce=off nmi_watchdog=0 nohz_full=1-23 nosoftlockup numa_balancing=disable␣
↪processor.max_cstate=1 rcu_nocbs=1-23 tsc=reliable console=ttyS0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 2 /home/testuser/pma_tools/jitter/jitter -c 2 -i 20
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
160370,165364,4994,160380,160370,165364,1042874368,3211228620,1
160370,165308,4938,160430,160370,165364,1279852544,3211283594,2
160370,169968,9598,160394,160370,169968,1516830720,3211446352,3
160370,166026,5656,160430,160370,169968,1753808896,3211263720,4
160370,165516,5146,160414,160370,169968,1990787072,3211249674,5
160370,165594,5224,160448,160370,169968,2227765248,3211267504,6
160370,169988,9618,160374,160370,169988,2464743424,3211426160,7
160370,165384,5014,160382,160370,169988,2701721600,3211243706,8
160370,165514,5144,160444,160370,169988,2938699776,3211233152,9
160370,168954,8584,160392,160370,169988,3175677952,3211338334,10
160370,167270,6900,160374,160370,169988,3412656128,3211329846,11
160370,165430,5060,160408,160370,169988,3649634304,3211240244,12
160370,166196,5826,160398,160370,169988,3886612480,3211256920,13
160370,169678,9308,160398,160370,169988,4123590656,3211415892,14
160370,165718,5348,160418,160370,169988,65601536,3211259448,15
160370,165256,4886,160372,160370,169988,302579712,3211236834,16
160370,167840,7470,160382,160370,169988,539557888,3211260000,17
160370,169332,8962,160400,160370,169988,776536064,3211432972,18
160370,165272,4902,160428,160370,169988,1013514240,3211246698,19
160370,165906,5536,160398,160370,169988,1250492416,3211262146,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[241].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is Intel Atom(R) P5362 processor

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (Intel SSBD)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: YES (L1D flush feature bit)
  * Microarchitectural Data Sampling
    * VERW instruction is available: YES (MD_CLEAR feature bit)
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: YES
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: YES
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): YES
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): NO
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): YES
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): NO
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x86␣
↪stepping 0x7 ucode 0x4c000019 cpuid 0x80667)
  * CPU microcode is the latest known available version: UNKNOWN (latest microcode␣
↪version for your CPU model is unknown)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
```

(continues on next page)

[241] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): YES
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Enhanced IBRS, IBPB:␣
↪conditional, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
```

(continues on next page)

(continued from previous page)

```
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Altra

Following sections include sample calibration data measured on server running in one of the Altra testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=7d1d0e77-4df0-43df-9619-a99db29ffb83␣
↪ro audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 iommu.passthrough=1 isolcpus=1-10,29-38 nmi_watchdog=0 nohz_full=1-10,
↪29-38 nosoftlockup processor.max_cstate=1 rcu_nocbs=1-10,29-38 console=ttyAMA0,115200n8␣
↪quiet
```

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:08:11 UTC 2022 aarch64 aarch64␣
↪aarch64 GNU/Linux
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[242].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:08:11 UTC 2022 aarch64
CPU is ARM v8 model 0xd0c

Hardware check
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): NO
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): NO
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: CSV2, BHB)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)
```

(continues on next page)

[242] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:OK CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

TaiShan

Following sections include sample calibration data measured on s17-t33-sut1 server running in one of the Cortex-A72 testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.0-65-generic root=UUID=7d1d0e77-4df0-43df-9619-a99db29ffb83␣
↪ro audit=0 intel_iommu=on isolcpus=1-27,29-55 nmi_watchdog=0 nohz_full=1-27,29-55␣
↪nosoftlockup processor.max_cstate=1 rcu_nocbs=1-27,29-55 console=ttyAMA0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[243].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:08:11 UTC 2022 aarch64
CPU is ARM v8 model 0xd08

Hardware check
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
```

(continues on next page)

[243] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: NO (Vulnerable)
> STATUS: VULNERABLE (Branch predictor hardening is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: VULNERABLE (an up-to-date CPU microcode is needed to mitigate this␣
↪vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: NO (Vulnerable)
> STATUS: VULNERABLE (Neither your CPU nor your kernel support SSBD)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:KO CVE-2018-
↪3639:KO CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

ThunderX2

Following sections include sample calibration data measured on s27-t211-sut1 server running in one of the ThunderX2 testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.0-65-generic root=UUID=7d1d0e77-4df0-43df-9619-a99db29ffb83␣
↪ro audit=0 intel_iommu=on isolcpus=1-27,29-55 nmi_watchdog=0 nohz_full=1-27,29-55␣
↪nosoftlockup processor.max_cstate=1 rcu_nocbs=1-27,29-55 console=ttyAMA0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[244].

```
Spectre and Meltdown mitigation detection tool v0.44+

Checking for vulnerabilities on current system
Kernel is Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:27:25 UTC 2021 aarch64
CPU is

Hardware check
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
```

(continues on next page)

[244] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): NO
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: __user pointer sanitization)
* Kernel has array_index_mask_nospec: NO
* Kernel has the Red Hat/Ubuntu patch: NO
* Kernel has mask_nospec64 (arm64): NO
* Kernel has array_index_nospec (arm64): NO
* Checking count of LFENCE instructions following a jump in kernel... NO (only 0 jump-
↪then-lfence instructions found, should be >= 30 (heuristic))
> STATUS: NOT VULNERABLE (Mitigation: __user pointer sanitization)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: NO (Vulnerable)
* Mitigation 1
  * Kernel is compiled with IBRS support: YES
    * IBRS enabled and active: NO
  * Kernel is compiled with IBPB support: NO
    * IBPB enabled and active: NO
* Mitigation 2
  * Kernel has branch predictor hardening (arm): YES
  * Kernel compiled with retpoline option: NO
> STATUS: NOT VULNERABLE (Branch predictor hardening mitigates the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports Page Table Isolation (PTI): YES
  * PTI enabled and active: UNKNOWN (dmesg truncated, please reboot and relaunch this␣
↪script)
  * Reduced performance impact of PTI: NO (PCID/INVPCID not supported, performance impact␣
↪of PTI will be significant)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: NO (Vulnerable)
```

(continues on next page)

(continued from previous page)

```
* Kernel supports disabling speculative store bypass (SSB): YES (found in /proc/self/
↪status)
* SSB mitigation is enabled and active: NO
> STATUS: VULNERABLE (Your CPU doesnt support SSBD)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports PTE inversion: NO
* PTE inversion enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
* This system is a host running a hypervisor: NO
* Mitigation 1 (KVM)
  * EPT is disabled: N/A (the kvm_intel module is not loaded)
* Mitigation 2
  * L1D flush is supported by kernel: NO
  * L1D flush enabled: NO
  * Hardware-backed L1D flush supported: NO (flush will be done in software, this is␣
↪slower)
  * Hyper-Threading (SMT) is enabled: UNKNOWN
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)
```

(continues on next page)

(continued from previous page)

```
CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
* TAA mitigation is supported by kernel: YES (found tsx_async_abort in kernel image)
* TAA mitigation enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
* This system is a host running a hypervisor: NO
* iTLB Multihit mitigation is supported by kernel: YES (found itlb_multihit in kernel␣
↪image)
* iTLB Multihit mitigation enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
* SRBDS mitigation control is supported by the kernel: NO
* SRBDS mitigation control is enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

> SUMMARY: CVE-2017-5753:OK CVE-2017-5715:OK CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:KO CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

CHAPTER

THREE

DPDK PERFORMANCE

3.1 Overview

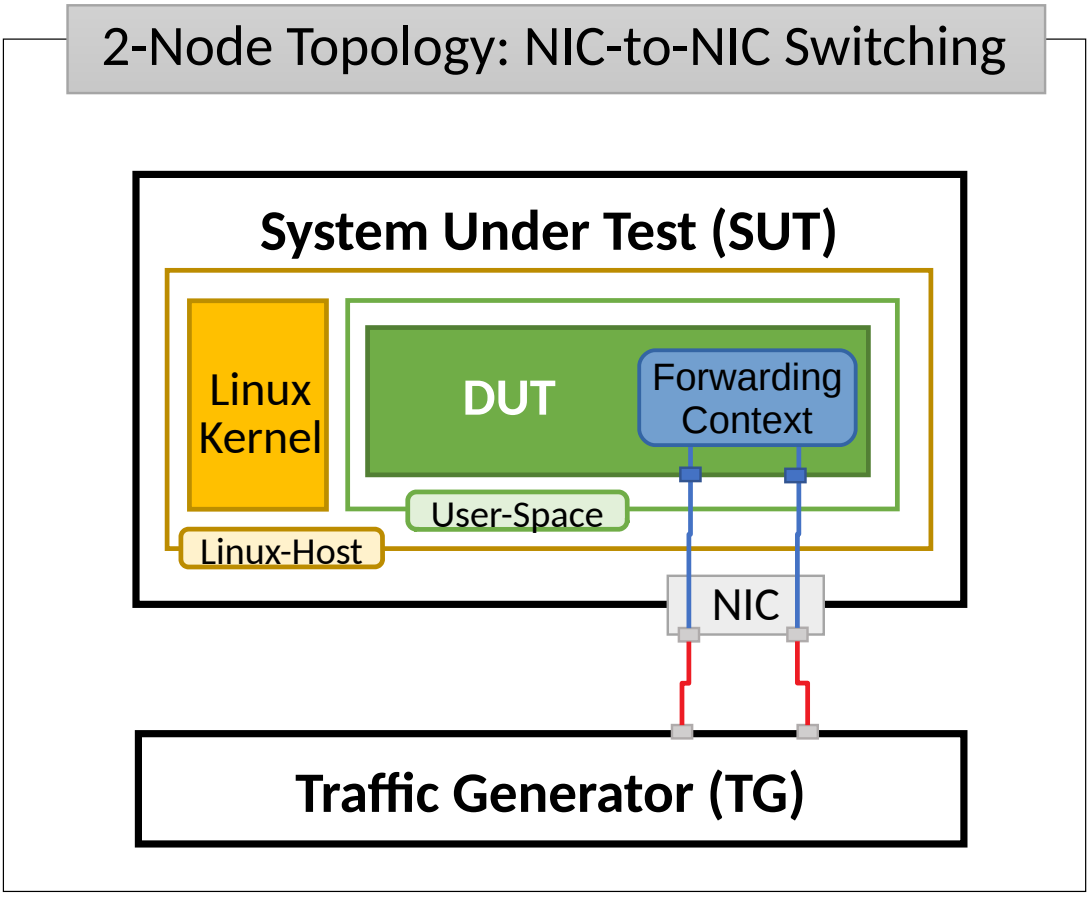
DPDK performance test results are reported for all physical testbed types present in FD.io labs and installed NIC models. For description of physical testbeds used for DPDK performance tests please refer to *Performance Physical Testbeds* (page 4).

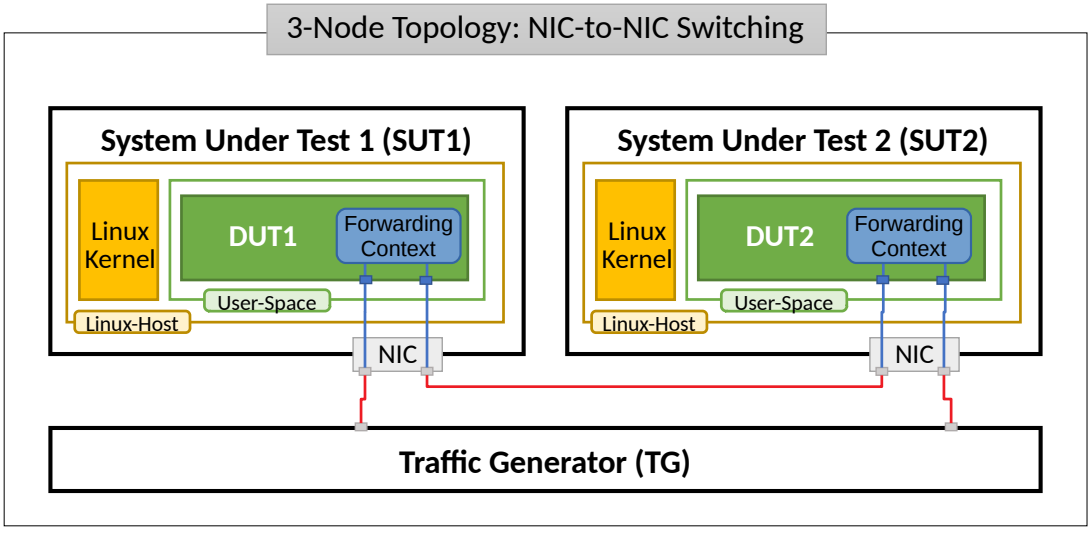
3.1.1 Logical Topologies

CSIT DPDK performance tests are executed on physical testbeds described in *Performance Physical Testbeds* (page 4). Based on the packet path through server SUTs, one distinct logical topology type is used for DPDK DUT data plane testing: NIC-to-NIC switching topology.

NIC-to-NIC Switching

The simplest logical topology for software data plane application like DPDK is NIC-to-NIC switching. Tested topologies for 2-Node and 3-Node testbeds are shown in figures below.





Server Systems Under Test (SUT) run DPDK Testpmd or L3fwd application in Linux user-mode as a Device Under Test (DUT). Server Traffic Generator (TG) runs T-Rex application. Physical connectivity between SUTs and TG is provided using different drivers and NIC models that need to be tested for performance (packet/bandwidth throughput and latency).

From SUT and DUT perspectives, all performance tests involve forwarding packets between two physical Ethernet ports (10GE, 25GE, 40GE, 100GE). In most cases both physical ports on SUT are located on the same NIC. The only exceptions are link bonding and 100GE tests. In the latter case only one port per NIC can be driven at linerate due to PCIe Gen3 x16 slot bandwidth limiations. 100GE NICs are not supported in PCIe Gen3 x8 slots.

Note that reported DPDK DUT performance results are specific to the SUTs tested. SUTs with other processors than the ones used in FD.io lab are likely to yield different results. A good rule of thumb, that can be applied to estimate DPDK packet thoughput for NIC-to-NIC switching topology, is to expect the forwarding performance to be proportional to processor core frequency for the same processor architecture, assuming processor is the only limiting factor and all other SUT parameters are equivalent to FD.io CSIT environment.

3.1.2 Performance Tests Coverage

Performance tests measure following metrics for tested DPDK DUT topologies and configurations:

- Packet Throughput: measured in accordance with **RFC 2544**[245], using FD.io CSIT Multiple Loss Ratio search (MLRsearch), an optimized binary search algorithm, producing throughput at different Packet Loss Ratio (PLR) values:
  - Non Drop Rate (NDR): packet throughput at PLR=0%.
  - Partial Drop Rate (PDR): packet throughput at PLR=0.5%.
- One-Way Packet Latency: measured at different offered packet loads:
  - 100% of discovered NDR throughput.
  - 100% of discovered PDR throughput.
- Maximum Receive Rate (MRR): measured packet forwarding rate under the maximum load offered by traffic generator over a set trial duration, regardless of packet loss. Maximum load for specified Ethernet frame size is set to the bi-directional link rate.

CSIT-2210 includes following DPDK Testpmd and L3fwd data plane functionality performance tested across a range of NIC drivers and NIC models:

Functionality	Description
L2IntLoop	L2 Interface Loop forwarding all Ethernet frames between two Interfaces.
IPv4 Routed Forwarding	Longest Prefix Match (LPM) L3 IPv4 forwarding of Ethernet frames between two Interfaces, with two /8 prefixes in lookup table.

3.2 Release Notes

3.2.1 Changes in CSIT-2210

1. TEST FRAMEWORK
   - **CSIT test environment** version has been updated to ver. 11, see *Environment Versioning* (page 1206).
2. DPDK PERFORMANCE TESTS
   - No updates
3. DPDK RELEASE VERSION CHANGE
   - CSIT-2210 tested DPDK-22.07, as used by VPP-22.10 release.

[245] https://datatracker.ietf.org/doc/html/rfc2544.html

3.2.2 Known Issues

List of known issues in CSIT-2210 for DPDK performance tests:

#	JiraID	Issue Description
1	CSIT-1762[246]	TRex reports link DOWN in case of dpdk testpmd tests on FD.io CSIT Denverton systems (2n-dnv and 3n-dnv).
2	CSIT-1848[247]	2n-clx, 3n-alt: sporadic testpmd/l3fwd tests fail with no or low traffic.

New

List of new issues in CSIT-2210 for DPDK performance tests:

#	JiraID	Issue Description
1	CSIT-1870[248]	2n-clx, 2n-icx, 2n-zn2: DPDK testpmd 9000b tests on xxv710 nic are failing with no traffic.

[246] https://jira.fd.io/browse/CSIT-1762
[247] https://jira.fd.io/browse/CSIT-1848
[248] https://jira.fd.io/browse/CSIT-1870

3.3 Packet Throughput

Throughput graphs are generated by multiple executions of the same performance tests across physical testbeds hosted LF FD.io labs: 2n-icx, 3n-icx, 2n-clx, 2n-zn2, 3n-alt, 3n-tsh, 2n-tx2. Box-and-Whisker plots are used to display variations in measured throughput values, without making any assumptions of the underlying statistical distribution.

For each test case, Box-and-Whisker plots show the quartiles (Min, 1st quartile / 25th percentile, 2nd quartile / 50th percentile / mean, 3rd quartile / 75th percentile, Max) across collected data set. Outliers are plotted as individual points.

Additional information about graph data:

1. **Graph Title**: describes tested packet path, testbed topology, processor model, NIC model, packet size, number of cores and threads used by data plane workers and indication of DPDK DUT configuration.
2. **X-axis Labels**: indices of individual test suites as listed in Graph Legend.
3. **Y-axis Labels**: measured Packets Per Second [pps] throughput values.
4. **Graph Legend**: lists X-axis indices with associated CSIT test suites executed to generate graphed test results.
5. **Hover Information**: lists minimum, first quartile, median, third quartile, and maximum. If either type of outlier is present the whisker on the appropriate side is taken to 1.5×IQR from the quartile (the "inner fence") rather than the max or min, and individual outlying data points are displayed as unfilled circles (for suspected outliers) or filled circles (for outliers). (The "outer fence" is 3×IQR from the quartile.)

Note: Test results are stored in build logs from FD.io dpdk performance job 2n-icx[249], build logs from FD.io dpdk performance job 3n-icx[250], build logs from FD.io dpdk performance job 2n-clx[251], build logs from FD.io dpdk performance job 2n-zn2[252], build logs from FD.io dpdk performance job 3n-alt[253], build logs from FD.io dpdk performance job 3n-tsh[254], **`build logs from FD.io dpdk performance job 3n-snr`_**, build logs from FD.io dpdk performance job 2n-tx2[255] with RF result files csit-dpdk-perf-2210-*.zip archived here. Required per test case data set size is **10** and for DPDK tests this is the actual size, as all scheduled test executions completed successfully.

---

[249] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-icx
[250] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-icx
[251] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-clx
[252] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-zn2
[253] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-alt
[254] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-tsh
[255] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-tx2

3.3.1 2n-icx-xxv710

Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[256].

[256] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

64b-4t2c-base

3.3.2 3n-icx-xxv710

Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[257].

[257] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

64b-4t2c-base

3.3.3 2n-clx-xxv710

Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[258].

[258] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

64b-4t2c-base

3.3.4 2n-clx-x710

Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[259].

[259] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

64b-4t2c-base

3.3.5 2n-zn2-xxv710

Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[260].

[260] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

64b-4t2c-base

3.3.6 2n-zn2-x710

Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[261].

[261] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

64b-4t2c-base

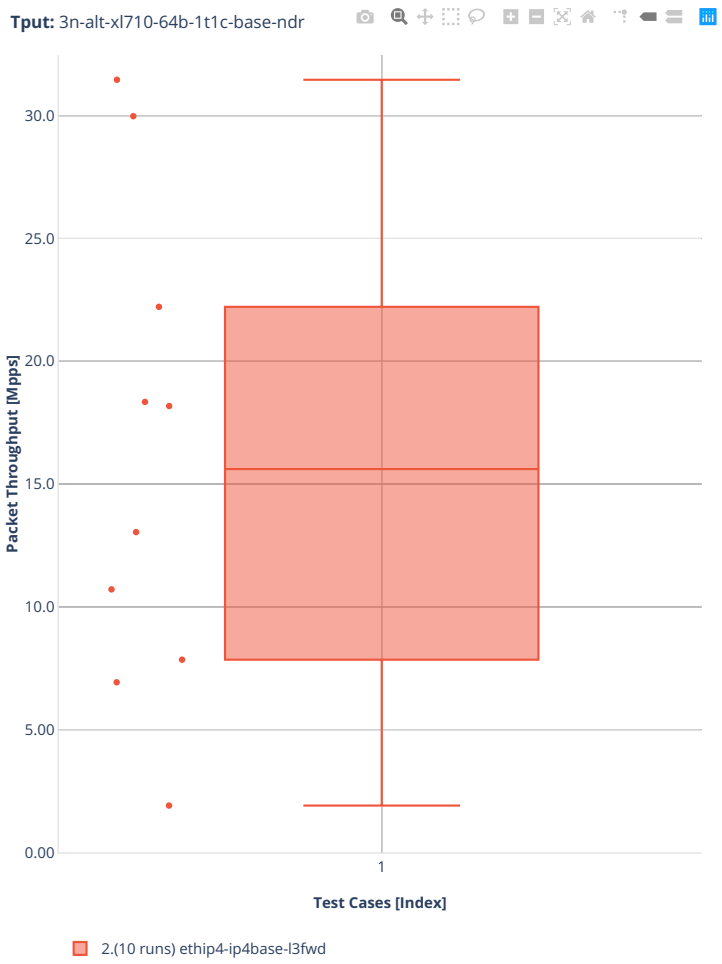
3.3.7 3n-alt-xl710

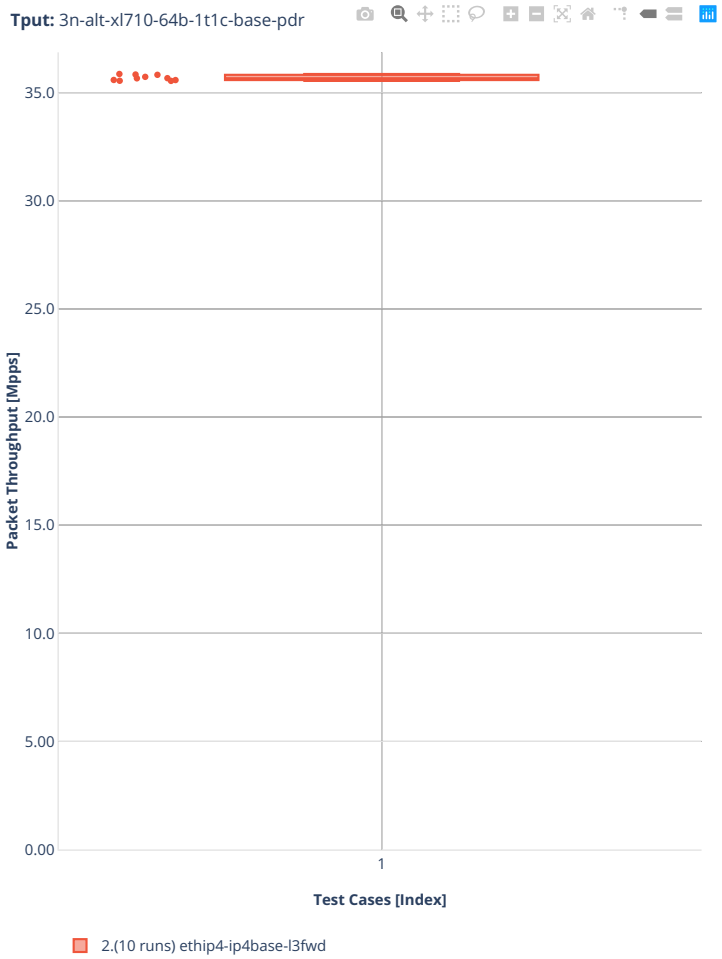
Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[262].

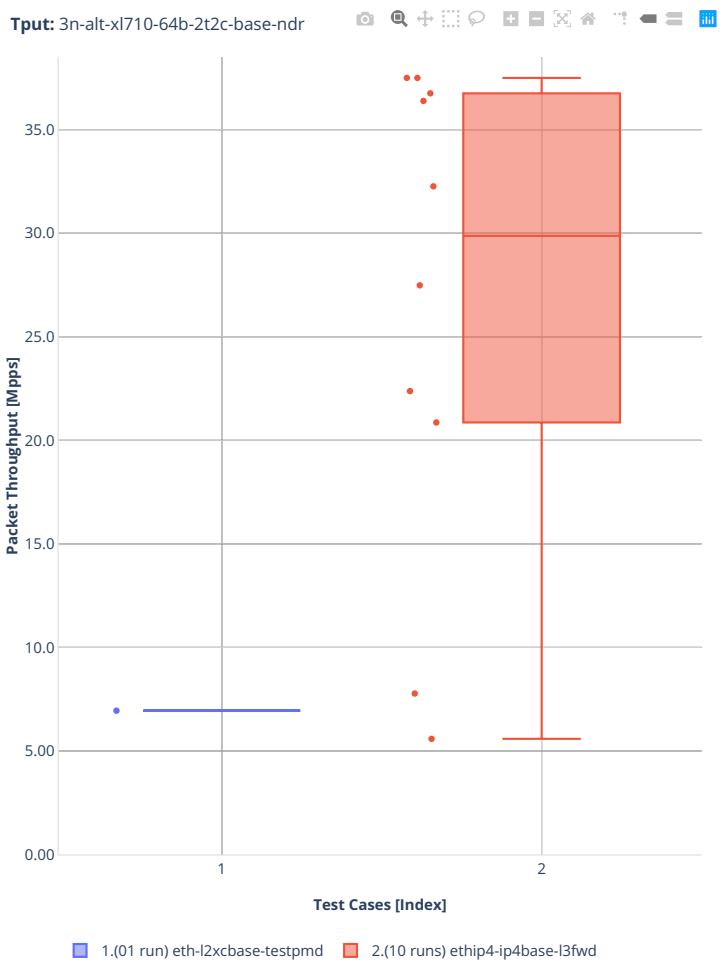
[262] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

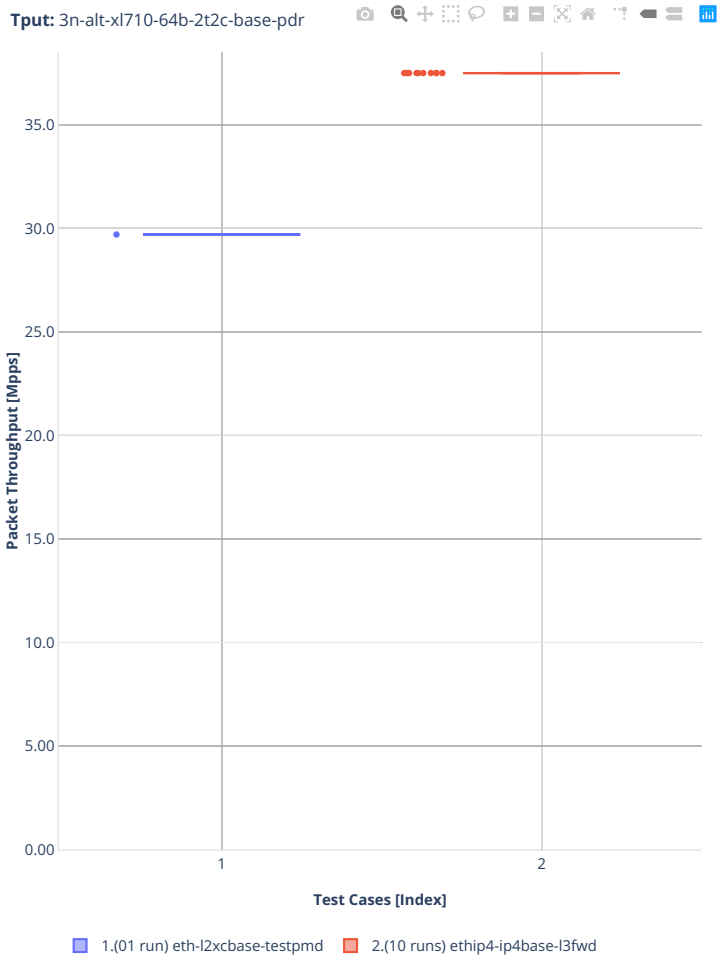
64b-1t1c-base




Tput: 3n-alt-xl710-64b-1t1c-base-pdr
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
2.(10 runs) ethip4-ip4base-l3fwd

64b-2t2c-base




Tput: 3n-alt-xl710-64b-2t2c-base-pdr
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(01 run) eth-l2xcbase-testpmd
2.(10 runs) ethip4-ip4base-l3fwd

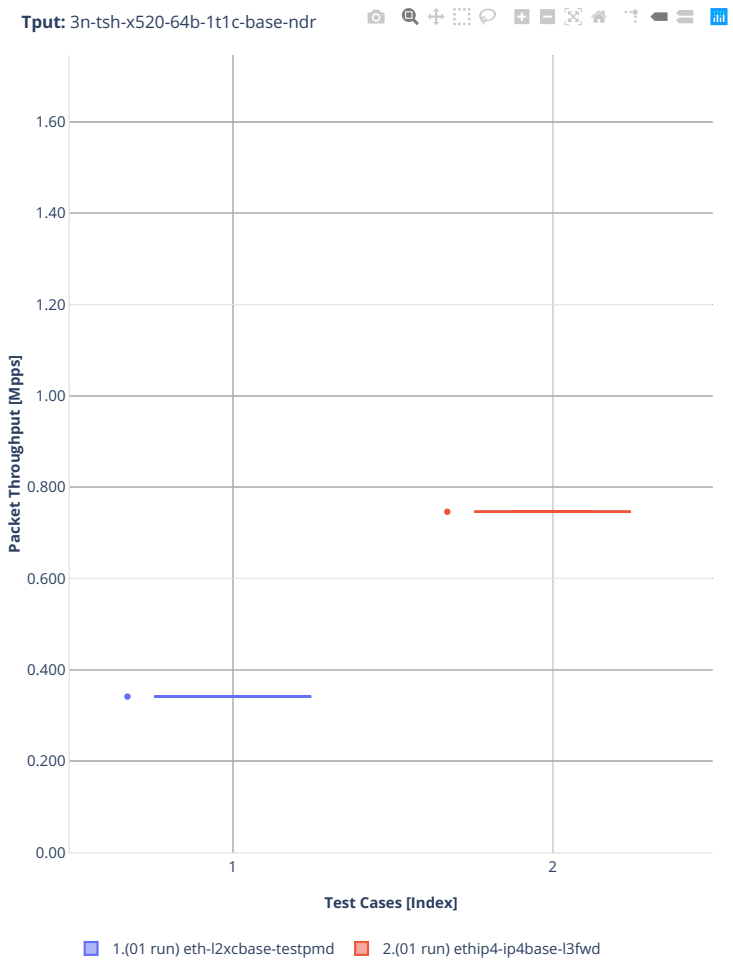
3.3.8 3n-tsh-x520

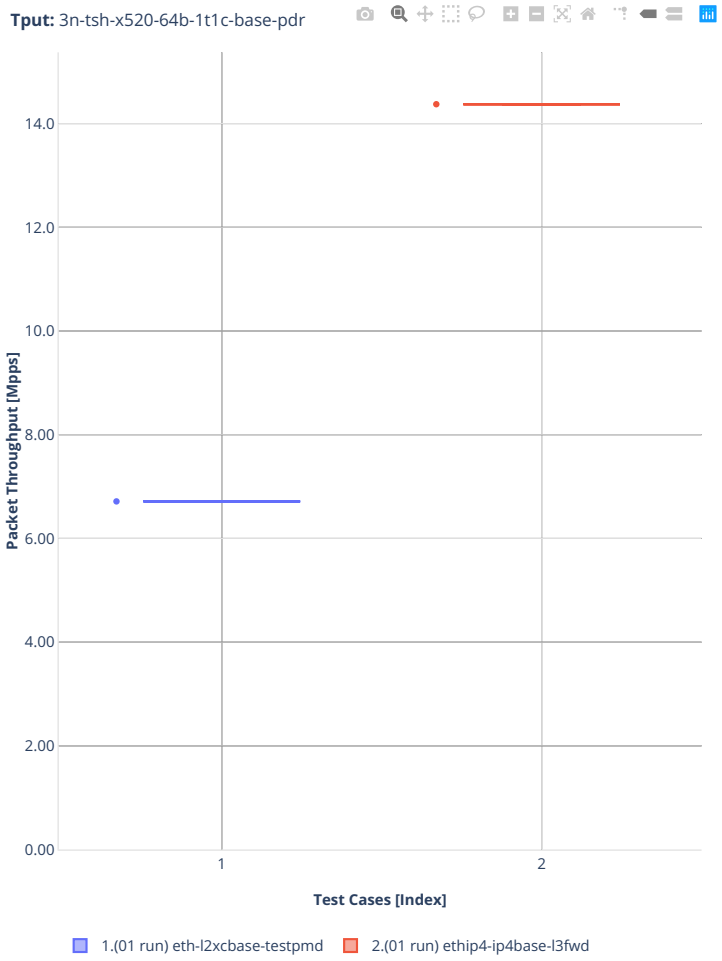
Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[263].

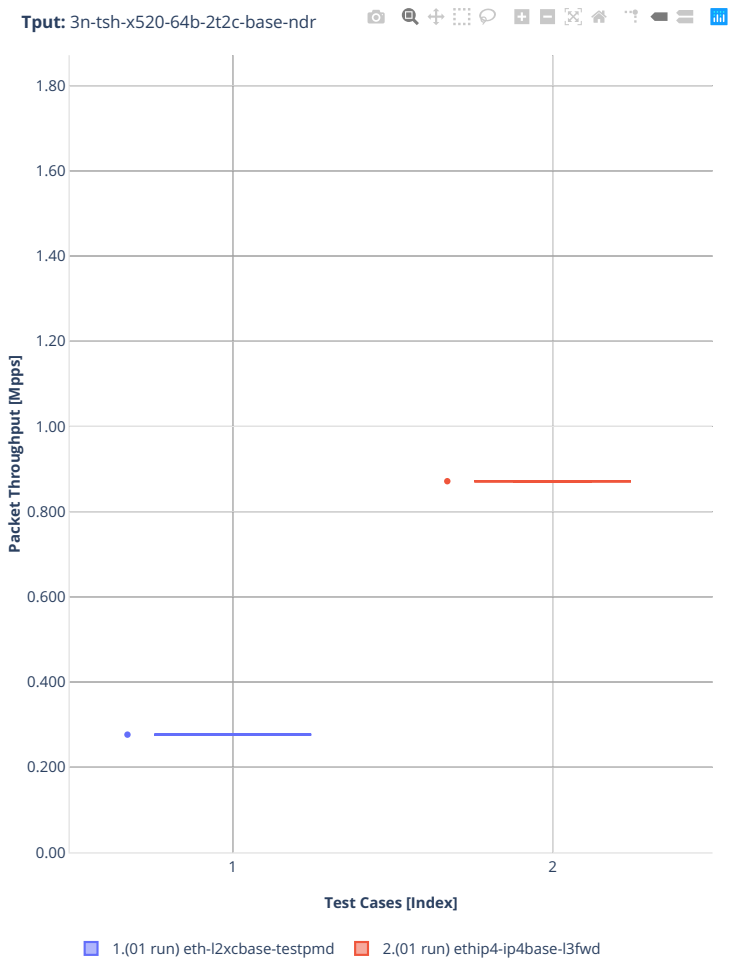
[263] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

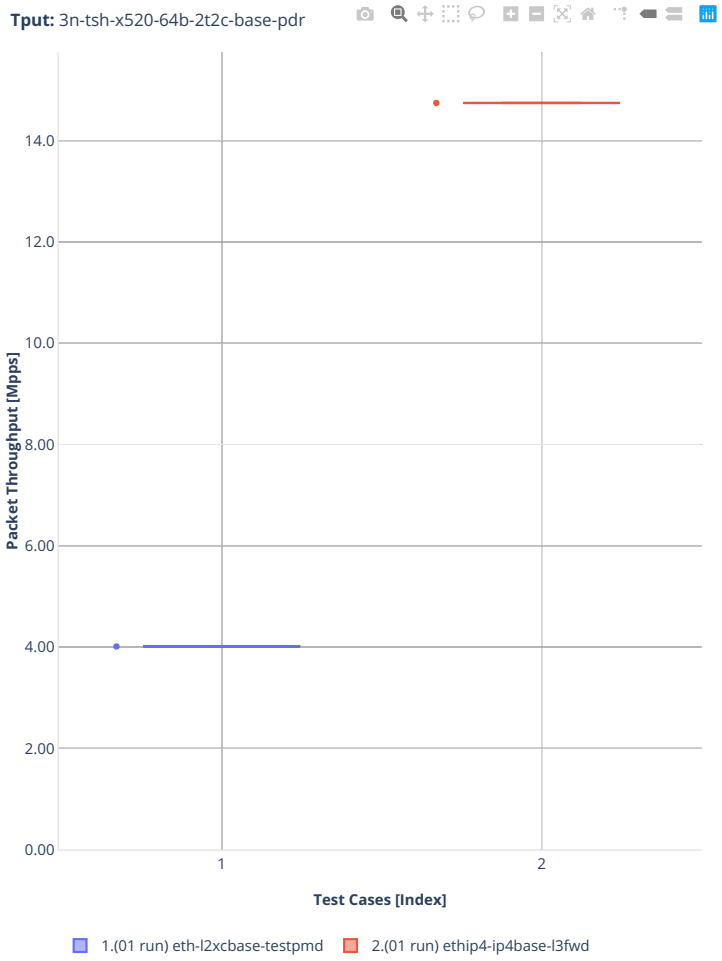
64b-1t1c-base




Tput: 3n-tsh-x520-64b-1t1c-base-pdr
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(01 run) eth-l2xcbase-testpmd
2.(01 run) ethip4-ip4base-l3fwd

64b-2t2c-base




Tput: 3n-tsh-x520-64b-2t2c-base-pdr
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(01 run) eth-l2xcbase-testpmd
2.(01 run) ethip4-ip4base-l3fwd

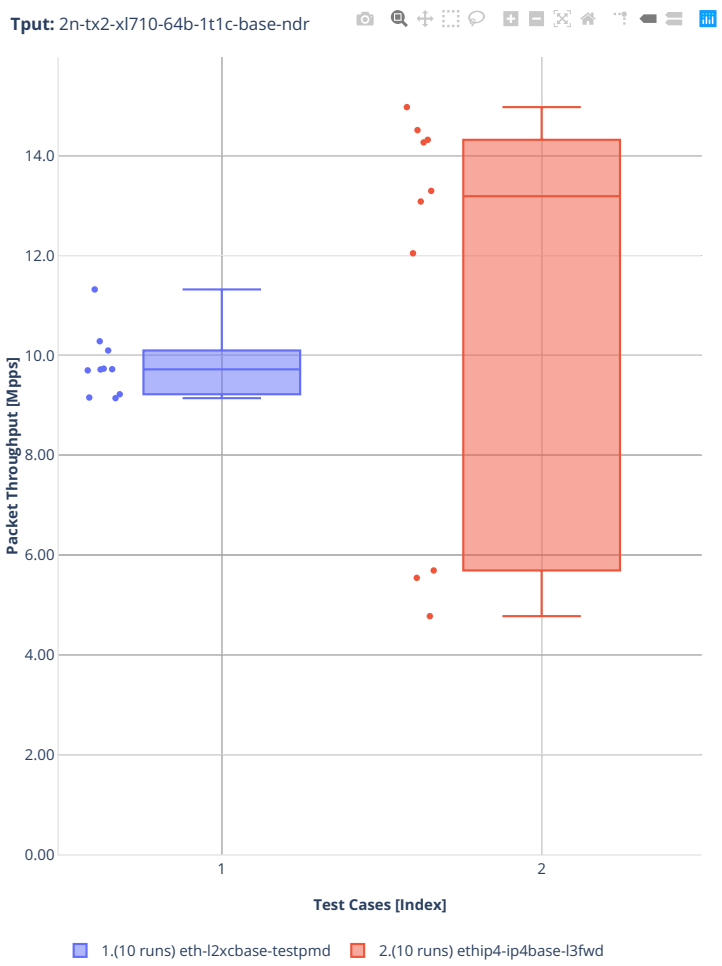
3.3.9 2n-tx2-xl710

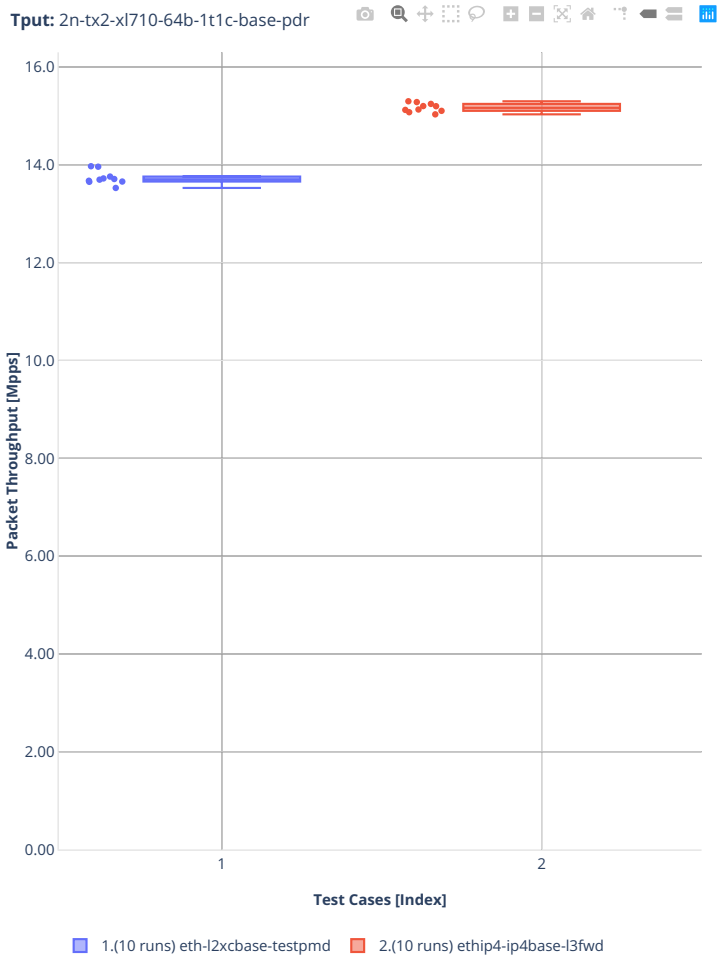
Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[264].

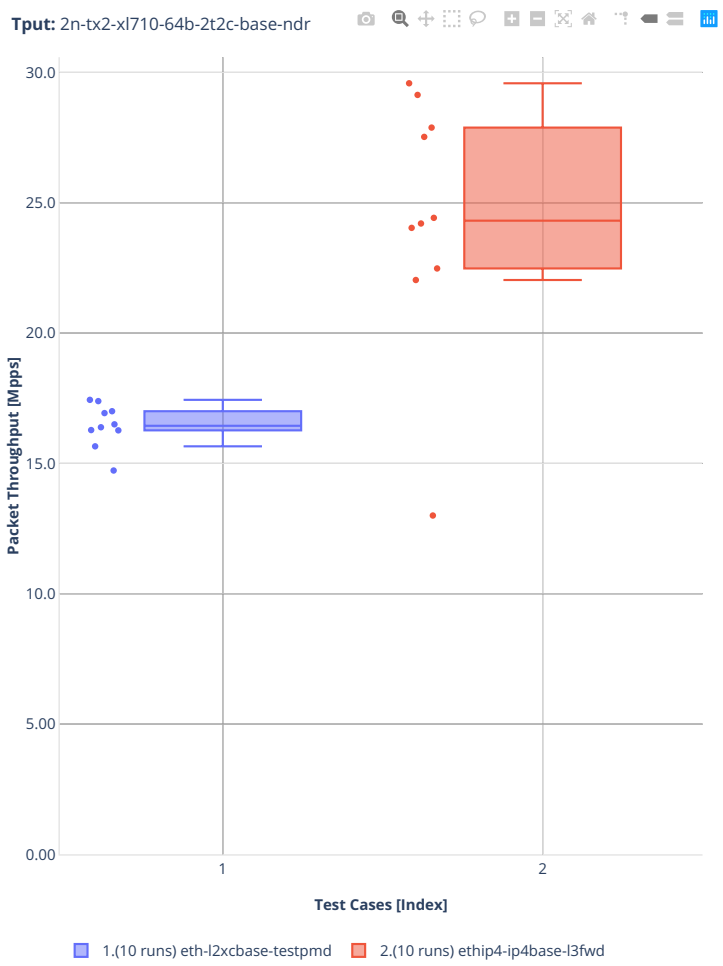
[264] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

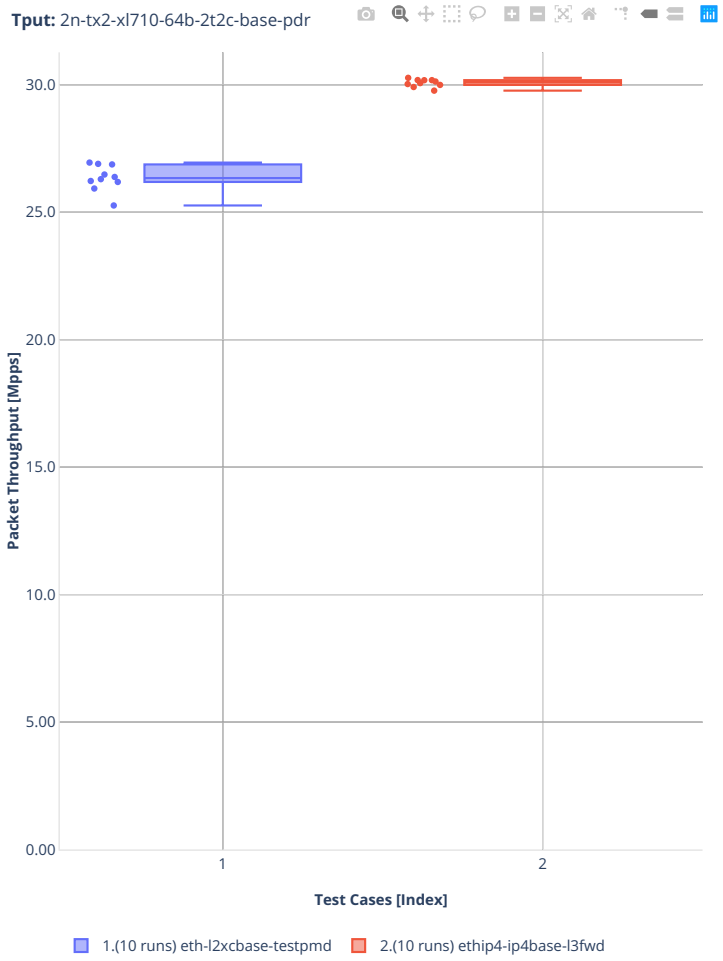
64b-1t1c-base




Tput: 2n-tx2-xl710-64b-1t1c-base-pdr
16.0
14.0
12.0
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) eth-l2xcbase-testpmd
2.(10 runs) ethip4-ip4base-l3fwd

64b-2t2c-base




Tput: 2n-tx2-xl710-64b-2t2c-base-pdr
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) eth-l2xcbase-testpmd
2.(10 runs) ethip4-ip4base-l3fwd

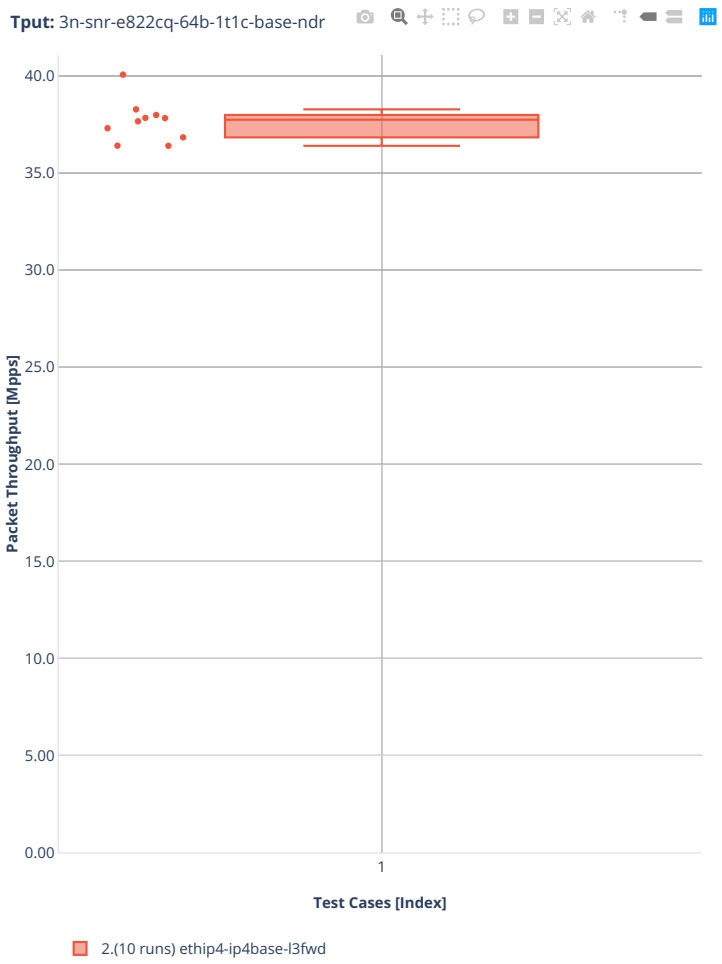
3.3.10 3n-snr-e822cq

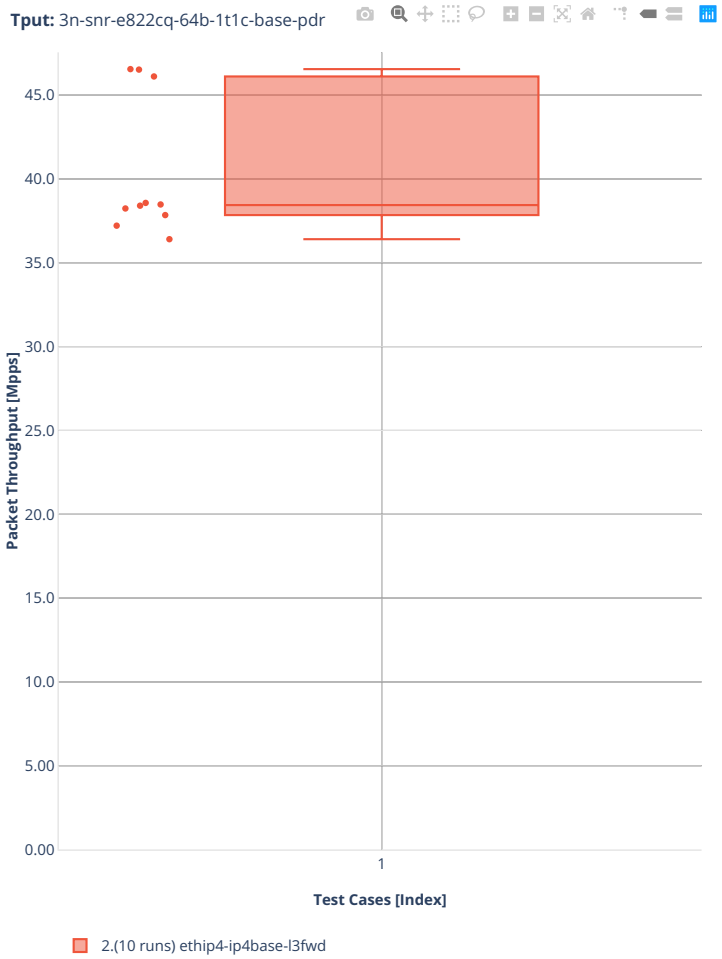
Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[265].

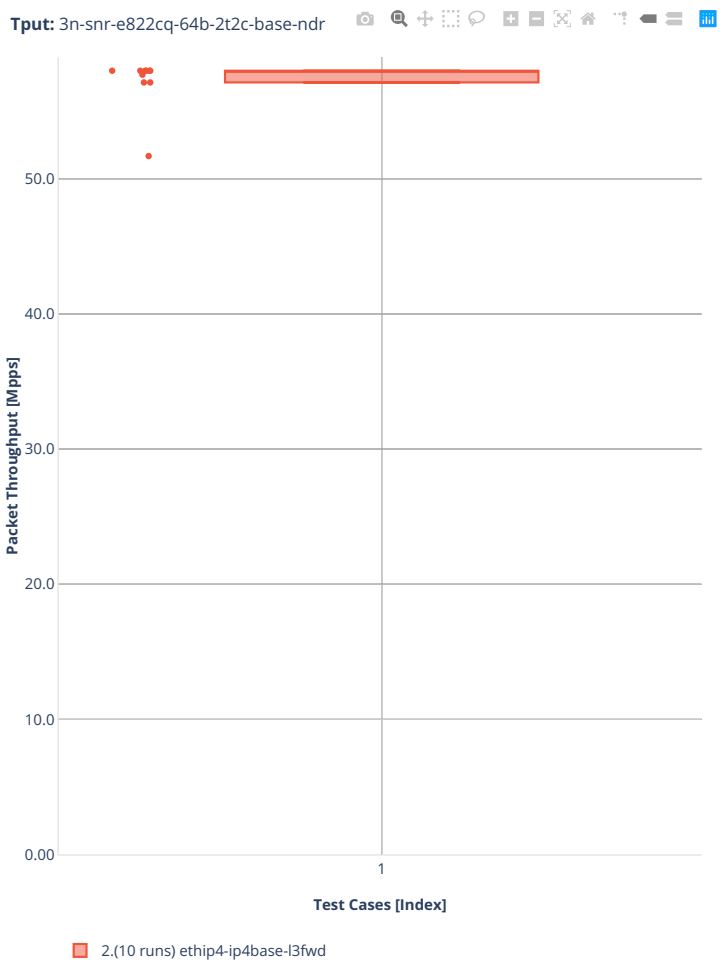
[265] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

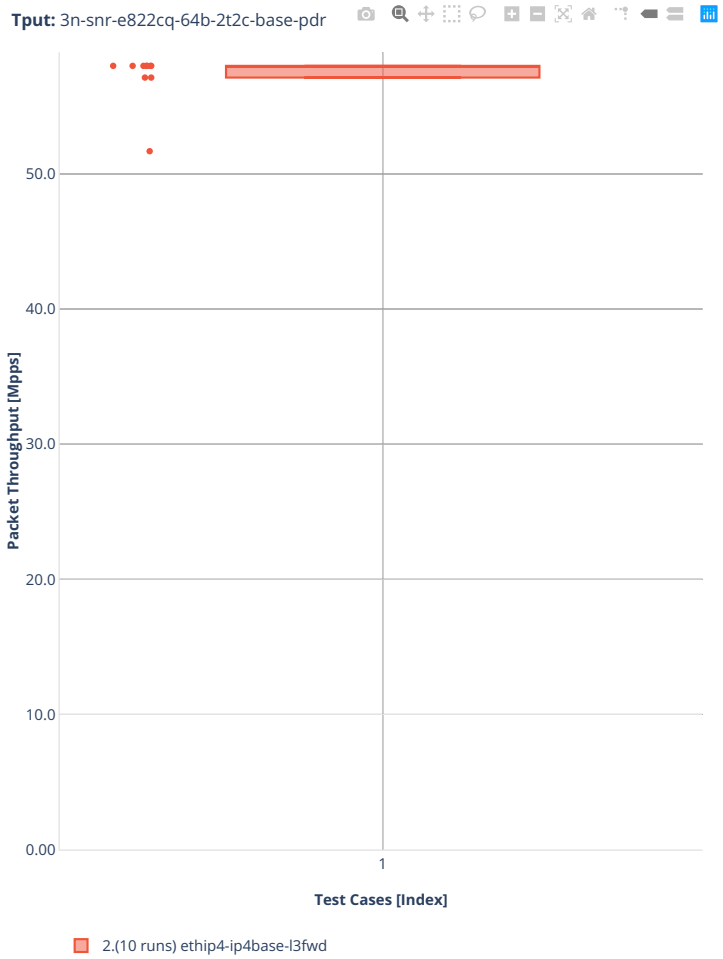
64b-1t1c-base




Tput: 3n-snr-e822cq-64b-1t1c-base-pdr
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
2.(10 runs) ethip4-ip4base-l3fwd

64b-2t2c-base




Tput: 3n-snr-e822cq-64b-2t2c-base-pdr
50.0
40.0
30.0
20.0
10.0
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
2.(10 runs) ethip4-ip4base-l3fwd

3.4 Speedup Multi-Core

Speedup Multi-Core throughput graphs are generated by multiple executions of the same performance tests across physical testbeds hosted LF FD.io labs: 2n-icx, 3n-icx, 2n-clx, 2n-zn2, 3n-alt, 3n-tsh, 2n-tx2. Grouped bars illustrate the 64B packet throughput speedup ratio for 2- and 4-core multi-threaded DPDK configurations relative to 1-core configurations.

Additional information about graph data:

1. **Graph Title**: describes tested packet path, testbed topology, processor model, NIC model, packet size used by data plane workers and indication of VPP DUT configuration.
2. **X-axis Labels**: number of cores.
3. **Y-axis Labels**: measured Packets Per Second [pps] throughput values.
4. **Graph Legend**: lists CSIT test suites executed to generate graphed test results.
5. **Hover Information**: lists number of runs executed, specific test substring, mean value of the measured packet throughput, calculated perfect throughput value, difference between measured and perfect values and relative speedup value.

Note: Test results are stored in build logs from FD.io dpdk performance job 2n-icx[266], build logs from FD.io dpdk performance job 3n-icx[267], build logs from FD.io dpdk performance job 2n-clx[268], build logs from FD.io dpdk performance job 2n-zn2[269], build logs from FD.io dpdk performance job 3n-alt[270], build logs from FD.io dpdk performance job 3n-tsh[271], **`build logs from FD.io dpdk performance job 3n-snr`_**, build logs from FD.io dpdk performance job 2n-tx2[272] with RF result files csit-vpp-perf-2210-*.zip archived here. Required per test case data set size is **10**, but for VPP tests the actual size varies per test case and is <=10.

---

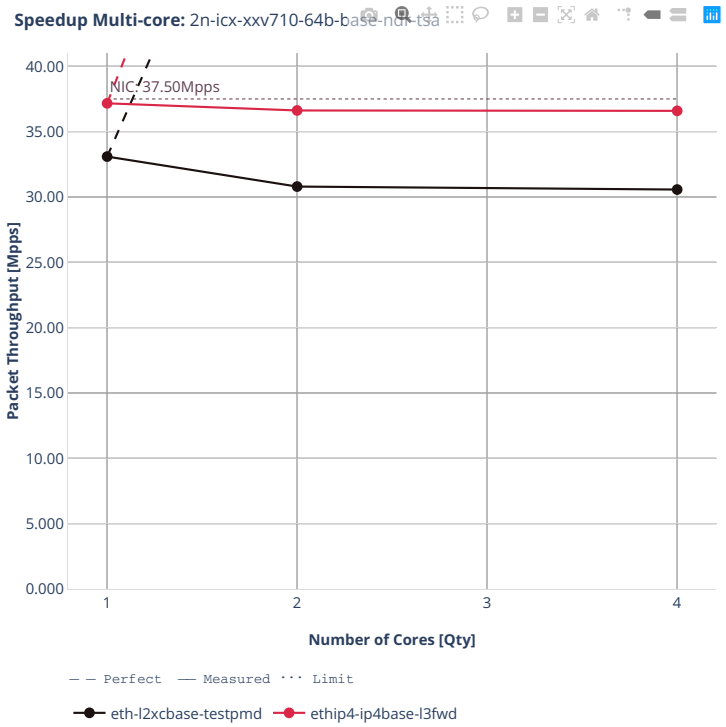
[266] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-icx
[267] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-icx
[268] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-clx
[269] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-zn2
[270] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-alt
[271] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-tsh
[272] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-tx2

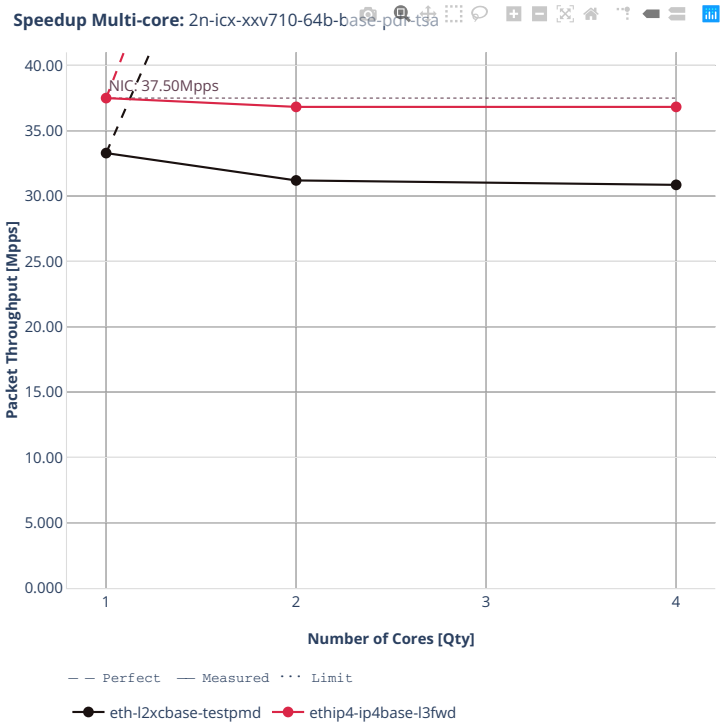
3.4.1 2n-icx-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[273].

[273] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 2n-icx-xxv710-64b-base-ndr-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

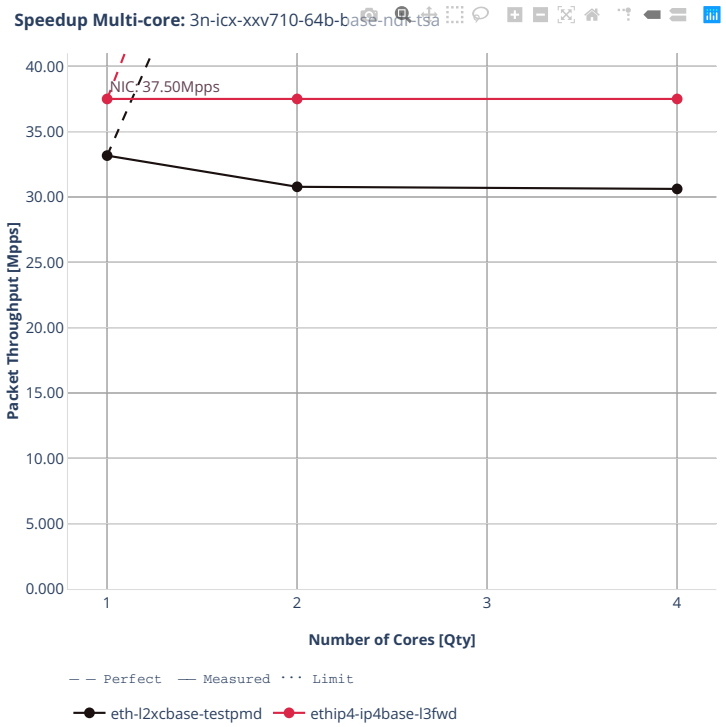

Speedup Multi-core: 2n-icx-xxv710-64b-base-pdr-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

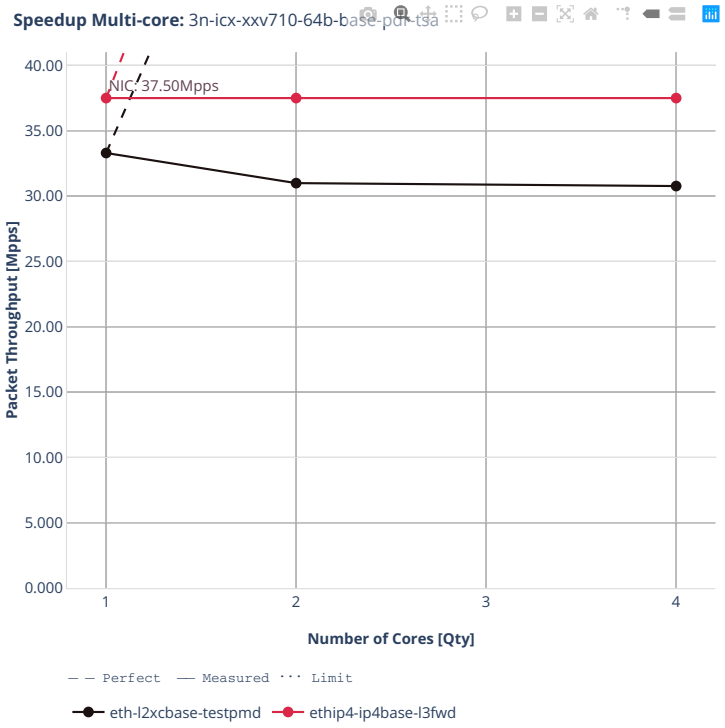
3.4.2 3n-icx-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[274].

[274] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 3n-icx-xxv710-64b-base-ndr-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

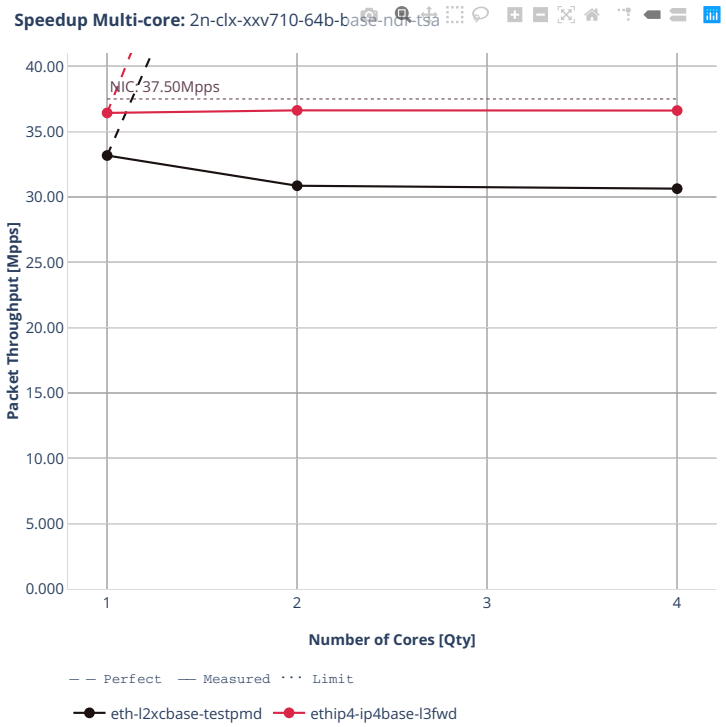

Speedup Multi-core: 3n-icx-xxv710-64b-base-pdr-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

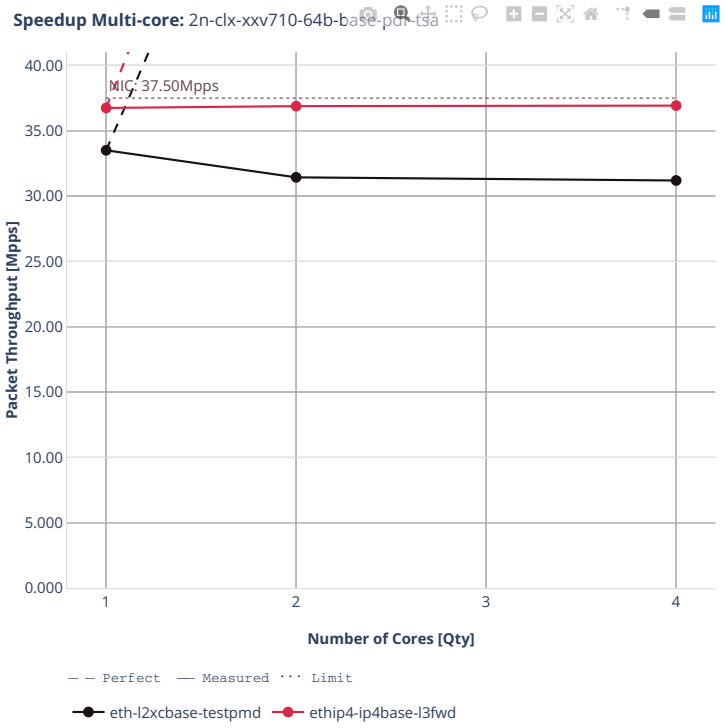
3.4.3 2n-clx-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[275].

[275] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 2n-clx-xxv710-64b-base-ndr-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

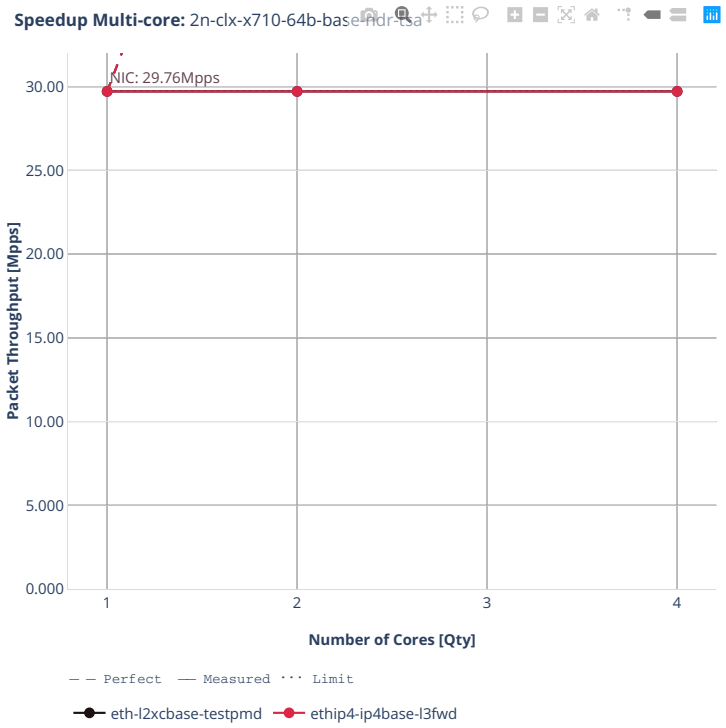

Speedup Multi-core: 2n-clx-xxv710-64b-base-pdr-tsa
MIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

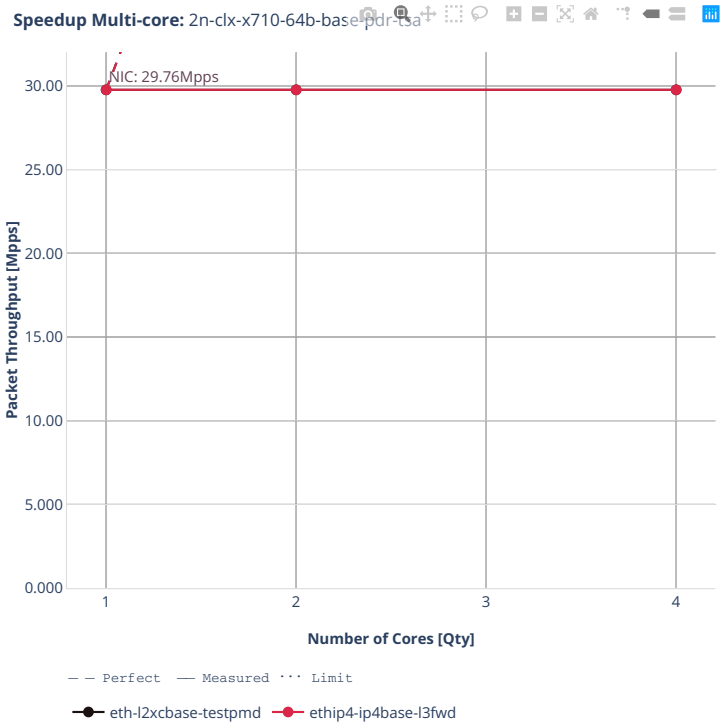
3.4.4 2n-clx-x710

CSIT source code for the test cases used for plots can be found in CSIT git repository[276].

[276] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 2n-clx-x710-64b-base-ndr-tsa
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

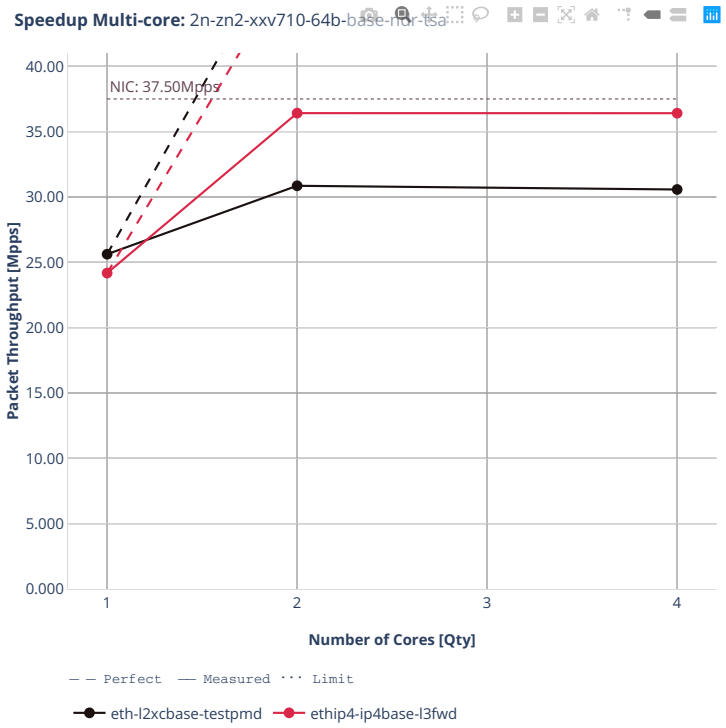

Speedup Multi-core: 2n-clx-x710-64b-base-pdr-tsa
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

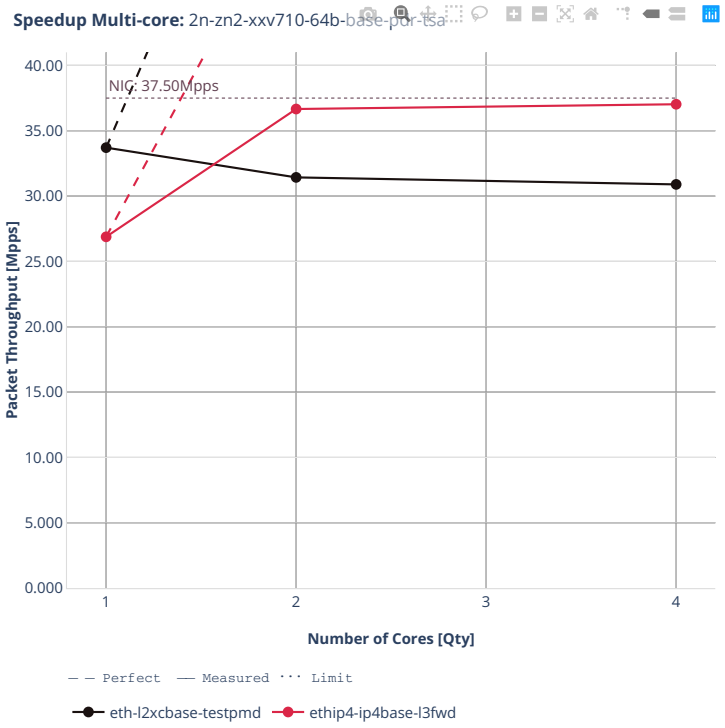
3.4.5 2n-zn2-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[277].

[277] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 2n-zn2-xxv710-64b-base-mrr-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

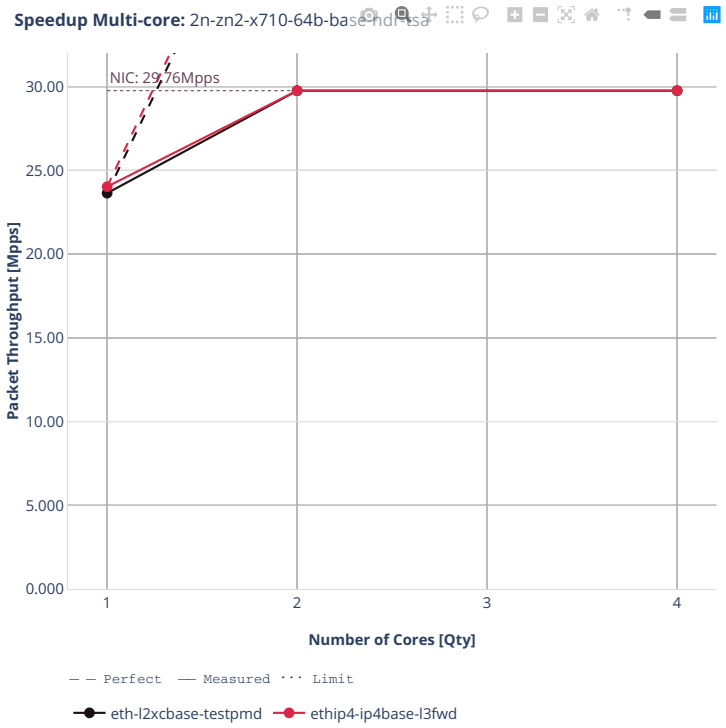

Speedup Multi-core: 2n-zn2-xxv710-64b-base-por-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

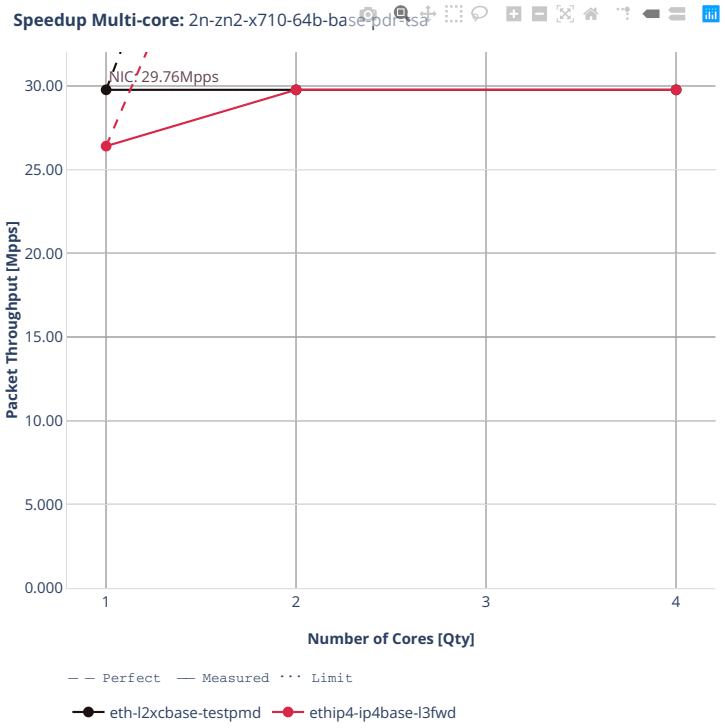
3.4.6 2n-zn2-x710

CSIT source code for the test cases used for plots can be found in CSIT git repository[278].

[278] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 2n-zn2-x710-64b-base-ndrpsa
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

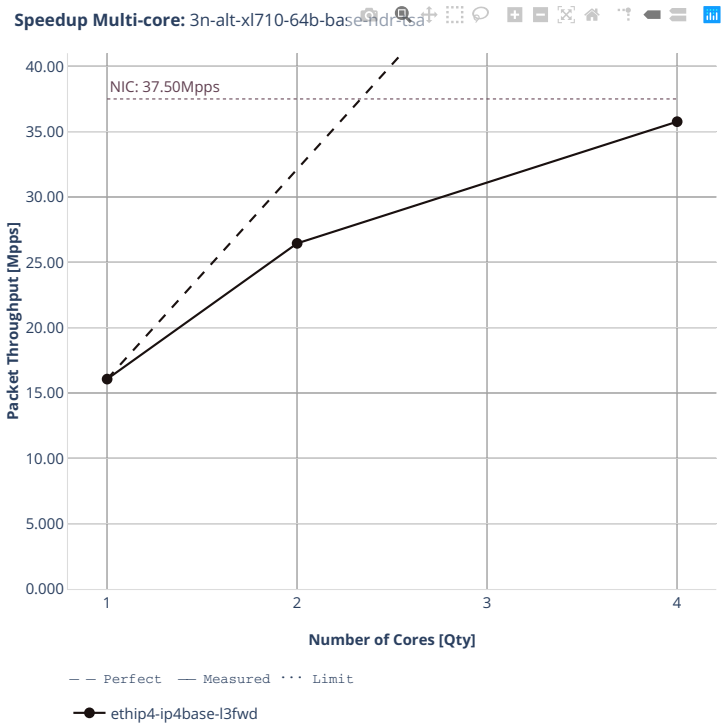

Speedup Multi-core: 2n-zn2-x710-64b-base-pdr-tsa
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

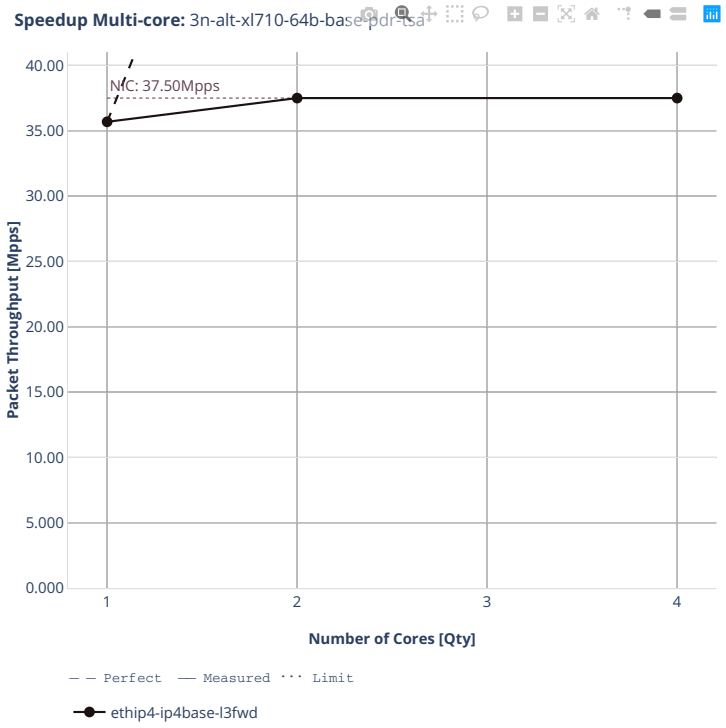
3.4.7 3n-alt-xl710

CSIT source code for the test cases used for plots can be found in CSIT git repository[279].

[279] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 3n-alt-xl710-64b-base-ndr-lsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base-l3fwd

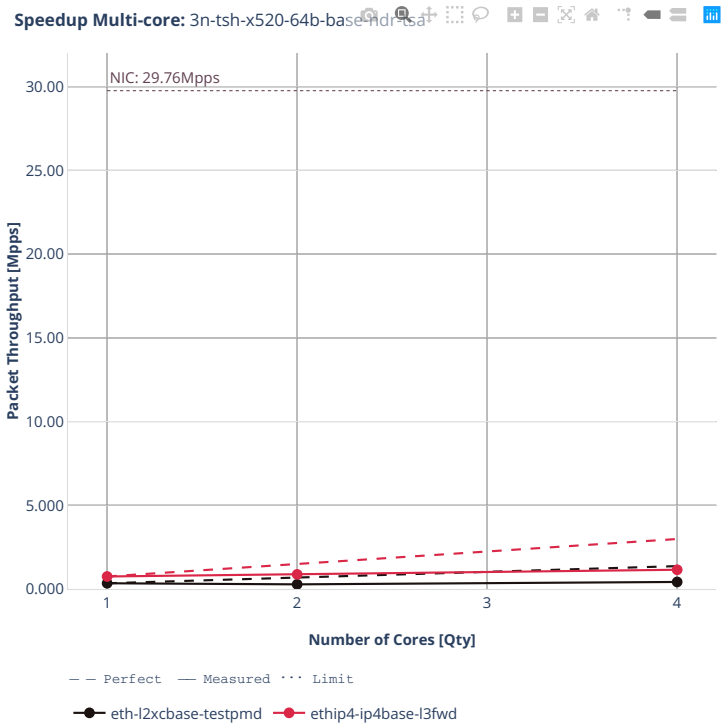

Speedup Multi-core: 3n-alt-xl710-64b-base-pdr-lsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base-l3fwd

3.4.8 3n-tsh-x520

CSIT source code for the test cases used for plots can be found in CSIT git repository[280].

[280] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 3n-tsh-x520-64b-base-ndr-tsa
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

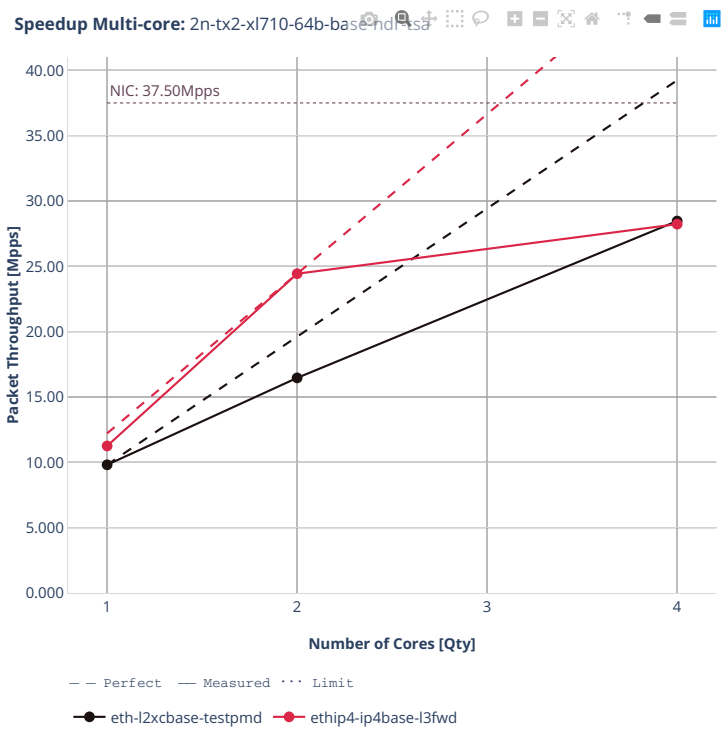

Speedup Multi-core: 3n-tsh-x520-64b-base-pdr-tsa
NIC: 29.76Mpps
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

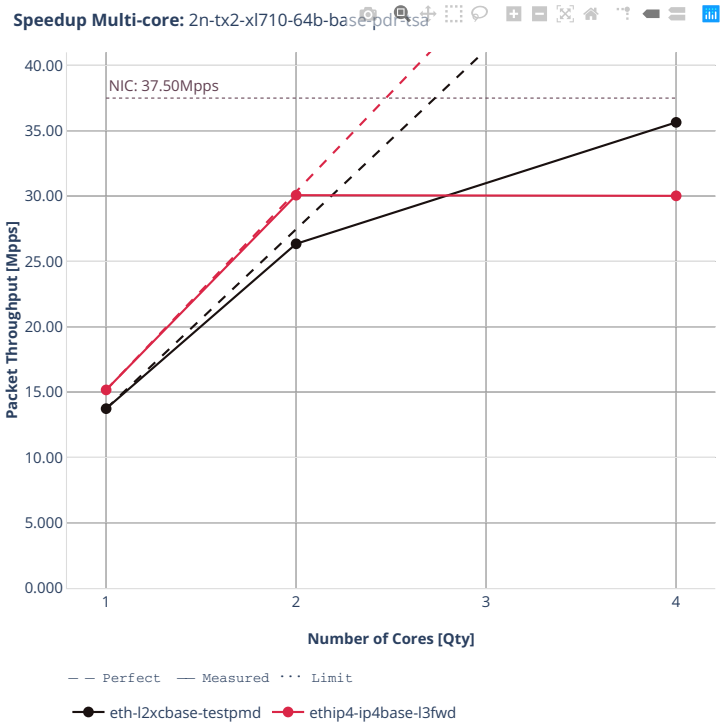
3.4.9 2n-tx2-xl710

CSIT source code for the test cases used for plots can be found in CSIT git repository[281].

[281] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 2n-tx2-xl710-64b-base-ndrpsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

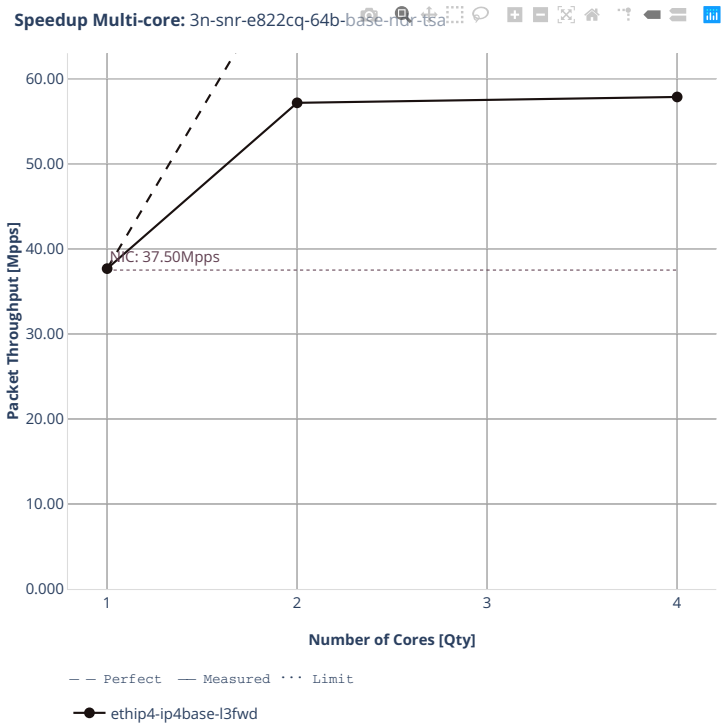

Speedup Multi-core: 2n-tx2-xl710-64b-base-pdr-tsa
NIC: 37.50Mpps
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.000
0.000
Packet Throughput [Mpps]
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
eth-l2xcbase-testpmd
ethip4-ip4base-l3fwd

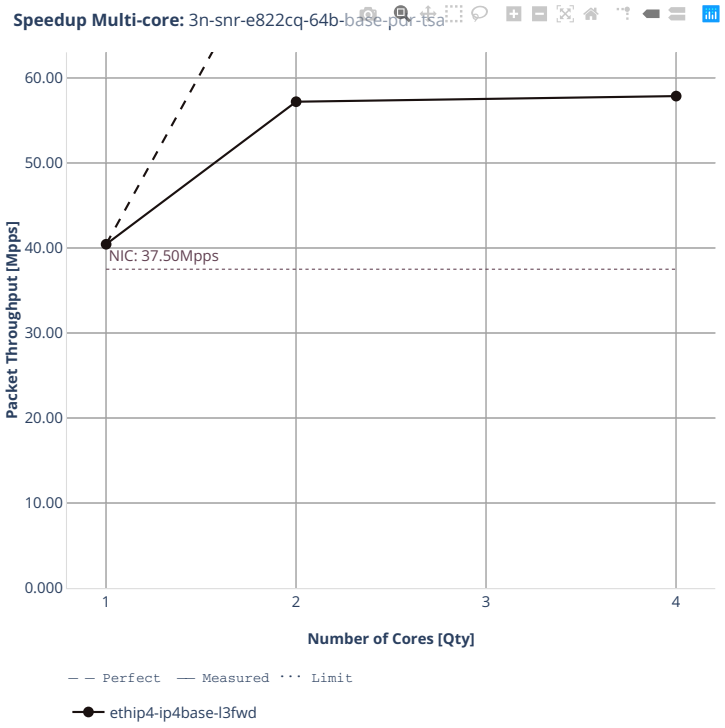
3.4.10 3n-snr-e822cq

CSIT source code for the test cases used for plots can be found in CSIT git repository[282].

[282] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-base


Speedup Multi-core: 3n-snr-e822cq-64b-base-mrr-tsa
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
NIC: 37.50Mpps
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base-l3fwd


Speedup Multi-core: 3n-snr-e822cq-64b-base-pdr-tsa
60.00
50.00
40.00
30.00
20.00
10.00
0.000
Packet Throughput [Mpps]
NIC: 37.50Mpps
1
2
3
4
Number of Cores [Qty]
Perfect
Measured
Limit
ethip4-ip4base-l3fwd

3.5 Packet Latency

DPDK Testpmd and L3fwd latency results are generated based on the test data obtained from CSIT-2210 NDR-PDR throughput tests executed across physical testbeds hosted in LF FD.io labs: 2n-icx, 3n-icx, 2n-clx, 2n-zn2, 3n-alt, 3n-tsh, 2n-tx2.

Latency by percentile distribution plots are used to show packet latency percentiles at different packet rate load levels: i) No-Load latency streams only, ii) Low-Load at 10% PDR, iii) Mid-Load at 50% PDR and iv) High-Load at 90% PDR.

For more details, see *Packet Latency* (page 43).

Additional information about graph data:

1. **Graph Title**: describes tested DUT packet path.
2. **X-axis Labels**: percentile of packets.
3. **Y-axis Labels**: measured one-way packet latency values in [uSec].
4. **Graph Legend**: list of latency tests at different packet rate load level.
5. **Hover Information**: packet rate load level, stream direction (East-West, West-East), percentile, one-way latency.

Note: Test results are stored in build logs from FD.io dpdk performance job 2n-icx[283], build logs from FD.io dpdk performance job 3n-icx[284], build logs from FD.io dpdk performance job 2n-clx[285], build logs from FD.io dpdk performance job 2n-zn2[286], build logs from FD.io dpdk performance job 3n-alt[287], build logs from FD.io dpdk performance job 3n-tsh[288] and build logs from FD.io dpdk performance job 2n-tx2[289] with RF result files csit-dpdk-perf-2210-*.zip archived here.

---

[283] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-icx
[284] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-icx
[285] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-clx
[286] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-zn2
[287] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-alt
[288] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-tsh
[289] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-tx2

3.5.1 2n-icx-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[290].

[290] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

3.5.2 3n-icx-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[291].

[291] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

3.5.3 2n-clx-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[292].

[292] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-2t1c-base

3.5.4 2n-zn2-xxv710

CSIT source code for the test cases used for plots can be found in CSIT git repository[293].

[293] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

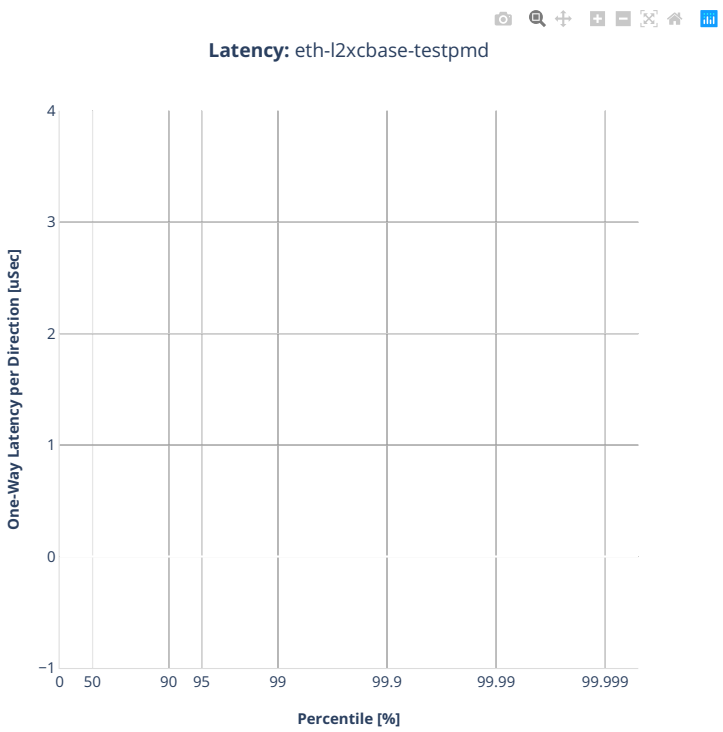
64b-2t1c-base

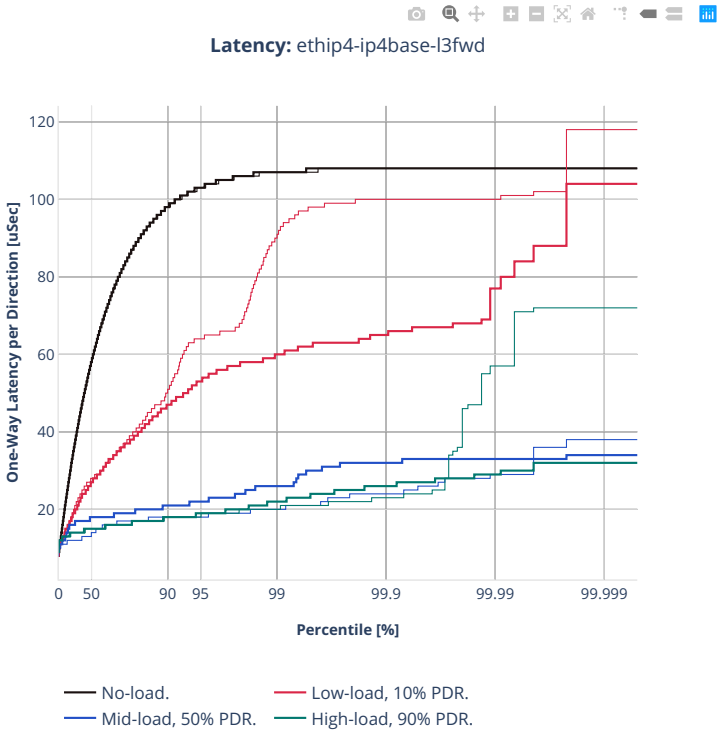
3.5.5 3n-alt-xl710

CSIT source code for the test cases used for plots can be found in CSIT git repository[294].

[294] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-1t1c-base



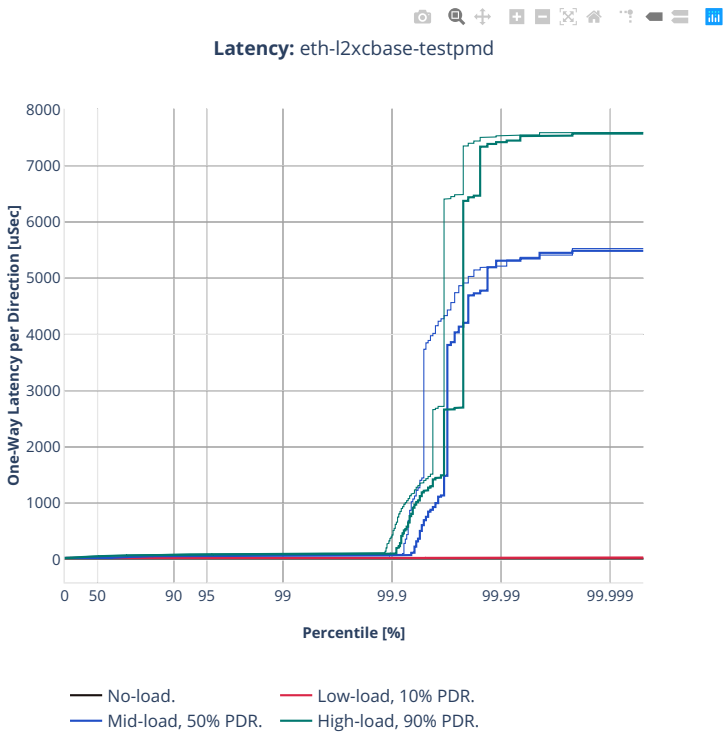

Latency: ethip4-ip4base-l3fwd
One-Way Latency per Direction [uSec]
120
100
80
60
40
20
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

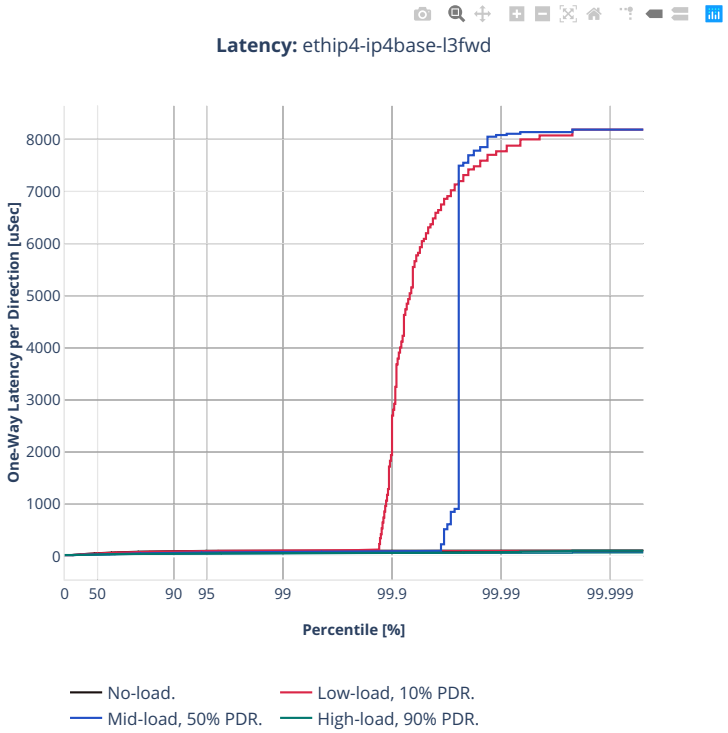
3.5.6 3n-tsh-x520

CSIT source code for the test cases used for plots can be found in CSIT git repository[295].

[295] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-1t1c-base



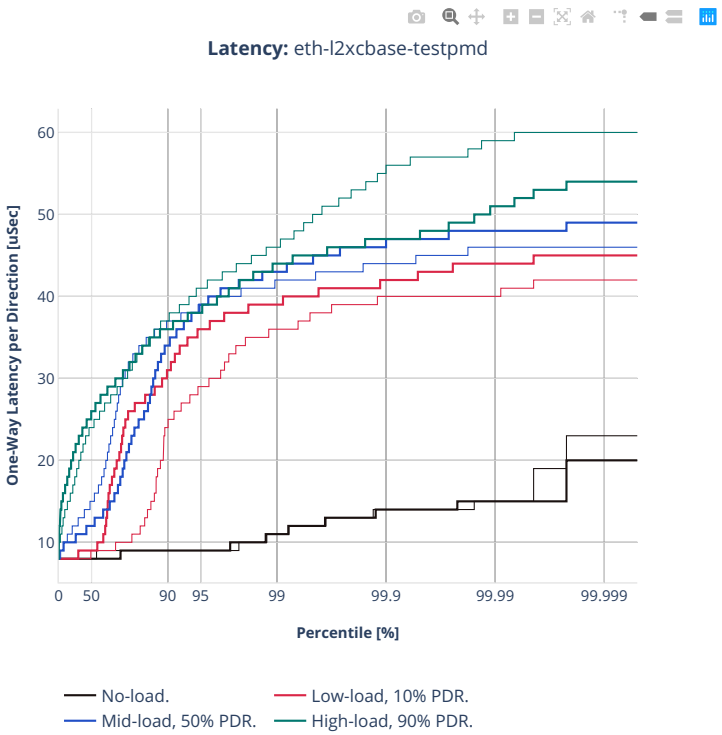

Latency: ethip4-ip4base-l3fwd
One-Way Latency per Direction [uSec]
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

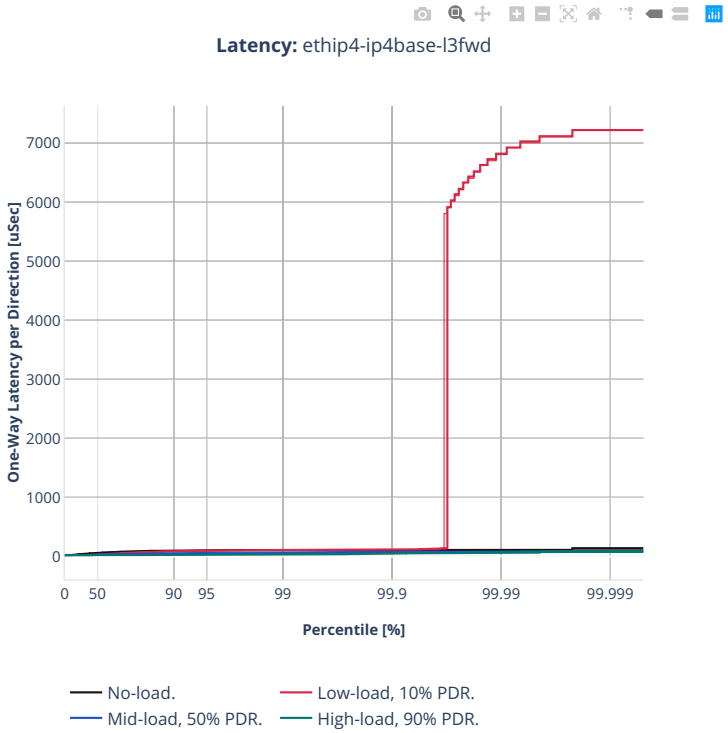
3.5.7 2n-tx2-xl710

CSIT source code for the test cases used for plots can be found in CSIT git repository[296].

[296] https://git.fd.io/csit/tree/tests/dpdk/perf?h=rls2210

64b-1t1c-base




Latency: ethip4-ip4base-l3fwd
One-Way Latency per Direction [uSec]
7000
6000
5000
4000
3000
2000
1000
0
0
50
90
95
99
99.9
99.99
99.999
Percentile [%]
No-load.
Low-load, 10% PDR.
Mid-load, 50% PDR.
High-load, 90% PDR.

3.6 Comparisons

3.6.1 Current vs Previous Release

Relative comparison of DPDK Testpmd and L3fwd packet throughput (NDR, PDR and MRR) between DPDK-22.07 and DPDK-22.03 (measured for CSIT-2210 and CSIT-2206 respectively) is calculated from results of tests running on 3n-icx, 2n-ics, 2n-clx, 2n-zn2, 3n-tsh, 2n-tx2 testbeds in 1-core and 2-core configurations.

Listed mean and standard deviation values are computed based on a series of the same tests executed against respective DPDK releases to verify test results repeatability, with percentage change calculated for mean values.

Note: Test results are stored in build logs from FD.io dpdk performance job 3n-icx[297], build logs from FD.io dpdk performance job 2n-icx[298], build logs from FD.io dpdk performance job 2n-clx[299], build logs from FD.io dpdk performance job 2n-zn2[300], build logs from FD.io dpdk performance job 3n-tsh[301], build logs from FD.io dpdk performance job 2n-tx2[302], with RF result files csit-dpdk-perf-2210-*.zip archived here.

3n-icx-xxv710

NDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

[297] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-icx
[298] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-icx
[299] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-clx
[300] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-zn2
[301] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-3n-tsh
[302] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-tx2

2n-icx-xxv710

NDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

2n-clx-xxv710

NDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

2n-zn2-xxv710

NDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c NDR comparison
- ASCII 4t2c NDR comparison
- CSV 2t1c NDR comparison
- CSV 4t2c NDR comparison

PDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 2t1c PDR comparison
- ASCII 4t2c PDR comparison
- CSV 2t1c PDR comparison
- CSV 4t2c PDR comparison

3n-tsh-x520

NDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 1t1c NDR comparison
- ASCII 2t2c NDR comparison
- CSV 1t1c NDR comparison
- CSV 2t2c NDR comparison

PDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 1t1c PDR comparison
- ASCII 2t2c PDR comparison
- CSV 1t1c PDR comparison
- CSV 2t2c PDR comparison

2n-tx2-xl710

NDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 1t1c NDR comparison
- ASCII 2t2c NDR comparison
- CSV 1t1c NDR comparison
- CSV 2t2c NDR comparison

PDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII 1t1c PDR comparison
- ASCII 2t2c PDR comparison
- CSV 1t1c PDR comparison
- CSV 2t2c PDR comparison

3.6.2 2n-Icx vs 2n-Clx Testbeds

Relative comparison of DPDK-22.07 Testpmd and L3fwd packet throughput (NDR, PDR) is calculated for the same tests executed on 2-Node Cascadelake (2n-clx) and 2-Node Ice Lake (2n-icx) physical testbed types, in 1-core, 2-core and 4-core configurations.

Note: Test results are stored in build logs from FD.io dpdk performance job 2n-clx[303] and build logs from FD.io dpdk performance job 2n-icx[304] with RF result files csit-dpdk-perf-2210-*.zip archived here.

NDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII NDR comparison
- CSV NDR comparison

PDR Comparison

Comparison tables in ASCII and CSV formats:

- ASCII PDR comparison
- CSV PDR comparison

[303] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-clx
[304] https://s3-logs.fd.io/vex-yul-rot-jenkins-1/csit-dpdk-perf-report-iterative-2210-2n-icx

3.7 Throughput Trending

In addition to reporting throughput comparison between DPDK releases, CSIT provides regular performance trending for DPDK release branches: C-Dash[305]

[305] http://csit.fd.io/trending/

3.8 Test Environment

3.8.1 Environment Versioning

CSIT test environment versioning has been introduced to track modifications of the test environment.

Any benchmark anomalies (progressions, regressions) between releases of a DUT application (e.g. VPP, DPDK), are determined by testing it in the same test environment, to avoid test environment changes clouding the picture. To beter distinguish impact of test environment changes, we also execute tests without any SUT (just with TRex TG sending packets over a link looping back to TG).

A mirror approach is introduced to determine benchmarking anomalies due to the test environment change. This is achieved by testing the same DUT application version between releases of CSIT test system. This works under the assumption that the behaviour of the DUT is deterministic under the test conditions.

CSIT test environment versioning scheme ensures integrity of all the test system components, including their HW revisions, compiled SW code versions and SW source code, within a specific CSIT version. Components included in the CSIT environment versioning include:

- **HW** Server hardware firmware and BIOS (motherboard, processsor, NIC(s), accelerator card(s)), tracked in CSIT branch.
- **Linux** Server Linux OS version and configuration, tracked in CSIT Reports.
- **TRex** TRex Traffic Generator version, drivers and configuration tracked in TG Settings.
- **CSIT** CSIT framework code tracked in CSIT release branches.

Following is the list of CSIT versions to date:

- Ver. 1 associated with CSIT rls1908 branch (HW[306], Linux[307], TRex[308], CSIT[309]).
- Ver. 2 associated with CSIT rls2001 branch (HW[310], Linux[311], TRex[312], CSIT[313]).
- Ver. 4 associated with CSIT rls2005 branch (HW[314], Linux[315], TRex[316], CSIT[317]).
- Ver. 5 associated with CSIT rls2009 branch (HW[318], Linux[319], TRex[320], CSIT[321]).
  - The main change is TRex data-plane core resource adjustments: increase from 7 to 8 cores and pinning cores to interfaces[322] for better TRex performance with symmetric traffic profiles.
- Ver. 6 associated with CSIT rls2101 branch (HW[323], Linux[324], TRex[325], CSIT[326]).
  - The main change is TRex version upgrade: increase from 2.82 to 2.86.

[306] https://git.fd.io/csit/tree/docs/lab?h=rls1908
[307] https://docs.fd.io/csit/rls1908/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[308] https://docs.fd.io/csit/rls1908/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[309] https://git.fd.io/csit/tree/?h=rls1908
[310] https://git.fd.io/csit/tree/docs/lab?h=rls2001
[311] https://docs.fd.io/csit/rls2001/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[312] https://docs.fd.io/csit/rls2001/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[313] https://git.fd.io/csit/tree/?h=rls2001
[314] https://git.fd.io/csit/tree/docs/lab?h=rls2005
[315] https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[316] https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[317] https://git.fd.io/csit/tree/?h=rls2005
[318] https://git.fd.io/csit/tree/docs/lab?h=rls2009
[319] https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[320] https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[321] https://git.fd.io/csit/tree/?h=rls2009
[322] https://gerrit.fd.io/r/c/csit/+/28184
[323] https://git.fd.io/csit/tree/docs/lab?h=rls2101
[324] https://docs.fd.io/csit/rls2101/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[325] https://docs.fd.io/csit/rls2101/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[326] https://git.fd.io/csit/tree/?h=rls2101

- Ver. 7 associated with CSIT rls2106 branch (HW[327], Linux[328], TRex[329], CSIT[330]).
  - TRex version upgrade: increase from 2.86 to 2.88.
  - Ubuntu upgrade from 18.04 LTS to 20.04.2 LTS.
- Ver. 8 associated with CSIT rls2110 branch (HW[331], Linux[332], TRex[333], CSIT[334]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
- Ver. 9 associated with CSIT rls2202 branch (HW[335], Linux[336], TRex[337], CSIT[338]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
- Ver. 10 associated with CSIT rls2206 branch (HW[339], Linux[340], TRex[341], CSIT[342]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
  - Mellanox 556A series firmware upgrade based on DPDK compatibility matrix.
  - Intel IceLake all core turbo frequency turned off. Current base frequency is 2.6GHz.
  - TRex version upgrade: increase from 2.88 to 2.97.
- Ver. 11 associated with CSIT rls2210 branch (HW[343], Linux[344], TRex[345], CSIT[346]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
  - Mellanox 556A series firmware upgrade based on DPDK compatibility matrix.
  - Ubuntu upgrade from 20.04.2 LTS to 22.04.1 LTS. (2n-dnv and 3n-dnv keeps the Ubuntu 20.04.2LTS as a part of decomission).
  - TRex version upgrade: increase from 2.97 to 3.00.

3.8.2 SUT Settings - Linux

System provisioning is done by combination of PXE boot unattented install and Ansible[347] described in CSIT Testbed Setup[348].

[327] https://git.fd.io/csit/tree/docs/lab?h=rls2106
[328] https://s3-docs.fd.io/csit/rls2106/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[329] https://s3-docs.fd.io/csit/rls2106/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[330] https://git.fd.io/csit/tree/?h=rls2106
[331] https://git.fd.io/csit/tree/docs/lab?h=rls2110
[332] https://s3-docs.fd.io/csit/rls2110/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[333] https://s3-docs.fd.io/csit/rls2110/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[334] https://git.fd.io/csit/tree/?h=rls2110
[335] https://git.fd.io/csit/tree/docs/lab?h=rls2202
[336] https://s3-docs.fd.io/csit/rls2202/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[337] https://s3-docs.fd.io/csit/rls2202/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[338] https://git.fd.io/csit/tree/?h=rls2202
[339] https://git.fd.io/csit/tree/docs/lab?h=rls2206
[340] https://s3-docs.fd.io/csit/rls2206/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[341] https://s3-docs.fd.io/csit/rls2206/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[342] https://git.fd.io/csit/tree/?h=rls2206
[343] https://git.fd.io/csit/tree/docs/lab?h=rls2210
[344] https://s3-docs.fd.io/csit/rls2210/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[345] https://s3-docs.fd.io/csit/rls2210/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[346] https://git.fd.io/csit/tree/?h=rls2210
[347] https://www.ansible.com
[348] https://git.fd.io/csit/tree/fdio.infra.ansible?h=rls2210

Linux Boot Parameters

- **isolcpus=<cpu number>-<cpu number>** used for all cpu cores apart from first core of each socket used for running VPP worker threads and Qemu/LXC processes https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt
- **intel_pstate=disable** - [X86] Do not enable intel_pstate as the default scaling driver for the supported processors. Intel P-State driver decide what P-state (CPU core power state) to use based on requesting policy from the cpufreq core. [X86 - Either 32-bit or 64-bit x86] https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt
- **nohz_full=<cpu number>-<cpu number>** - [KNL,BOOT] In kernels built with CONFIG_NO_HZ_FULL=y, set the specified list of CPUs whose tick will be stopped whenever possible. The boot CPU will be forced outside the range to maintain the timekeeping. The CPUs in this range must also be included in the rcu_nocbs= set. Specifies the adaptive-ticks CPU cores, causing kernel to avoid sending scheduling-clock interrupts to listed cores as long as they have a single runnable task. [KNL - Is a kernel start-up parameter, SMP - The kernel is an SMP kernel]. https://www.kernel.org/doc/Documentation/timers/NO_HZ.txt
- **rcu_nocbs** - [KNL] In kernels built with CONFIG_RCU_NOCB_CPU=y, set the specified list of CPUs to be no-callback CPUs, that never queue RCU callbacks (read-copy update). https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt
- **numa_balancing=disable** - [KNL,X86] Disable automatic NUMA balancing.
- **intel_iommu=enable** - [DMAR] Enable Intel IOMMU driver (DMAR) option.
- **iommu=on, iommu=pt** - [x86, IA-64] Disable IOMMU bypass, using IOMMU for PCI devices.
- **nmi_watchdog=0** - [KNL,BUGS=X86] Debugging features for SMP kernels. Turn hardlockup detector in nmi_watchdog off.
- **nosoftlockup** - [KNL] Disable the soft-lockup detector.
- **tsc=reliable** - Disable clocksource stability checks for TSC. [x86] reliable: mark tsc clocksource as reliable, this disables clocksource verification at runtime, as well as the stability checks done at bootup. Used to enable high-resolution timer mode on older hardware, and in virtualized environment.
- **hpet=disable** - [X86-32,HPET] Disable HPET and use PIT instead.

Hugepages Configuration

Huge pages are managed via sysctl configuration located in */etc/sysctl.d/90-csit.conf* on each testbed. Default huge page size is 2M. The exact amount of huge pages depends on testbed. All the values are defined in *Ansible inventory - hosts* files.

3.8.3 DUT Settings - DPDK

DPDK Version

DPDK-22.07

DPDK Compile Parameters

```
make install T=<arch>-<machine>-linuxapp-gcc -j
```

Testpmd Startup Configuration

Testpmd startup configuration changes per test case with different settings for *$$INT*, *$$CORES*, *$$RXQ*, *$$RXD* and max-pkt-len parameter if test is sending jumbo frames. Startup command template:

```
testpmd -v -l $$CORE_LIST -w $$INT1 -w $$INT2 --master-lcore 0 --in-memory -- --forward-
↪mode=io --burst=64 --txd=$$TXD --rxd=$$RXD --txq=$$TXQ --rxq=$$RXQ --tx-offloads=0x0 --
↪numa --auto-start --total-num-mbufs=16384 --nb-ports=2 --portmask=0x3 --disable-link-
↪check --max-pkt-len=$$PKT_LEN [--mbuf-size=16384] --nb-cores=$$CORES
```

L3FWD Startup Configuration

L3FWD startup configuration changes per test case with different settings for *$$INT*, *$$CORES* and enable-jumbo parameter if test is sending jumbo frames. Startup command template:

```
l3fwd -v -l $$CORE_LIST -w $$INT1 -w $$INT2 --master-lcore 0 --in-memory -- --parse-ptype␣
↪--eth-dest="0,${adj_mac0}" --eth-dest="1,${adj_mac1}" --config="${port_config}" [--
↪enable-jumbo] -P -L -p 0x3
```

3.8.4 TG Settings - TRex

TG Version

TRex v3.00

DPDK Version

DPDK v21.02

TG Installation

T-Rex installation is managed via Ansible role.

TG Startup Configuration

```
$ sudo -E -S sh -c 'cat << EOF > /etc/trex_cfg.yaml
- version: 2
  c: 8
  limit_memory: 8192
  interfaces: ["${pci1}","${pci2}"]
  port_info:
      - dest_mac: [${dest_mac1}]
        src_mac: [${src_mac1}]
      - dest_mac: [${dest_mac2}]
        src_mac: [${src_mac2}]
  platform :
      master_thread_id: 0
```

(continues on next page)

(continued from previous page)

```
      latency_thread_id: 9
      dual_if:
          - socket: 0
            threads: [1, 2, 3, 4, 5, 6, 7, 8]
EOF'
```

TG Startup Command (Stateless Mode)

```
$ sudo -E -S sh -c "cd '${trex_install_dir}/scripts/' && \
  nohup ./t-rex-64 -i --prefix $(hostname) --hdrh --no-scapy-server \
  --mbuf-factor 32 > /tmp/trex.log 2>&1 &" > /dev/null
```

Also, Python client is now starting traffic with:

```
core_mask=STLClient.CORE_MASK_PIN
```

TG Startup Command (Stateful Mode)

```
$ sudo -E -S sh -c "cd '${trex_install_dir}/scripts/' && \
  nohup ./t-rex-64 -i --prefix $(hostname) --astf --hdrh --no-scapy-server \
  --mbuf-factor 32 > /tmp/trex.log 2>&1 &" > /dev/null
```

TG API Driver

TRex driver[349]

3.8.5 Pre-Test Server Calibration

Number of SUT server sub-system runtime parameters have been identified as impacting data plane performance tests. Calibrating those parameters is part of FD.io CSIT pre-test activities, and includes measuring and reporting following:

1. System level core jitter - measure duration of core interrupts by Linux in clock cycles and how often interrupts happen. Using CPU core jitter tool[350].
2. Memory bandwidth - measure bandwidth with Intel MLC tool[351].
3. Memory latency - measure memory latency with Intel MLC tool.
4. Cache latency at all levels (L1, L2, and Last Level Cache) - measure cache latency with Intel MLC tool.

Measured values of listed parameters are especially important for repeatable zero packet loss throughput measurements across multiple system instances. Generally they come useful as a background data for comparing data plane performance results across disparate servers.

Following sections include measured calibration data for testbeds.

[349] https://git.fd.io/csit/tree/GPL/tools/trex/trex_stl_profile.py?h=rls2210
[350] https://git.fd.io/pma_tools/tree/jitter
[351] https://software.intel.com/en-us/articles/intelr-memory-latency-checker

Ice Lake

Following sections include sample calibration data measured on server running in one of the Intel Xeon Ice Lake testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=6ff26c8a-8c65-4025-a6e7-d97dee6025d0␣
↪ro audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_
↪pstate=disable iommu=pt isolcpus=1-31,33-63,65-95,97-127 mce=off nmi_watchdog=0 nohz_
↪full=1-31,33-63,65-95,97-127 nosoftlockup numa_balancing=disable processor.max_cstate=1␣
↪rcu_nocbs=1-31,33-63,65-95,97-127 tsc=reliable console=ttyS0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 3 /home/testuser/pma_tools/jitter/jitter -i 30
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
160022,167912,7890,160034,160022,167912,854327296,3203987030,1
160022,168114,8092,160042,160022,168114,4234936320,3204004240,2
160022,168386,8364,160040,160022,168386,3320578048,3204007496,3
160022,169432,9410,160028,160022,169432,2406219776,3204213462,4
160022,168050,8028,160040,160022,169432,1491861504,3203982428,5
160022,166384,6362,160040,160022,169432,577503232,3203969006,6
```

(continues on next page)

(continued from previous page)

```
160022,168962,8940,160042,160022,169432,3958112256,3204002514,7
160020,169248,9228,160038,160020,169432,3043753984,3204208318,8
160022,168854,8832,160038,160020,169432,2129395712,3203987894,9
160022,166754,6732,160042,160020,169432,1215037440,3203984104,10
160022,168208,8186,160040,160020,169432,300679168,3203980640,11
160022,172450,12428,160040,160020,172450,3681288192,3204208216,12
160022,168244,8222,160042,160020,172450,2766929920,3204037074,13
160022,166894,6872,160040,160020,172450,1852571648,3203979376,14
160022,169068,9046,160038,160020,172450,938213376,3204009714,15
160020,168528,8508,160036,160020,172450,23855104,3204028382,16
160022,169458,9436,160042,160020,172450,3404464128,3204179220,17
160020,167056,7036,160040,160020,172450,2490105856,3203990218,18
160022,167038,7016,160038,160020,172450,1575747584,3203976712,19
160022,168610,8588,160040,160020,172450,661389312,3204025230,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several speculative execution CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[352].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (Intel SSBD)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: YES (L1D flush feature bit)
  * Microarchitectural Data Sampling
    * VERW instruction is available: YES (MD_CLEAR feature bit)
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: YES
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: YES
```

(continues on next page)

[352] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): YES
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): YES
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): YES
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): YES
    * TSX_CTRL MSR indicates TSX RTM is disabled: YES
    * TSX_CTRL MSR indicates TSX CPUID bit is cleared: YES
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): YES
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x6a␣
↪stepping 0x6 ucode 0xd000280 cpuid 0x606a6)
  * CPU microcode is the latest known available version: NO (latest version is 0xd000331␣
↪dated 2021/12/03 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): YES
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): YES
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): YES
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): YES
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Enhanced IBRS, IBPB:␣
↪conditional, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your kernel reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Cascade Lake

Following sections include sample calibration data measured on server running in one of the Intel Xeon Skylake testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=2d6f4d44-76b1-4343-bc73-c066a3e95b32␣
↪ro audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_
↪pstate=disable iommu=pt isolcpus=1-23,25-47,49-71,73-95 mce=off nmi_watchdog=0 nohz_
↪full=1-23,25-47,49-71,73-95 nosoftlockup numa_balancing=disable processor.max_cstate=1␣
↪rcu_nocbs=1-23,25-47,49-71,73-95 tsc=reliable console=ttyS0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 3 /home/testuser/pma_tools/jitter/jitter -i 30
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
160026,167568,7542,160032,160026,167568,183238656,3204033176,1
160026,171174,11148,160028,160026,171174,3563847680,3204142488,2
160024,170002,9978,160032,160024,171174,2649489408,3204224288,3
160026,169124,9098,160032,160024,171174,1735131136,3204142126,4
160026,169096,9070,160030,160024,171174,820772864,3204069082,5
160026,168788,8762,160028,160024,171174,4201381888,3204056954,6
```

(continues on next page)

(continued from previous page)

```
160024,169196,9172,160030,160024,171174,3287023616,3204364824,7
160026,168176,8150,160028,160024,171174,2372665344,3204073670,8
160026,169466,9440,160032,160024,171174,1458307072,3204068092,9
160026,168858,8832,160032,160024,171174,543948800,3204109862,10
160026,169418,9392,160028,160024,171174,3924557824,3204289508,11
160026,167776,7750,160032,160024,171174,3010199552,3204089538,12
160024,170538,10514,160032,160024,171174,2095841280,3204109170,13
160026,169320,9294,160034,160024,171174,1181483008,3204108772,14
160026,169976,9950,160034,160024,171174,267124736,3204259754,15
160026,166826,6800,160030,160024,171174,3647733760,3204058488,16
160026,168314,8288,160032,160024,171174,2733375488,3204110518,17
160026,170176,10150,160028,160024,171174,1819017216,3204283146,18
160024,168698,8674,160030,160024,171174,904658944,3204162904,19
160026,168234,8208,160034,160024,171174,4285267968,3204059562,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several speculative execution CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[353].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is Intel(R) Xeon(R) Gold 6252N CPU @ 2.30GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (Intel SSBD)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: YES (L1D flush feature bit)
  * Microarchitectural Data Sampling
    * VERW instruction is available: YES (MD_CLEAR feature bit)
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: YES
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: YES
```

(continues on next page)

[353] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): YES
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): NO
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): NO
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): YES
    * TSX_CTRL MSR indicates TSX RTM is disabled: YES
    * TSX_CTRL MSR indicates TSX CPUID bit is cleared: YES
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x55␣
↪stepping 0x7 ucode 0x500002c cpuid 0x50657)
  * CPU microcode is the latest known available version: NO (latest version is 0x500320a␣
↪dated 2021/08/13 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): YES
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): YES
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): YES
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Enhanced IBRS, IBPB:␣
↪conditional, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB+RSB filling, is needed to mitigate the␣
↪vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your kernel reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Mitigation: TSX disabled)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (KVM: Mitigation: VMX disabled)
> STATUS: NOT VULNERABLE (KVM: Mitigation: VMX disabled)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

EPYC Zen2

Following sections include sample calibration data measured on server running in one of the AMD EPYC testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=cac1254f-9426-4ea6-a8db-2554f075db99␣
↪ro amd_iommu=on audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 hpet=disable iommu=pt isolcpus=1-15,17-31,33-47,49-63 nmi_watchdog=0␣
↪nohz_full=off nosoftlockup numa_balancing=disable processor.max_cstate=0 rcu_nocbs=1-15,
↪17-31,33-47,49-63 tsc=reliable console=ttyS0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux s60-t210-sut1 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64␣
↪x86_64 x86_64 GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 3 /home/testuser/pma_tools/jitter/jitter -i 30
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
116400,145848,29448,116400,116400,145848,2076377088,2375383296,1
116400,145848,29448,116400,116400,145848,388169728,2363555544,2
116400,145848,29448,116400,116400,145848,2994929664,2359881480,3
116400,145848,29448,116400,116400,145848,1306722304,2367487104,4
116400,145848,29448,116400,116400,145848,3913482240,2357721768,5
116400,145848,29448,116400,116400,145848,2225274880,2381723112,6
116400,145848,29448,116424,116400,145848,537067520,2373138432,7
```

(continues on next page)

(continued from previous page)

```
116400,145848,29448,116424,116400,145848,3143827456,2372221464,8
116400,145848,29448,116400,116400,145848,1455620096,2365450272,9
116400,145848,29448,116400,116400,145848,4062380032,2364814440,10
116400,145848,29448,116400,116400,145848,2374172672,2375992608,11
116400,145848,29448,116400,116400,145848,685965312,2362608552,12
116400,145848,29448,116400,116400,145848,3292725248,2362597944,13
116400,145848,29448,145512,116400,145848,1604517888,2370049344,14
116400,145848,29448,116400,116400,145848,4211277824,2366291784,15
116400,145848,29448,116400,116400,145848,2523070464,2349077352,16
116400,145848,29448,116400,116400,145848,834863104,2375406360,17
116400,145848,29448,116400,116400,145848,3441623040,2373272976,18
116400,145848,29448,116400,116400,145848,1753415680,2382267192,19
116400,145848,29448,116400,116400,145848,65208320,2359406040,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several speculative execution CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[354].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is AMD EPYC 7532 32-Core Processor

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (IBRS_SUPPORT feature bit)
    * CPU indicates preferring IBRS always-on: NO
    * CPU indicates preferring IBRS over retpoline: YES
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (IBPB_SUPPORT feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (AMD STIBP feature bit)
    * CPU indicates preferring STIBP always-on: NO
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (AMD SSBD in SPEC_CTRL)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: NO
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x17 model 0x31␣
↪stepping 0x0 ucode 0x8301038 cpuid 0x830f10)
  * CPU microcode is the latest known available version: NO (latest version is 0x8301052␣
↪dated 2021/11/11 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
```

(continues on next page)

[354] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): NO
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Retpolines, IBPB:␣
↪conditional, IBRS_FW, STIBP: always-on, RSB filling)
> STATUS: VULNERABLE (retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Denverton

Following sections include sample calibration data measured on server running in one of the Intel Atom Denverton testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.0-65-generic root=UUID=26ca7b0f-904a-462d-a1c6-98c420c29515␣
↪ro audit=0 hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_pstate=disable␣
↪iommu=pt isolcpus=1-5 mce=off nmi_watchdog=0 nohz_full=1-5 nosoftlockup numa_
↪balancing=disable processor.max_cstate=1 rcu_nocbs=1-5 tsc=reliable console=tty0␣
↪console=ttyS0,115200n8
```

Linux uname

```
$ uname -a
Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 2 /home/testuser/pma_tools/jitter/jitter -c 2 -i 20
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number
Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
177008,217292,40284,177552,177008,217292,80543744,3555521762,1
167862,222370,54508,177552,167862,222370,191692800,3555482758,2
172576,251932,79356,177538,167862,251932,302841856,3556013278,3
177368,215300,37932,177552,167862,251932,413990912,3555428816,4
167914,215066,47152,177552,167862,251932,525139968,3555415700,5
177494,241748,64254,177552,167862,251932,636289024,3555835494,6
177038,210186,33148,177552,167862,251932,747438080,3555398164,7
170956,211022,40066,177552,167862,251932,858587136,3555435464,8
174130,237428,63298,177552,167862,251932,969736192,3555771752,9
174726,205252,30526,177552,167862,251932,1080885248,3555426516,10
177104,234502,57398,177554,167862,251932,1192034304,3555785760,11
175304,240416,65112,177550,167862,251932,1303183360,3555908234,12
166674,216176,49502,177552,166674,251932,1414332416,3555468016,13
177532,205792,28260,177552,166674,251932,1525481472,3555440968,14
177516,235032,57516,177550,166674,251932,1636630528,3555832414,15
177522,207292,29770,177552,166674,251932,1747779584,3555495058,16
177532,205174,27642,177552,166674,251932,1858928640,3555458754,17
177528,234230,56702,177552,166674,251932,1970077696,3555837046,18
177530,209364,31834,177552,166674,251932,2081226752,3555469590,19
177530,205002,27472,177552,166674,251932,2192375808,3555397840,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[355].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64
CPU is Intel(R) Atom(TM) CPU C3858 @ 2.00GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: NO
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: NO
  * Microarchitectural Data Sampling
    * VERW instruction is available: NO
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: NO
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: NO
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): NO
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): NO
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): NO
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): NO
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x5f␣
↪stepping 0x1 ucode 0x20 cpuid 0x506f1)
  * CPU microcode is the latest known available version: NO (latest version is 0x36 dated␣
↪2021/05/10 according to builtin firmwares DB v222+i20220208)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
```

(continues on next page)

[355] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Full generic retpoline,␣
↪IBPB: conditional, IBRS_FW, STIBP: disabled, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: VULNERABLE (an up-to-date CPU microcode is needed to mitigate this␣
↪vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: NO (Vulnerable)
> STATUS: VULNERABLE (Neither your CPU nor your kernel support SSBD)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:KO CVE-2018-
↪3639:KO CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Snowridge

Following sections include sample calibration data measured on server running in one of the Intel Atom Snowridge testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-5.15.0-46-generic root=/dev/mapper/ubuntu--vg-ubuntu--lv ro audit=0␣
↪default_hugepagesz=2M hugepagesz=1G hugepages=2 hugepagesz=2M hugepages=4096␣
↪hpet=disable intel_idle.max_cstate=1 intel_iommu=on intel_pstate=disable iommu=pt␣
↪isolcpus=1-23 mce=off nmi_watchdog=0 nohz_full=1-23 nosoftlockup numa_balancing=disable␣
↪processor.max_cstate=1 rcu_nocbs=1-23 tsc=reliable console=ttyS0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

System-level Core Jitter

```
$ sudo taskset -c 2 /home/testuser/pma_tools/jitter/jitter -c 2 -i 20
Linux Jitter testing program version 1.9
Iterations=20
The pragram will execute a dummy function 80000 times
Display is updated every 20000 displayUpdate intervals
Thread affinity will be set to core_id:7
Timings are in CPU Core cycles
Inst_Min:    Minimum Excution time during the display update interval(default is ~1␣
↪second)
Inst_Max:    Maximum Excution time during the display update interval(default is ~1␣
↪second)
Inst_jitter: Jitter in the Excution time during rhe display update interval. This is the␣
↪value of interest
last_Exec:   The Excution time of last iteration just before the display update
Abs_Min:     Absolute Minimum Excution time since the program started or statistics were␣
↪reset
Abs_Max:     Absolute Maximum Excution time since the program started or statistics were␣
↪reset
tmp:         Cumulative value calcualted by the dummy function
Interval:    Time interval between the display updates in Core Cycles
Sample No:   Sample number

Inst_Min,Inst_Max,Inst_jitter,last_Exec,Abs_min,Abs_max,tmp,Interval,Sample No
160370,165364,4994,160380,160370,165364,1042874368,3211228620,1
160370,165308,4938,160430,160370,165364,1279852544,3211283594,2
160370,169968,9598,160394,160370,169968,1516830720,3211446352,3
160370,166026,5656,160430,160370,169968,1753808896,3211263720,4
160370,165516,5146,160414,160370,169968,1990787072,3211249674,5
160370,165594,5224,160448,160370,169968,2227765248,3211267504,6
160370,169988,9618,160374,160370,169988,2464743424,3211426160,7
160370,165384,5014,160382,160370,169988,2701721600,3211243706,8
160370,165514,5144,160444,160370,169988,2938699776,3211233152,9
160370,168954,8584,160392,160370,169988,3175677952,3211338334,10
160370,167270,6900,160374,160370,169988,3412656128,3211329846,11
160370,165430,5060,160408,160370,169988,3649634304,3211240244,12
160370,166196,5826,160398,160370,169988,3886612480,3211256920,13
160370,169678,9308,160398,160370,169988,4123590656,3211415892,14
160370,165718,5348,160418,160370,169988,65601536,3211259448,15
160370,165256,4886,160372,160370,169988,302579712,3211236834,16
160370,167840,7470,160382,160370,169988,539557888,3211260000,17
160370,169332,8962,160400,160370,169988,776536064,3211432972,18
160370,165272,4902,160428,160370,169988,1013514240,3211246698,19
160370,165906,5536,160398,160370,169988,1250492416,3211262146,20
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[356].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:03:25 UTC 2022 x86_64
CPU is Intel Atom(R) P5362 processor

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (Intel SSBD)
  * L1 data cache invalidation
    * CPU indicates L1D flush capability: YES (L1D flush feature bit)
  * Microarchitectural Data Sampling
    * VERW instruction is available: YES (MD_CLEAR feature bit)
  * Indirect Branch Predictor Controls
    * Indirect Predictor Disable feature is available: NO
    * Bottomless RSB Disable feature is available: NO
    * BHB-Focused Indirect Predictor Disable feature is available: NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: YES
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: YES
  * CPU explicitly indicates not being affected by Meltdown/L1TF (RDCL_NO): YES
  * CPU explicitly indicates not being affected by Variant 4 (SSB_NO): NO
  * CPU/Hypervisor indicates L1D flushing is not necessary on this system: YES
  * Hypervisor indicates host CPU might be affected by RSB underflow (RSBA): NO
  * CPU explicitly indicates not being affected by Microarchitectural Data Sampling (MDS_
↪NO): YES
  * CPU explicitly indicates not being affected by TSX Asynchronous Abort (TAA_NO): NO
  * CPU explicitly indicates not being affected by iTLB Multihit (PSCHANGE_MSC_NO): YES
  * CPU explicitly indicates having MSR for TSX control (TSX_CTRL_MSR): NO
  * CPU supports Transactional Synchronization Extensions (TSX): NO
  * CPU supports Software Guard Extensions (SGX): NO
  * CPU supports Special Register Buffer Data Sampling (SRBDS): NO
  * CPU microcode is known to cause stability problems: NO (family 0x6 model 0x86␣
↪stepping 0x7 ucode 0x4c000019 cpuid 0x80667)
  * CPU microcode is the latest known available version: UNKNOWN (latest microcode␣
↪version for your CPU model is unknown)
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
```

(continues on next page)

[356] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): YES
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: usercopy/swapgs barriers␣
↪and __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: Enhanced IBRS, IBPB:␣
↪conditional, RSB filling)
> STATUS: VULNERABLE (IBRS+IBPB or retpoline+IBPB is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: YES
> STATUS: NOT VULNERABLE (your CPU microcode mitigates the vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl and seccomp)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl and␣
↪seccomp)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
```

(continues on next page)

(continued from previous page)

```
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (Not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

Altra

Following sections include sample calibration data measured on server running in one of the Altra testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.15.0-46-generic root=UUID=7d1d0e77-4df0-43df-9619-a99db29ffb83␣
↪ro audit=0 default_hugepagesz=2M hugepagesz=1G hugepages=32 hugepagesz=2M␣
↪hugepages=32768 iommu.passthrough=1 isolcpus=1-10,29-38 nmi_watchdog=0 nohz_full=1-10,
↪29-38 nosoftlockup processor.max_cstate=1 rcu_nocbs=1-10,29-38 console=ttyAMA0,115200n8␣
↪quiet
```

Linux uname

```
$ uname -a
Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:08:11 UTC 2022 aarch64 aarch64␣
↪aarch64 GNU/Linux
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[357].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:08:11 UTC 2022 aarch64
CPU is ARM v8 model 0xd0c

Hardware check
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): NO
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): NO
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: YES (Mitigation: CSV2, BHB)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)
```

(continues on next page)

[357] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: YES (Mitigation: Speculative Store Bypass␣
↪disabled via prctl)
> STATUS: NOT VULNERABLE (Mitigation: Speculative Store Bypass disabled via prctl)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:OK CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:OK CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

TaiShan

Following sections include sample calibration data measured on s17-t33-sut1 server running in one of the Cortex-A72 testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.0-65-generic root=UUID=7d1d0e77-4df0-43df-9619-a99db29ffb83␣
↪ro audit=0 intel_iommu=on isolcpus=1-27,29-55 nmi_watchdog=0 nohz_full=1-27,29-55␣
↪nosoftlockup processor.max_cstate=1 rcu_nocbs=1-27,29-55 console=ttyAMA0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[358].

```
Spectre and Meltdown mitigation detection tool v0.45

Checking for vulnerabilities on current system
Kernel is Linux 5.15.0-46-generic #49-Ubuntu SMP Thu Aug 4 18:08:11 UTC 2022 aarch64
CPU is ARM v8 model 0xd08

Hardware check
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): YES
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
```

(continues on next page)

[358] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: __user pointer sanitization)
> STATUS: UNKNOWN (/sys vulnerability interface use forced, but its not available!)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: NO (Vulnerable)
> STATUS: VULNERABLE (Branch predictor hardening is needed to mitigate the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: VULNERABLE (an up-to-date CPU microcode is needed to mitigate this␣
↪vulnerability)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: NO (Vulnerable)
> STATUS: VULNERABLE (Neither your CPU nor your kernel support SSBD)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)
```

(continues on next page)

(continued from previous page)

```
CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not affected)

> SUMMARY: CVE-2017-5753:?? CVE-2017-5715:KO CVE-2017-5754:OK CVE-2018-3640:KO CVE-2018-
↪3639:KO CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

ThunderX2

Following sections include sample calibration data measured on s27-t211-sut1 server running in one of the ThunderX2 testbeds.

Linux cmdline

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.0-65-generic root=UUID=7d1d0e77-4df0-43df-9619-a99db29ffb83␣
↪ro audit=0 intel_iommu=on isolcpus=1-27,29-55 nmi_watchdog=0 nohz_full=1-27,29-55␣
↪nosoftlockup processor.max_cstate=1 rcu_nocbs=1-27,29-55 console=ttyAMA0,115200n8 quiet
```

Linux uname

```
$ uname -a
Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:25:17 UTC 2021 x86_64 x86_64 x86_64␣
↪GNU/Linux
```

Spectre and Meltdown Checks

Following section displays the output of a running shell script to tell if system is vulnerable against the several "speculative execution" CVEs that were made public in 2018. Script is available on Spectre & Meltdown Checker Github[359].

```
Spectre and Meltdown mitigation detection tool v0.44+

Checking for vulnerabilities on current system
Kernel is Linux 5.4.0-65-generic #73-Ubuntu SMP Mon Jan 18 17:27:25 UTC 2021 aarch64
CPU is

Hardware check
* CPU vulnerability to the speculative execution attack variants
  * Affected by CVE-2017-5753 (Spectre Variant 1, bounds check bypass): YES
  * Affected by CVE-2017-5715 (Spectre Variant 2, branch target injection): YES
  * Affected by CVE-2017-5754 (Variant 3, Meltdown, rogue data cache load): NO
```

(continues on next page)

[359] https://github.com/speed47/spectre-meltdown-checker

(continued from previous page)

```
  * Affected by CVE-2018-3640 (Variant 3a, rogue system register read): NO
  * Affected by CVE-2018-3639 (Variant 4, speculative store bypass): YES
  * Affected by CVE-2018-3615 (Foreshadow (SGX), L1 terminal fault): NO
  * Affected by CVE-2018-3620 (Foreshadow-NG (OS), L1 terminal fault): NO
  * Affected by CVE-2018-3646 (Foreshadow-NG (VMM), L1 terminal fault): NO
  * Affected by CVE-2018-12126 (Fallout, microarchitectural store buffer data sampling␣
↪(MSBDS)): NO
  * Affected by CVE-2018-12130 (ZombieLoad, microarchitectural fill buffer data sampling␣
↪(MFBDS)): NO
  * Affected by CVE-2018-12127 (RIDL, microarchitectural load port data sampling␣
↪(MLPDS)): NO
  * Affected by CVE-2019-11091 (RIDL, microarchitectural data sampling uncacheable memory␣
↪(MDSUM)): NO
  * Affected by CVE-2019-11135 (ZombieLoad V2, TSX Asynchronous Abort (TAA)): NO
  * Affected by CVE-2018-12207 (No eXcuses, iTLB Multihit, machine check exception on␣
↪page size changes (MCEPSC)): NO
  * Affected by CVE-2020-0543 (Special Register Buffer Data Sampling (SRBDS)): NO

CVE-2017-5753 aka Spectre Variant 1, bounds check bypass
* Mitigated according to the /sys interface: YES (Mitigation: __user pointer sanitization)
* Kernel has array_index_mask_nospec: NO
* Kernel has the Red Hat/Ubuntu patch: NO
* Kernel has mask_nospec64 (arm64): NO
* Kernel has array_index_nospec (arm64): NO
* Checking count of LFENCE instructions following a jump in kernel... NO (only 0 jump-
↪then-lfence instructions found, should be >= 30 (heuristic))
> STATUS: NOT VULNERABLE (Mitigation: __user pointer sanitization)

CVE-2017-5715 aka Spectre Variant 2, branch target injection
* Mitigated according to the /sys interface: NO (Vulnerable)
* Mitigation 1
  * Kernel is compiled with IBRS support: YES
    * IBRS enabled and active: NO
  * Kernel is compiled with IBPB support: NO
    * IBPB enabled and active: NO
* Mitigation 2
  * Kernel has branch predictor hardening (arm): YES
  * Kernel compiled with retpoline option: NO
> STATUS: NOT VULNERABLE (Branch predictor hardening mitigates the vulnerability)

CVE-2017-5754 aka Variant 3, Meltdown, rogue data cache load
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports Page Table Isolation (PTI): YES
  * PTI enabled and active: UNKNOWN (dmesg truncated, please reboot and relaunch this␣
↪script)
  * Reduced performance impact of PTI: NO (PCID/INVPCID not supported, performance impact␣
↪of PTI will be significant)
* Running as a Xen PV DomU: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3640 aka Variant 3a, rogue system register read
* CPU microcode mitigates the vulnerability: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3639 aka Variant 4, speculative store bypass
* Mitigated according to the /sys interface: NO (Vulnerable)
```

(continues on next page)

(continued from previous page)

```
* Kernel supports disabling speculative store bypass (SSB): YES (found in /proc/self/
↪status)
* SSB mitigation is enabled and active: NO
> STATUS: VULNERABLE (Your CPU doesnt support SSBD)

CVE-2018-3615 aka Foreshadow (SGX), L1 terminal fault
* CPU microcode mitigates the vulnerability: N/A
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3620 aka Foreshadow-NG (OS), L1 terminal fault
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports PTE inversion: NO
* PTE inversion enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-3646 aka Foreshadow-NG (VMM), L1 terminal fault
* Information from the /sys interface: Not affected
* This system is a host running a hypervisor: NO
* Mitigation 1 (KVM)
  * EPT is disabled: N/A (the kvm_intel module is not loaded)
* Mitigation 2
  * L1D flush is supported by kernel: NO
  * L1D flush enabled: NO
  * Hardware-backed L1D flush supported: NO (flush will be done in software, this is␣
↪slower)
  * Hyper-Threading (SMT) is enabled: UNKNOWN
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12126 aka Fallout, microarchitectural store buffer data sampling (MSBDS)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12130 aka ZombieLoad, microarchitectural fill buffer data sampling (MFBDS)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12127 aka RIDL, microarchitectural load port data sampling (MLPDS)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2019-11091 aka RIDL, microarchitectural data sampling uncacheable memory (MDSUM)
* Mitigated according to the /sys interface: YES (Not affected)
* Kernel supports using MD_CLEAR mitigation: NO
* Kernel mitigation is enabled and active: NO
* SMT is either mitigated or disabled: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)
```

(continues on next page)

(continued from previous page)

```
CVE-2019-11135 aka ZombieLoad V2, TSX Asynchronous Abort (TAA)
* Mitigated according to the /sys interface: YES (Not affected)
* TAA mitigation is supported by kernel: YES (found tsx_async_abort in kernel image)
* TAA mitigation enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2018-12207 aka No eXcuses, iTLB Multihit, machine check exception on page size␣
↪changes (MCEPSC)
* Mitigated according to the /sys interface: YES (Not affected)
* This system is a host running a hypervisor: NO
* iTLB Multihit mitigation is supported by kernel: YES (found itlb_multihit in kernel␣
↪image)
* iTLB Multihit mitigation enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

CVE-2020-0543 aka Special Register Buffer Data Sampling (SRBDS)
* Mitigated according to the /sys interface: YES (Not affected)
* SRBDS mitigation control is supported by the kernel: NO
* SRBDS mitigation control is enabled and active: NO
> STATUS: NOT VULNERABLE (your CPU vendor reported your CPU model as not vulnerable)

> SUMMARY: CVE-2017-5753:OK CVE-2017-5715:OK CVE-2017-5754:OK CVE-2018-3640:OK CVE-2018-
↪3639:KO CVE-2018-3615:OK CVE-2018-3620:OK CVE-2018-3646:OK CVE-2018-12126:OK CVE-2018-
↪12130:OK CVE-2018-12127:OK CVE-2019-11091:OK CVE-2019-11135:OK CVE-2018-12207:OK CVE-
↪2020-0543:OK
```

CHAPTER

FOUR

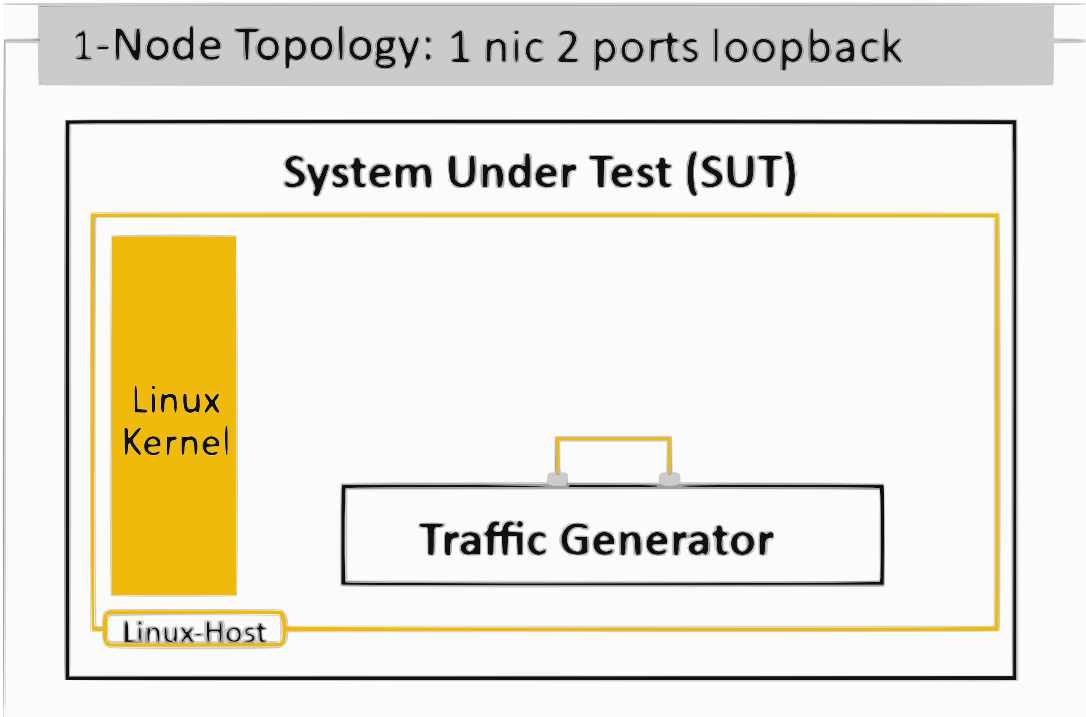
TREX PERFORMANCE

4.1 Overview

TRex performance test results are reported for a range of processors. For description of physical testbeds used for TRex performance tests please refer to *Performance Physical Testbeds* (page 4).

4.1.1 Logical Topology

CSIT TRex performance tests are executed on physical testbeds described in *Performance Physical Testbeds* (page 4). Logical topology use 1 nic that has loopback connected ports. See figure below.



4.1.2 Performance Tests Coverage

Performance tests measure following metrics for tested TRex topologies and configurations:

- Packet Throughput: measured in accordance with RFC 2544[360], using FD.io CSIT Multiple Loss Ratio search (MLRsearch), an optimized binary search algorithm, producing throughput at different Packet Loss Ratio (PLR) values:
  - Non Drop Rate (NDR): packet throughput at PLR=0%.
  - Partial Drop Rate (PDR): packet throughput at PLR=0.5%.
- Two-way Packet Latency: measured both east-west and west-east at different offered packet loads:
  - 90% of discovered PDR throughput.
  - 50% of discovered PDR throughput.
  - 10% of discovered PDR throughput.
  - Minimal offered load.

CSIT-2210 includes tests using the following TRex traffic profiles (corresponding to data plane functionality when DUT is used) performance tested across a range of NIC drivers and NIC models:

Traffic profile	Corresponding dataplane functionality
IPv4 Base	IPv4 routing.
IPv4 Scale	IPv4 routing with 2M entries.
IPv6 Base	IPv6 routing.
IPv6 Scale	IPv6 routing with 2M entries.
L2BD Scale	L2 Bridge-Domain switching of untagged Ethernet frames.

4.1.3 Performance Tests Naming

FD.io CSIT-2210 follows a common structured naming convention for all performance and system functional tests.

The naming should be intuitive for majority of the tests. Complete description of FD.io CSIT test naming convention is provided on *Test Naming* (page 1430).

4.2 Release Notes

4.2.1 Changes in CSIT-2210

1. TREX PERFORMANCE TESTS
   - **Intel Ice Lake**: Added tests for testing latency between 2 ports on Intel-E810Cq on the TRex. Used 2n-icx test bed. Added tests:
     - IP4Base
     - IP4scale2m
     - IP6Base
     - IP6scale2m
     - L2bscale1mmaclrn
   - **Amazon 1n-aws**: Added tests for testing latency between 2 ports on Amazon Nitro 50G on the TRex. Added tests:

[360] https://datatracker.ietf.org/doc/html/rfc2544.html

  - IP4Base
  - IP4scale2m
  - IP6Base
  - IP6scale2m

2. TEST FRAMEWORK
   - **CSIT test environment** version has been updated to ver. 11, see *Environment Versioning* (page 1206).

4.2.2 Known Issues

List of known issues in CSIT-2210 for TRex performance tests:

#	JiraID	Issue Description
1	CSIT-1876[361]	1n-aws: TRex NDR PDR ALL IP4 scale and L2 scale tests failing with 50% packet loss. CSIT removed ip4scale and l2scale except ip4scale2m where it's still failing.

[361] https://jira.fd.io/browse/CSIT-1876

4.3 Packet Throughput

Throughput graphs are generated by multiple executions of the same performance tests across physical testbeds hosted LF FD.io labs: 2n-skx. Box-and-Whisker plots are used to display variations in measured throughput values, without making any assumptions of the underlying statistical distribution.

For each test case, Box-and-Whisker plots show the quartiles (Min, 1st quartile / 25th percentile, 2nd quartile / 50th percentile / mean, 3rd quartile / 75th percentile, Max) across collected data set. Outliers are plotted as individual points.

Additional information about graph data:

1. **Graph Title**: describes tested packet path, testbed topology, processor model, NIC model, packet size, number of cores and threads used by data plane workers and indication of DPDK DUT configuration.
2. **X-axis Labels**: indices of individual test suites as listed in Graph Legend.
3. **Y-axis Labels**: measured Packets Per Second [pps] throughput values.
4. **Graph Legend**: lists X-axis indices with associated CSIT test suites executed to generate graphed test results.
5. **Hover Information**: lists minimum, first quartile, median, third quartile, and maximum. If either type of outlier is present the whisker on the appropriate side is taken to 1.5×IQR from the quartile (the "inner fence") rather than the max or min, and individual outlying data points are displayed as unfilled circles (for suspected outliers) or filled circles (for outliers). (The "outer fence" is 3×IQR from the quartile.)

Note: Test results are stored in **`build logs from FD.io trex performance job 2n-skx`_**. Required per test case data set size is **10** and for TRex tests this is the actual size, as all scheduled test executions completed successfully.

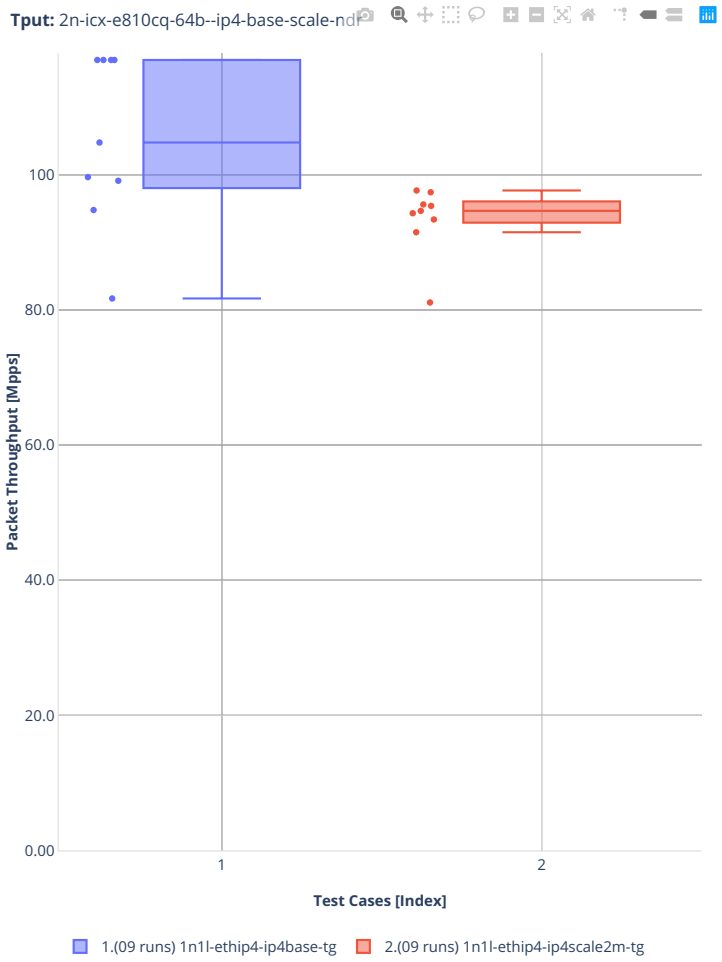
4.3.1 2n-icx-e810cq

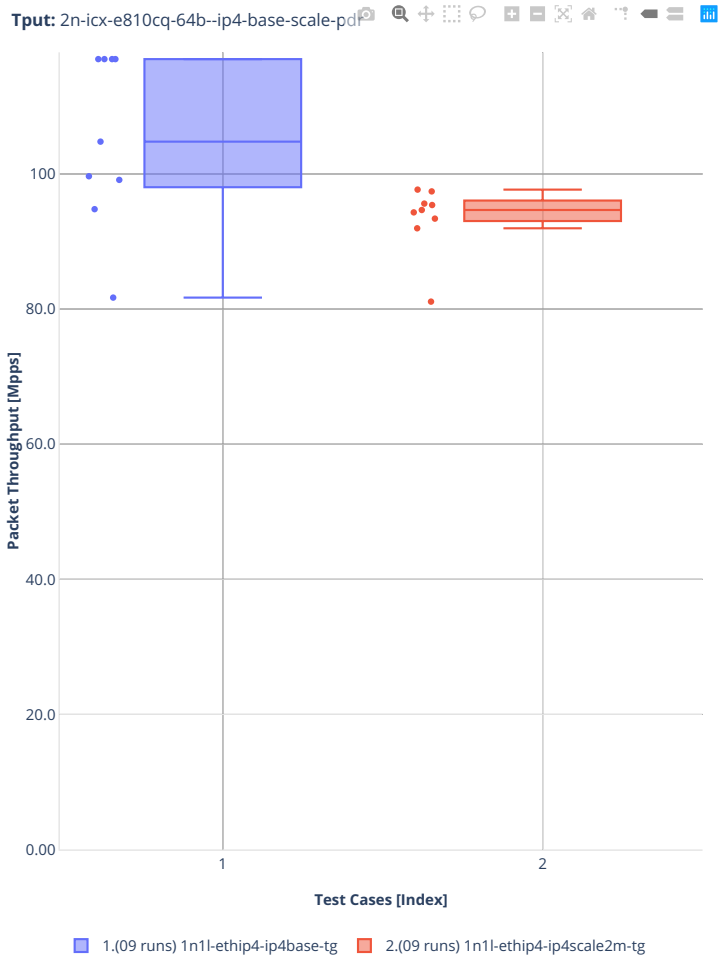
Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[362].

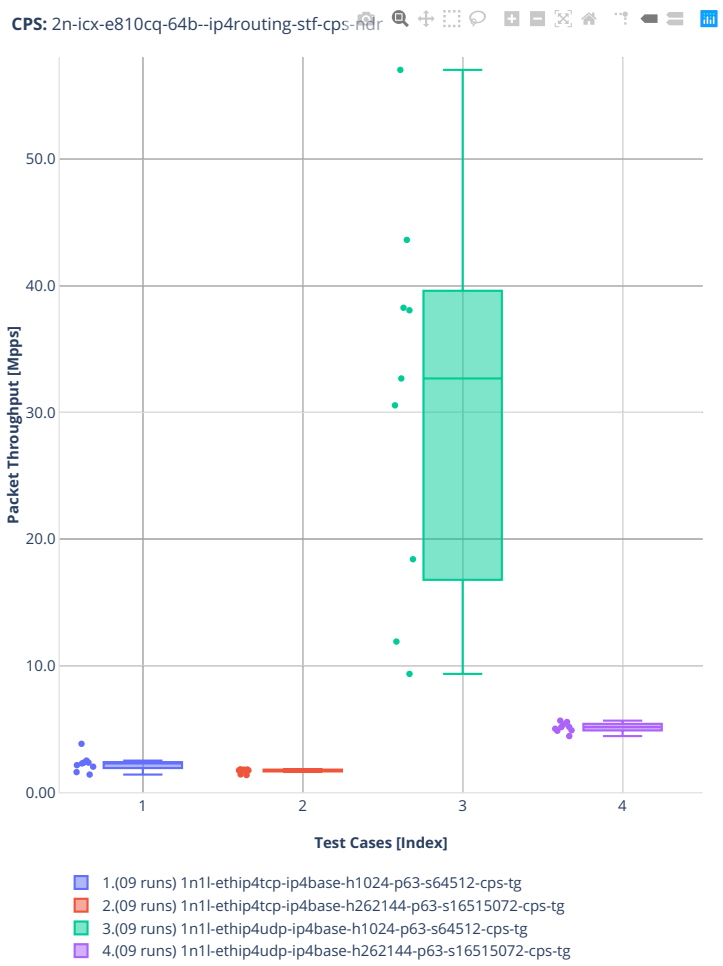
[362] https://git.fd.io/csit/tree/tests/trex/perf?h=rls2210

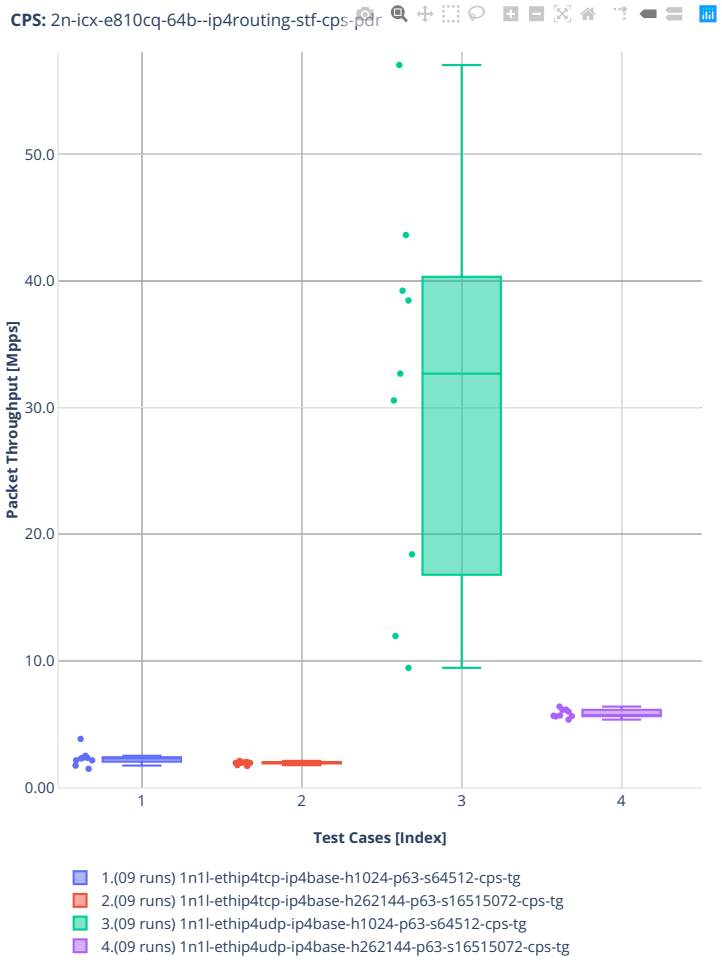
64b-ip4routing-base-scale



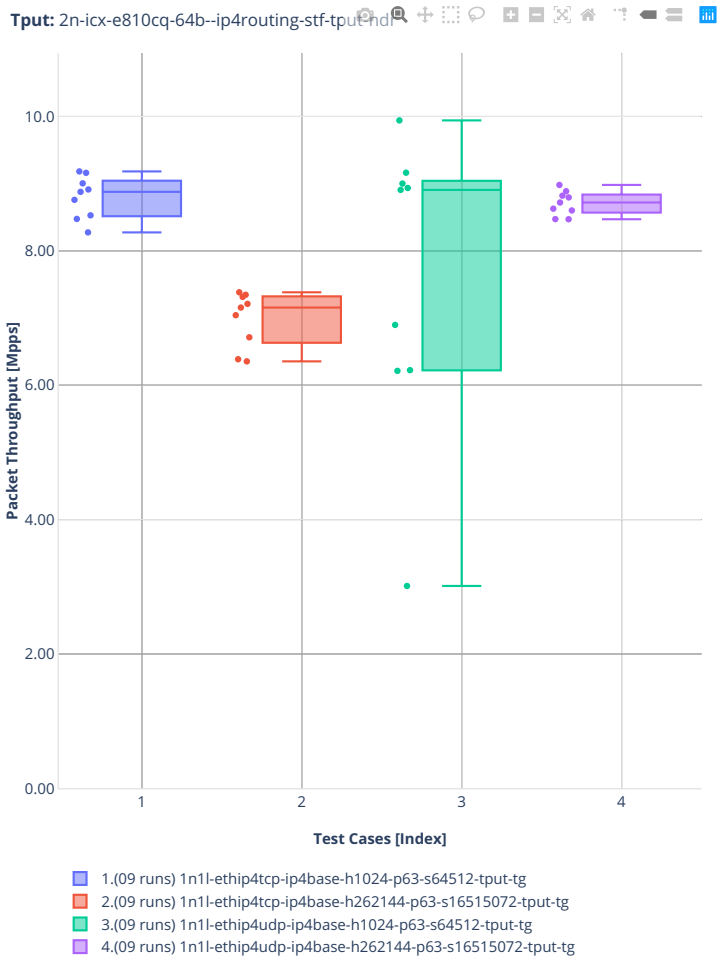

Tput: 2n-icx-e810cq-64b--ip4-base-scale-pdr
100
80.0
60.0
40.0
20.0
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(09 runs) 1n1l-ethip4-ip4base-tg
2.(09 runs) 1n1l-ethip4-ip4scale2m-tg

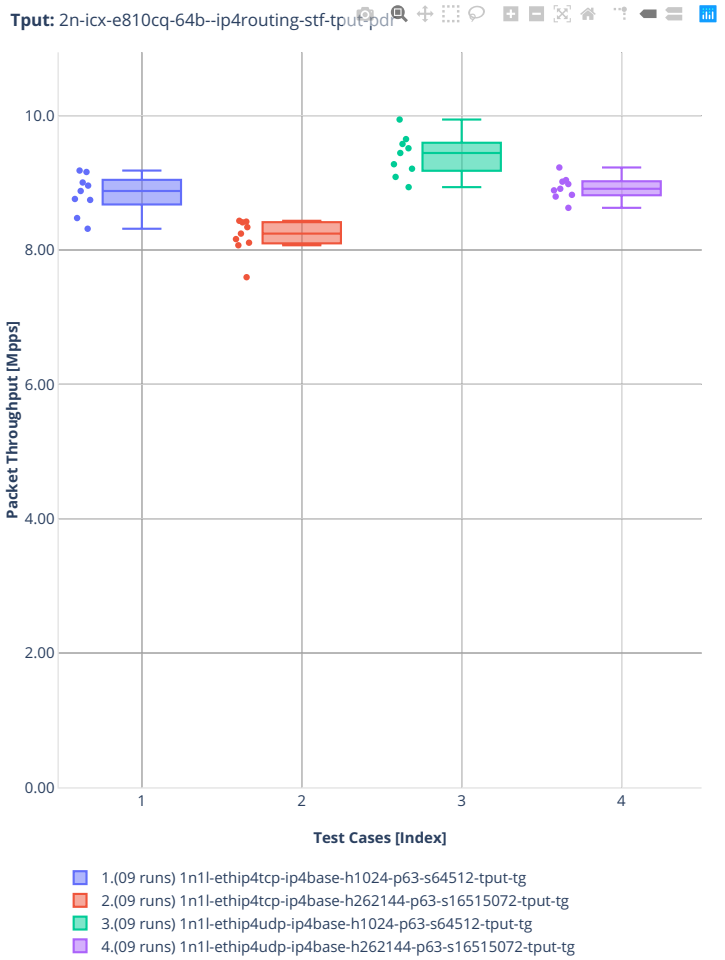
64b-ip4routing-[udp|tcp]-stf-cps



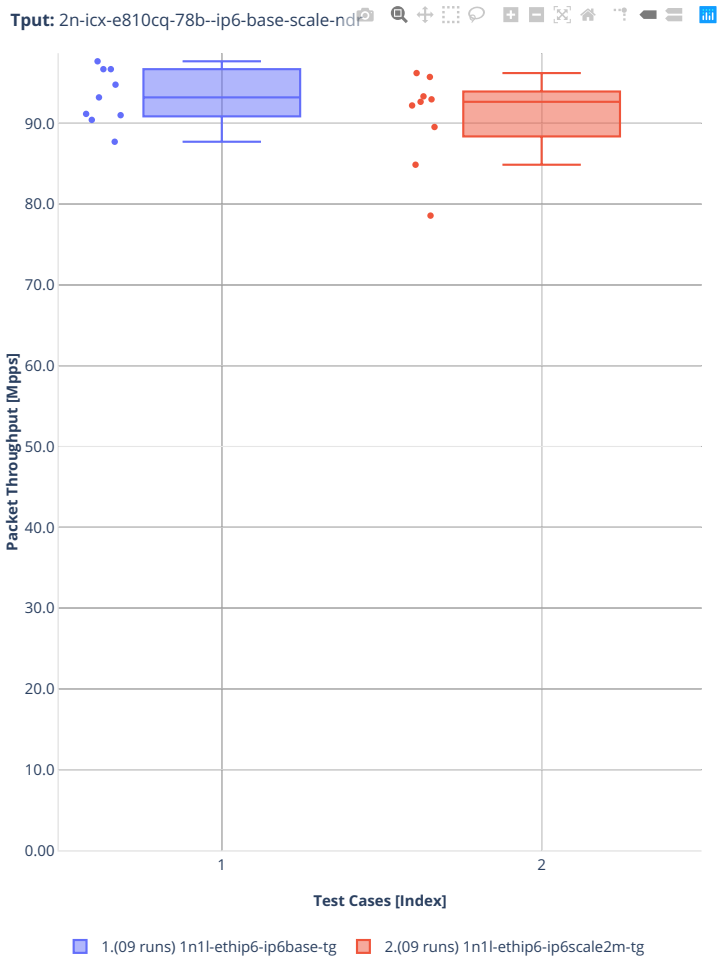

CPS: 2n-icx-e810cq-64b--ip4routing-stf-cps-pdr
Packet Throughput [Mpps]
50.0
40.0
30.0
20.0
10.0
0.00
1
2
3
4
Test Cases [Index]
1.(09 runs) 1n1l-ethip4tcp-ip4base-h1024-p63-s64512-cps-tg
2.(09 runs) 1n1l-ethip4tcp-ip4base-h262144-p63-s16515072-cps-tg
3.(09 runs) 1n1l-ethip4udp-ip4base-h1024-p63-s64512-cps-tg
4.(09 runs) 1n1l-ethip4udp-ip4base-h262144-p63-s16515072-cps-tg

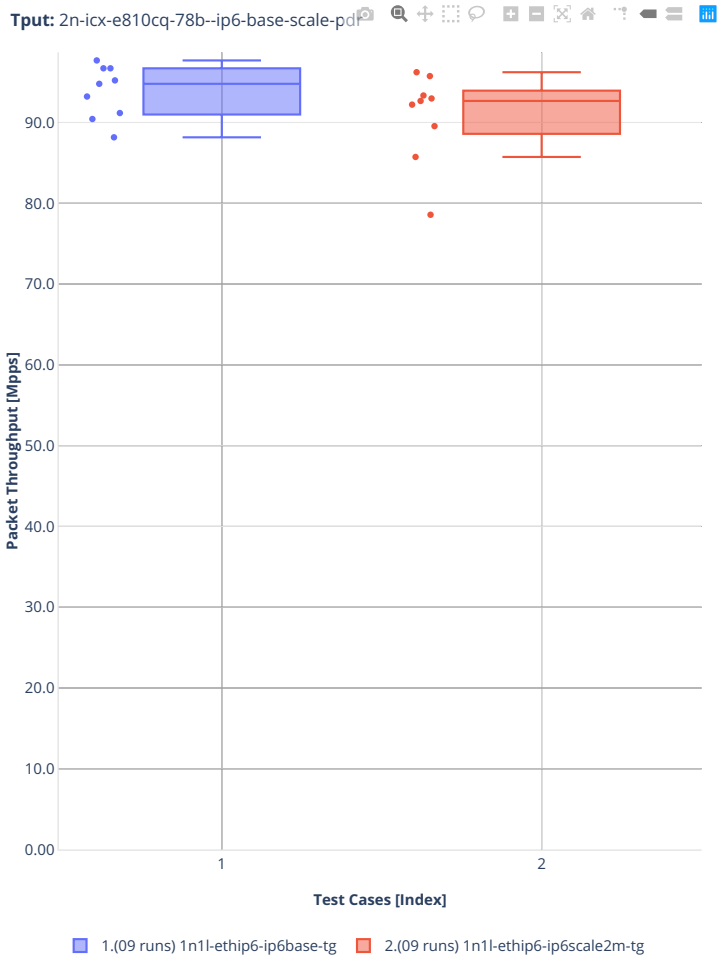
64b-ip4routing-[udp|tcp]-stf-tput



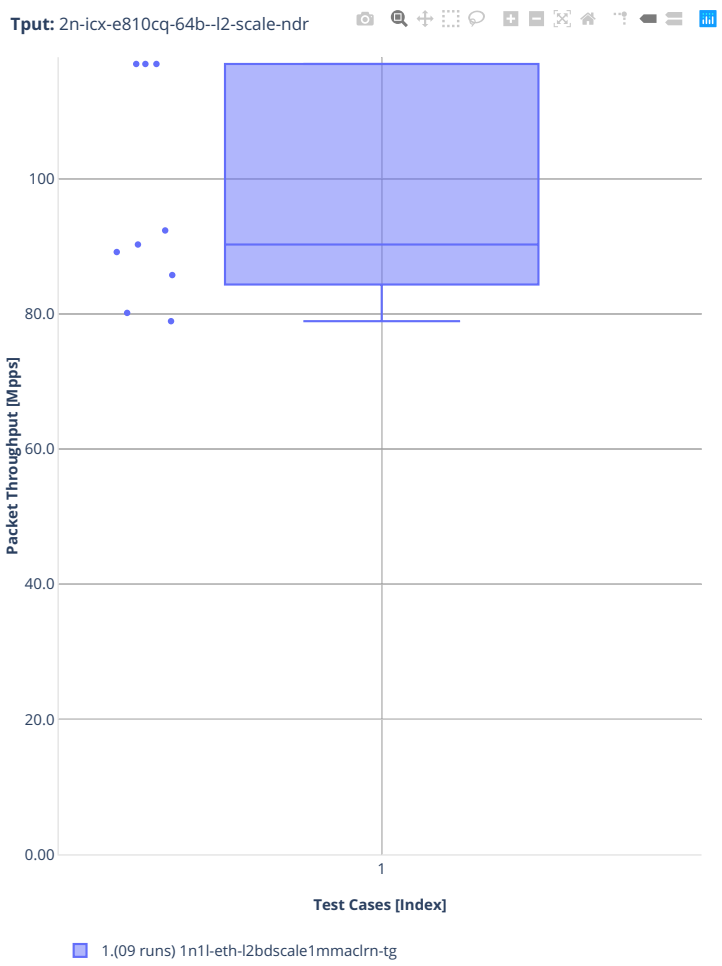

Tput: 2n-icx-e810cq-64b--ip4routing-stf-tput-pdr
10.0
8.00
6.00
4.00
2.00
0.00
Packet Throughput [Mpps]
1
2
3
4
Test Cases [Index]
1.(09 runs) 1n1l-ethip4tcp-ip4base-h1024-p63-s64512-tput-tg
2.(09 runs) 1n1l-ethip4tcp-ip4base-h262144-p63-s16515072-tput-tg
3.(09 runs) 1n1l-ethip4udp-ip4base-h1024-p63-s64512-tput-tg
4.(09 runs) 1n1l-ethip4udp-ip4base-h262144-p63-s16515072-tput-tg

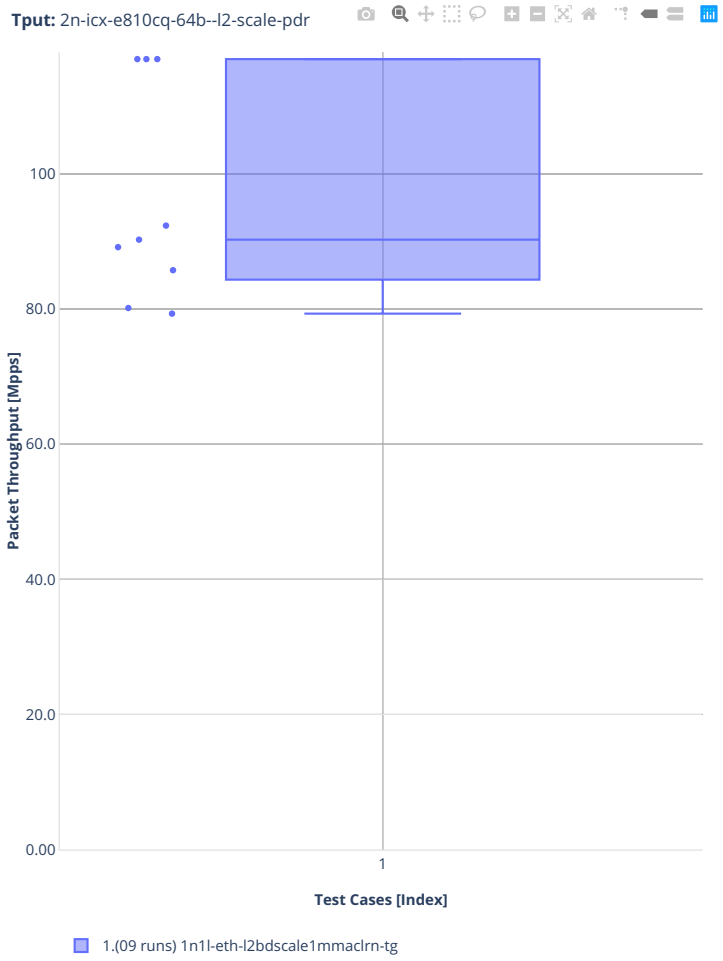
78b-ip6routing-base-scale




Tput: 2n-icx-e810cq-78b--ip6-base-scale-pdr
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(09 runs) 1n1l-ethip6-ip6base-tg
2.(09 runs) 1n1l-ethip6-ip6scale2m-tg

64b-l2switching-scale




Tput: 2n-icx-e810cq-64b--l2-scale-pdr
100
80.0
60.0
40.0
20.0
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
1.(09 runs) 1n1l-eth-l2bdscale1mmaclrn-tg

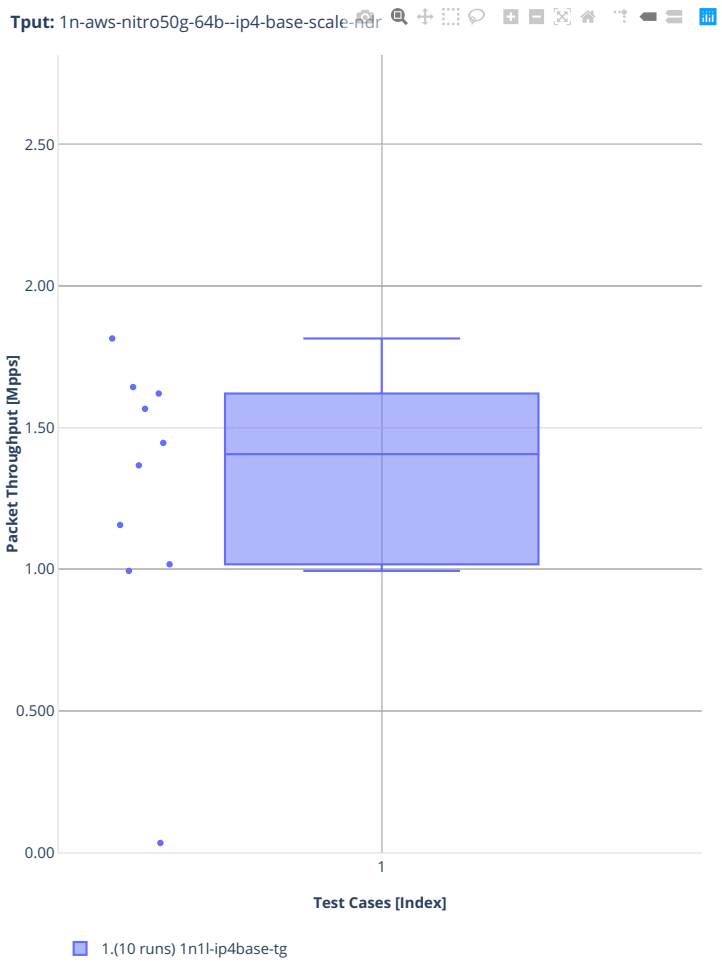
4.3.2 1n-aws-nitro50g

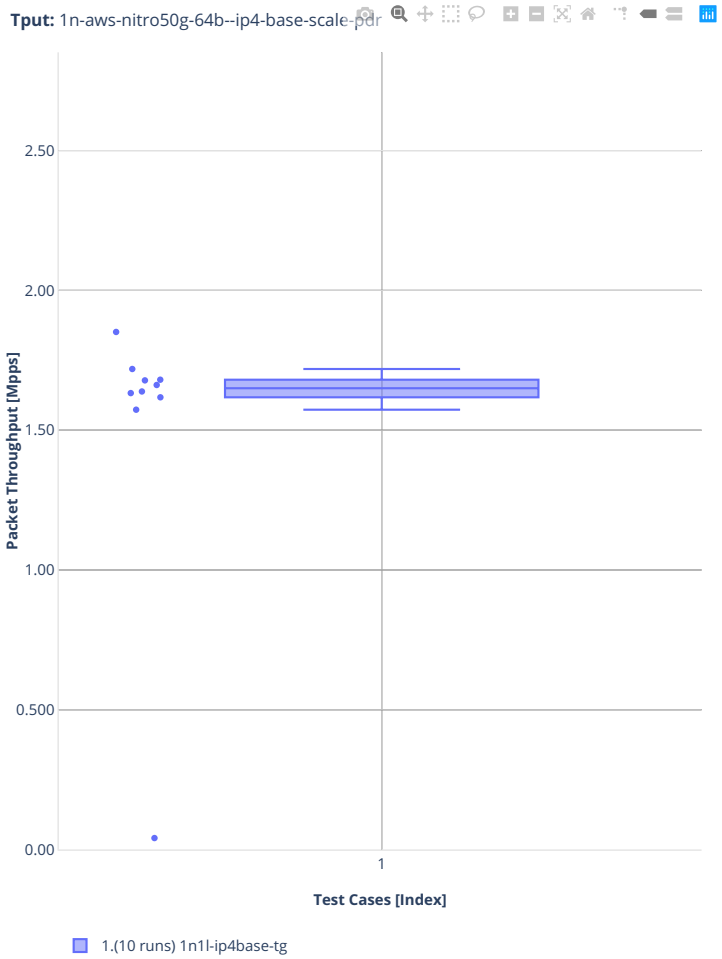
Following sections include summary graphs of Phy-to-Phy performance with packet routed forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

CSIT source code for the test cases used for plots can be found in CSIT git repository[363].

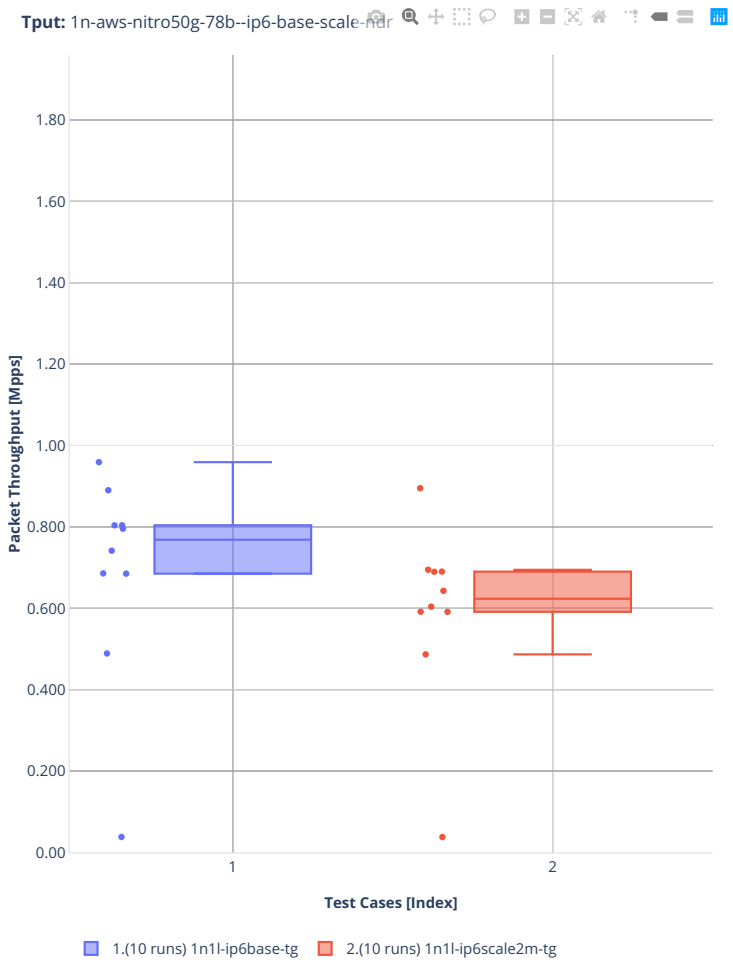
[363] https://git.fd.io/csit/tree/tests/trex/perf?h=rls2210

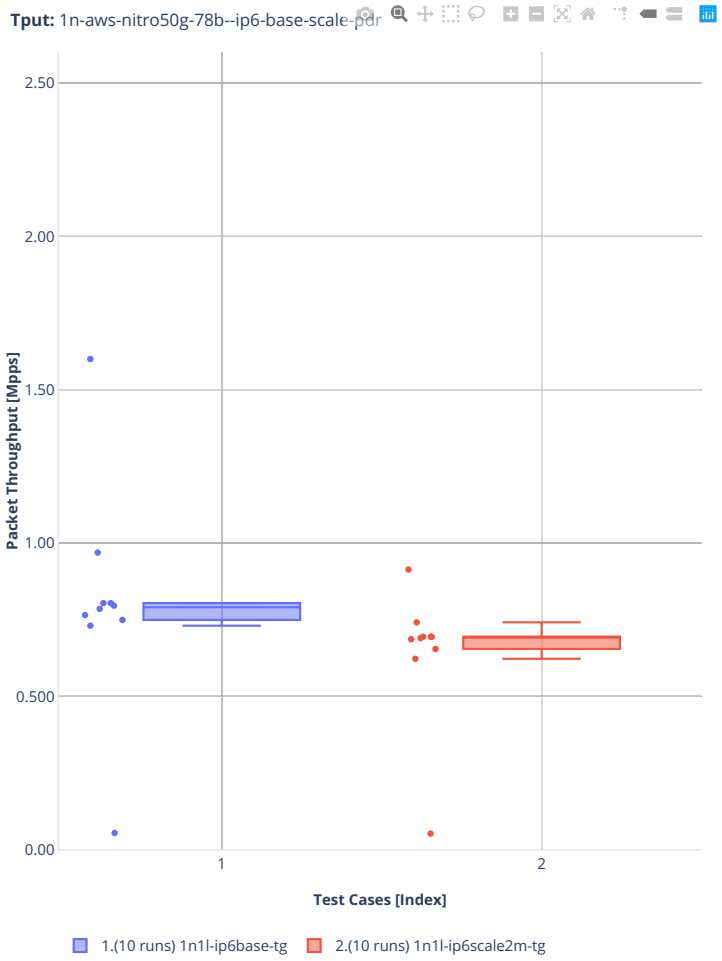
64b-ip4routing-base-scale




Tput: 1n-aws-nitro50g-64b--ip4-base-scale-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
Test Cases [Index]
1.(10 runs) 1n1l-ip4base-tg

78b-ip6routing-base-scale




Tput: 1n-aws-nitro50g-78b--ip6-base-scale-pdr
2.50
2.00
1.50
1.00
0.500
0.00
Packet Throughput [Mpps]
1
2
Test Cases [Index]
1.(10 runs) 1n1l-ip6base-tg
2.(10 runs) 1n1l-ip6scale2m-tg

4.4 Throughput Trending

CSIT provides continuous performance trending for master branch: C-Dash[364]

[364] http://csit.fd.io/trending/

4.5 Test Environment

4.5.1 Environment Versioning

CSIT test environment versioning has been introduced to track modifications of the test environment.

Any benchmark anomalies (progressions, regressions) between releases of a DUT application (e.g. VPP, DPDK), are determined by testing it in the same test environment, to avoid test environment changes clouding the picture. To beter distinguish impact of test environment changes, we also execute tests without any SUT (just with TRex TG sending packets over a link looping back to TG).

A mirror approach is introduced to determine benchmarking anomalies due to the test environment change. This is achieved by testing the same DUT application version between releases of CSIT test system. This works under the assumption that the behaviour of the DUT is deterministic under the test conditions.

CSIT test environment versioning scheme ensures integrity of all the test system components, including their HW revisions, compiled SW code versions and SW source code, within a specific CSIT version. Components included in the CSIT environment versioning include:

- **HW** Server hardware firmware and BIOS (motherboard, processsor, NIC(s), accelerator card(s)), tracked in CSIT branch.
- **Linux** Server Linux OS version and configuration, tracked in CSIT Reports.
- **TRex** TRex Traffic Generator version, drivers and configuration tracked in TG Settings.
- **CSIT** CSIT framework code tracked in CSIT release branches.

Following is the list of CSIT versions to date:

- Ver. 1 associated with CSIT rls1908 branch (HW[365], Linux[366], TRex[367], CSIT[368]).
- Ver. 2 associated with CSIT rls2001 branch (HW[369], Linux[370], TRex[371], CSIT[372]).
- Ver. 4 associated with CSIT rls2005 branch (HW[373], Linux[374], TRex[375], CSIT[376]).
- Ver. 5 associated with CSIT rls2009 branch (HW[377], Linux[378], TRex[379], CSIT[380]).
  - The main change is TRex data-plane core resource adjustments: increase from 7 to 8 cores and pinning cores to interfaces[381] for better TRex performance with symmetric traffic profiles.
- Ver. 6 associated with CSIT rls2101 branch (HW[382], Linux[383], TRex[384], CSIT[385]).
  - The main change is TRex version upgrade: increase from 2.82 to 2.86.

[365] https://git.fd.io/csit/tree/docs/lab?h=rls1908
[366] https://docs.fd.io/csit/rls1908/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[367] https://docs.fd.io/csit/rls1908/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[368] https://git.fd.io/csit/tree/?h=rls1908
[369] https://git.fd.io/csit/tree/docs/lab?h=rls2001
[370] https://docs.fd.io/csit/rls2001/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[371] https://docs.fd.io/csit/rls2001/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[372] https://git.fd.io/csit/tree/?h=rls2001
[373] https://git.fd.io/csit/tree/docs/lab?h=rls2005
[374] https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[375] https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[376] https://git.fd.io/csit/tree/?h=rls2005
[377] https://git.fd.io/csit/tree/docs/lab?h=rls2009
[378] https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[379] https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[380] https://git.fd.io/csit/tree/?h=rls2009
[381] https://gerrit.fd.io/r/c/csit/+/28184
[382] https://git.fd.io/csit/tree/docs/lab?h=rls2101
[383] https://docs.fd.io/csit/rls2101/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[384] https://docs.fd.io/csit/rls2101/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[385] https://git.fd.io/csit/tree/?h=rls2101

- Ver. 7 associated with CSIT rls2106 branch (HW[386], Linux[387], TRex[388], CSIT[389]).
  - TRex version upgrade: increase from 2.86 to 2.88.
  - Ubuntu upgrade from 18.04 LTS to 20.04.2 LTS.
- Ver. 8 associated with CSIT rls2110 branch (HW[390], Linux[391], TRex[392], CSIT[393]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
- Ver. 9 associated with CSIT rls2202 branch (HW[394], Linux[395], TRex[396], CSIT[397]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
- Ver. 10 associated with CSIT rls2206 branch (HW[398], Linux[399], TRex[400], CSIT[401]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
  - Mellanox 556A series firmware upgrade based on DPDK compatibility matrix.
  - Intel IceLake all core turbo frequency turned off. Current base frequency is 2.6GHz.
  - TRex version upgrade: increase from 2.88 to 2.97.
- Ver. 11 associated with CSIT rls2210 branch (HW[402], Linux[403], TRex[404], CSIT[405]).
  - Intel NIC 700/800 series firmware upgrade based on DPDK compatibility matrix.
  - Mellanox 556A series firmware upgrade based on DPDK compatibility matrix.
  - Ubuntu upgrade from 20.04.2 LTS to 22.04.1 LTS. (2n-dnv and 3n-dnv keeps the Ubuntu 20.04.2LTS as a part of decomission).
  - TRex version upgrade: increase from 2.97 to 3.00.

4.5.2 SUT Settings - TRex

4.5.3 TG Settings - TRex

TG Version

TRex v3.00

[386] https://git.fd.io/csit/tree/docs/lab?h=rls2106
[387] https://s3-docs.fd.io/csit/rls2106/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[388] https://s3-docs.fd.io/csit/rls2106/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[389] https://git.fd.io/csit/tree/?h=rls2106
[390] https://git.fd.io/csit/tree/docs/lab?h=rls2110
[391] https://s3-docs.fd.io/csit/rls2110/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[392] https://s3-docs.fd.io/csit/rls2110/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[393] https://git.fd.io/csit/tree/?h=rls2110
[394] https://git.fd.io/csit/tree/docs/lab?h=rls2202
[395] https://s3-docs.fd.io/csit/rls2202/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[396] https://s3-docs.fd.io/csit/rls2202/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[397] https://git.fd.io/csit/tree/?h=rls2202
[398] https://git.fd.io/csit/tree/docs/lab?h=rls2206
[399] https://s3-docs.fd.io/csit/rls2206/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[400] https://s3-docs.fd.io/csit/rls2206/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[401] https://git.fd.io/csit/tree/?h=rls2206
[402] https://git.fd.io/csit/tree/docs/lab?h=rls2210
[403] https://s3-docs.fd.io/csit/rls2210/report/vpp_performance_tests/test_environment.html#sut-settings-linux
[404] https://s3-docs.fd.io/csit/rls2210/report/vpp_performance_tests/test_environment.html#tg-settings-trex
[405] https://git.fd.io/csit/tree/?h=rls2210

DPDK Version

DPDK v21.02

TG Installation

T-Rex installation is managed via Ansible role.

TG Startup Configuration

```
$ sudo -E -S sh -c 'cat << EOF > /etc/trex_cfg.yaml
- version: 2
  c: 8
  limit_memory: 8192
  interfaces: ["${pci1}","${pci2}"]
  port_info:
      - dest_mac: [${dest_mac1}]
        src_mac: [${src_mac1}]
      - dest_mac: [${dest_mac2}]
        src_mac: [${src_mac2}]
  platform :
      master_thread_id: 0
      latency_thread_id: 9
      dual_if:
          - socket: 0
            threads: [1, 2, 3, 4, 5, 6, 7, 8]
EOF'
```

TG Startup Command (Stateless Mode)

```
$ sudo -E -S sh -c "cd '${trex_install_dir}/scripts/' && \
  nohup ./t-rex-64 -i --prefix $(hostname) --hdrh --no-scapy-server \
  --mbuf-factor 32 > /tmp/trex.log 2>&1 &" > /dev/null
```

Also, Python client is now starting traffic with:

```
core_mask=STLClient.CORE_MASK_PIN
```

TG Startup Command (Stateful Mode)

```
$ sudo -E -S sh -c "cd '${trex_install_dir}/scripts/' && \
  nohup ./t-rex-64 -i --prefix $(hostname) --astf --hdrh --no-scapy-server \
  --mbuf-factor 32 > /tmp/trex.log 2>&1 &" > /dev/null
```

TG API Driver

TRex driver[406]

[406] https://git.fd.io/csit/tree/GPL/tools/trex/trex_stl_profile.py?h=rls2210

CHAPTER

FIVE

VPP DEVICE

5.1 Overview

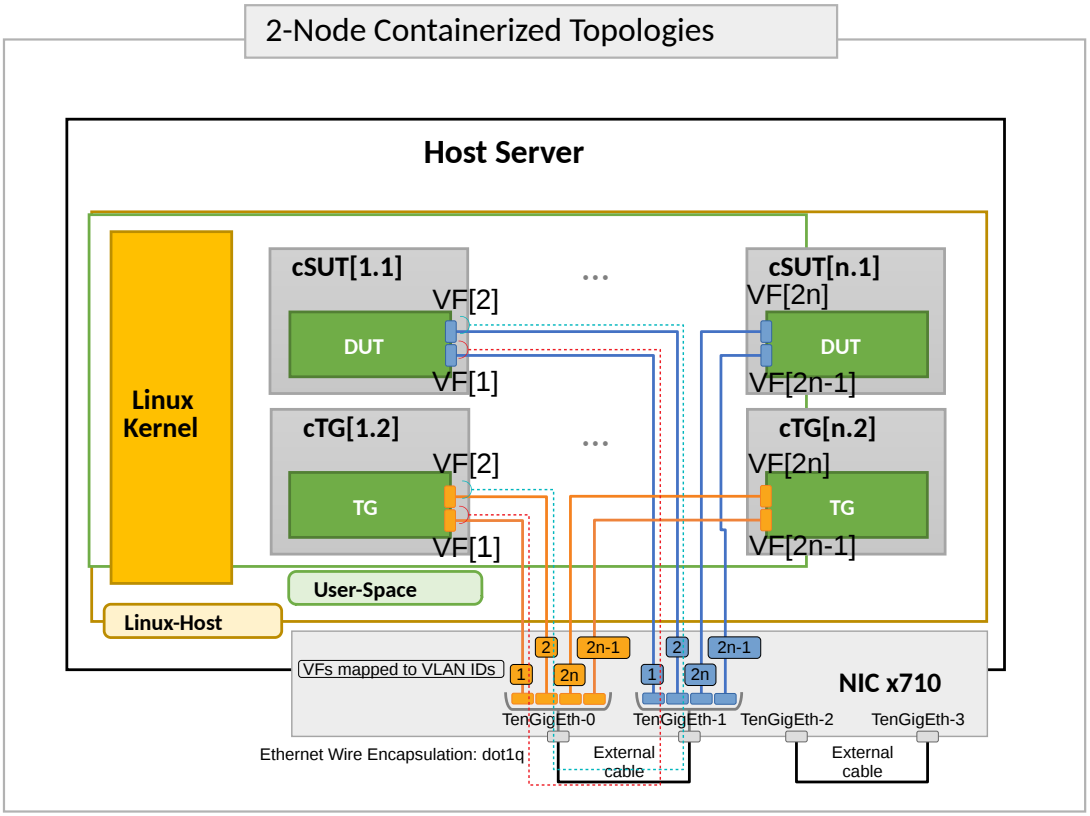
5.1.1 Virtual Topologies

CSIT VPP Device tests are executed in Physical containerized topologies created on demand using set of scripts hosted and developed under CSIT repository. It runs on physical baremetal servers hosted by LF FD.io project. Based on the packet path thru SUT Containers, three distinct logical topology types are used for VPP DUT data plane testing:

1. vfNIC-to-vfNIC switching topologies.
2. vfNIC-to-vhost-user switching topologies.
3. vfNIC-to-memif switching topologies.

vfNIC-to-vfNIC Switching

The simplest physical topology for software data plane application like VPP is vfNIC-to-vfNIC switching. Tested virtual topologies for 2-Node testbeds are shown in figures below.



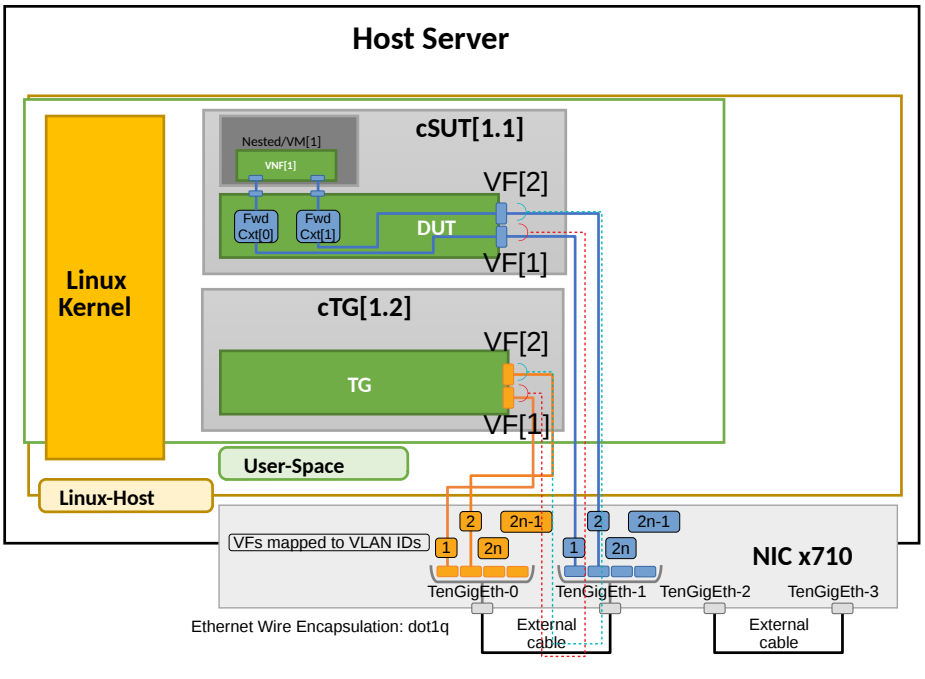
SUT1 is Docker Container (running Ubuntu, depending on the test suite), TG is a Traffic Generator (running Ubuntu Container). SUTs run VPP SW application in Linux user-mode as a Device Under Test (DUT) within the container. TG runs Scapy SW application as a packet Traffic Generator. Network connectivity between SUTs and to TG is provided using virtual function of physical NICs.

Virtual topologies are created on-demand whenever a verification job is started (e.g. triggered by the gerrit patch submission) and destroyed upon completion of all functional tests. Each node is a container running on physical server. During the test execution, all nodes are reachable thru the Management (not shown above for clarity).

vfNIC-to-vhost-user Switching

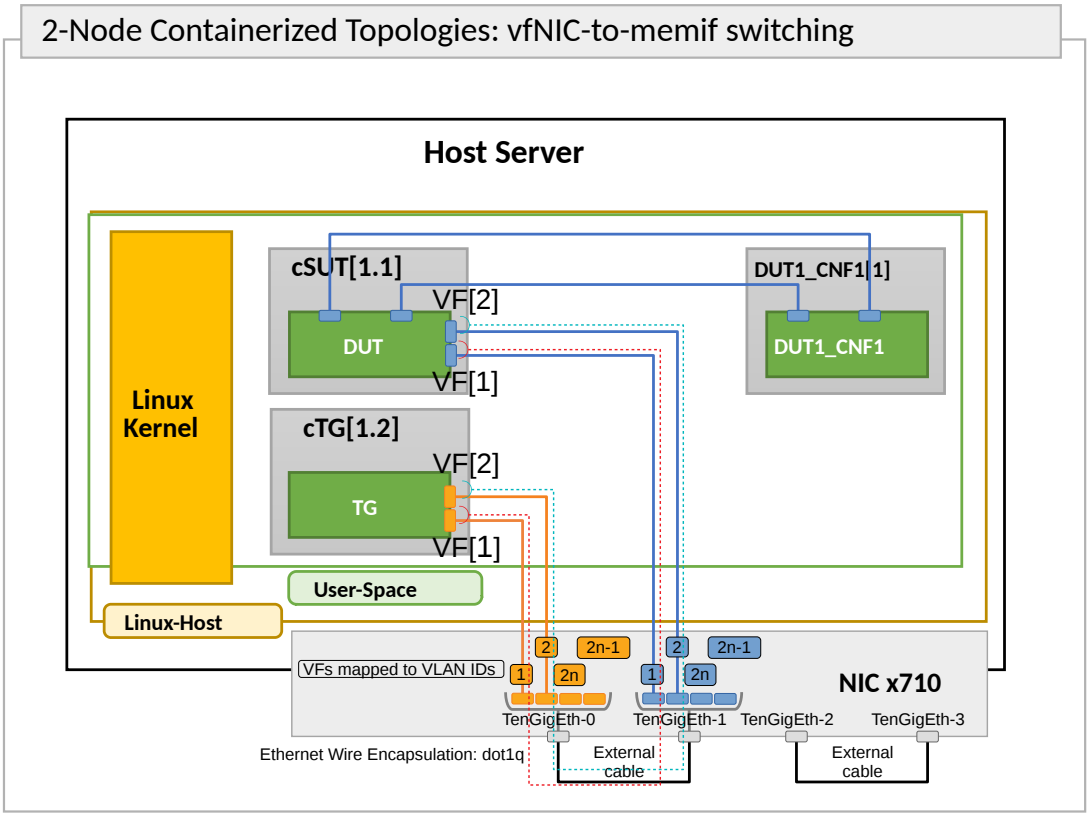
vfNIC-to-vhost-user switching topology test cases require VPP DUT to communicate with Virtual Machine (VM) over Vhost-user virtual interfaces. VM is created on SUT1 for the duration of these particular test cases only. Virtual test topology with VM is shown in the figure below.

2-Node Containerized Topologies: vfNIC-to-vhost-user switching



vfNIC-to-memif Switching

vfNIC-to-memif switching topology test cases require VPP DUT to communicate with another Docker Container over memif interfaces. Container is created for the duration of these particular test cases only and it is running the same VPP version as running on DUT. Virtual test topology with Memif is shown in the figure below.



5.1.2 Functional Tests Coverage

CSIT-2210 includes following VPP functionality tested in VPP Device environment:

Functionality	Description
ACL (classify)	Ingress Access Control List security for L2 Bridge-Domain MAC switching, IPv4 routing, IPv6 routing.
ACL (acl_plugin)	Ingress and Egress Access Control List security in stateless and stateful mode for L2 Bridge-Domain MAC switching, IPv4 routing, IPv6 routing.
ADL	ADL address allow-list and block-list filtering for IPv4 and IPv6 routing.
GENEVE	GENEVE tunnels for IPv4 routing.
IPSec	IPSec tunnel and transport modes.
IPv4	IPv4 routing, ICMPv4.
IPv6	IPv4 routing, ICMPv6.
L2BD	L2 Bridge-Domain switching for untagged Ethernet.
L2XC	L2 Cross-Connect switching for untagged Ethernet.
MACIP (acl_plugin)	Ingress Access Control List security for L2 Bridge-Domain MAC switching based on mix of MAC and IP address matches.
Memif Interface	Baseline VPP memif interface tests.
NAT44	Network Address and Port Translation deterministic mode and endpoint-dependent mode tests for IPv4.
QoS Policer Metering	Ingress packet rate metering and marking for IPv4, IPv6.
SRv6	Segment routing over IPv6, base and proxy.
Tap Interface	Baseline Linux tap interface tests.
VLAN Tag	L2 VLAN subinterfaces.
Vhost-user Interface	Baseline VPP vhost-user interface tests.
VXLAN	VXLAN overlay tunneling for L2-over-IPv4 and -over-IPv6.

5.1.3 Tests Naming

CSIT-2210 follows a common structured naming convention for all performance and system functional tests.

The naming should be intuitive for majority of the tests. Complete description of CSIT test naming convention is provided on *Test Naming* (page 1430).

5.2 Release Notes

5.2.1 Changes in CSIT-2210

1. TEST FRAMEWORK
    - **CSIT test environment** version has been updated to ver. 11, see *Environment Versioning* (page 1206).

5.2.2 Known Issues

List of known issues in CSIT-2210 for VPP functional tests in VPP Device:

#	JiraID	Issue Description

5.3 Integration Tests

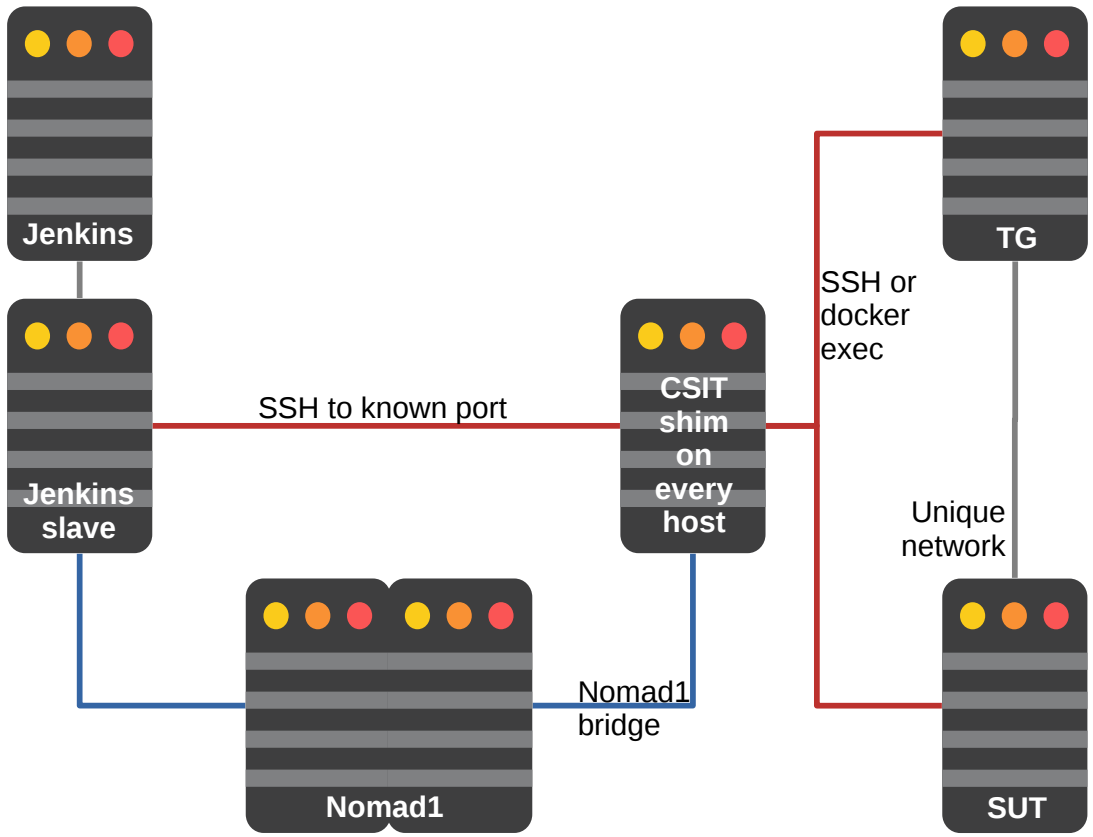
5.3.1 Abstract

FD.io VPP software data plane technology has become very popular across a wide range of VPP ecosystem use cases, putting higher pressure on continuous verification of VPP software quality.

This document describes a proposal for design and implementation of extended continuous VPP testing by extending existing test environments. Furthermore it describes and summarizes implementation details of Integration and System tests platform *1-Node VPP_Device*. It aims to provide a complete end-to-end view of *1-Node VPP_Device* environment in order to improve extendability and maintenance, under the guideline of VPP core team.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **RFC 8174**[407].

5.3.2 Overview



[407] https://datatracker.ietf.org/doc/html/rfc8174.html

5.3.3 Physical Testbeds

All FD.io CSIT vpp-device tests are executed on physical testbeds built with bare-metal servers hosted by LF FD.io project. Two 1-node testbed topologies are used:

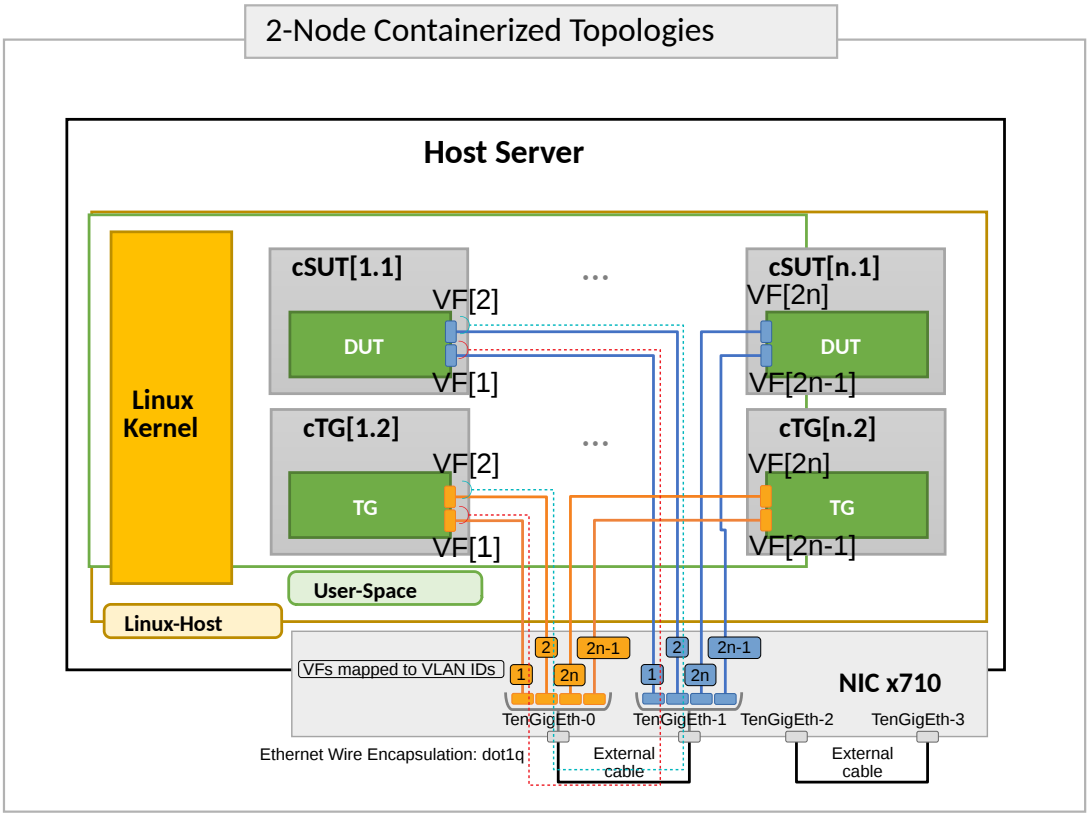
- **2-Container Topology**: Consisting of one Docker container acting as SUT (System Under Test) and one Docker container as TG (Traffic Generator), both connected in ring topology via physical NIC cross-connecting.

Current FD.io production testbeds are built with servers based on one processor generation of Intel Xeons: Skylake (Platinum 8180). Testbeds built with servers based on Arm processors are in the process of being added to FD.io production.

Following section describe existing production 1n-skx testbed.

1-Node Xeon Skylake (1n-skx)

1n-skx testbed is based on single SuperMicro SYS-7049GP-TRT server equipped with two Intel Xeon Skylake Platinum 8180 2.5 GHz 28 core processors. Physical testbed topology is depicted in a figure below.



Server is populated with the following NIC models:

1. NIC-1: x710-da4 4p10GE Intel.
2. NIC-2: E810-2CQDA2 2p100GbE Intel.

All Intel Xeon Skylake servers run with Intel Hyper-Threading enabled, doubling the number of logical cores exposed to Linux, with 56 logical cores and 28 physical cores per processor socket.

NIC interfaces are shared using Linux vfio_pci and VPP VF drivers:

- DPDK VF driver,

- Fortville AVF driver.

Provided Intel x710-da4 4p10GE NICs support 32 VFs per interface, 128 per NIC.

Total of two 1n-skx testbeds are in operation in FD.io labs.

1-Node Virtualbox (1n-vbox)

1n-skx testbed can run in single VirtualBox VM machine. This solution replaces the previously used Vagrant environment based on 3 VMs.

VirtualBox VM MAY be created by Vagrant and MUST have additional 4 virtio NICs each pair attached to separate private networks to simulate back-to-back connections. It SHOULD be 82545EM device model (otherwise can be changed in boostrap scripts). Example of Vagrant configuration:

```
Vagrant.configure(2) do |c|
  c.vm.network "private_network", type: "dhcp", auto_config: false,
      virtualbox__intnet: "port1", nic_type: "82545EM"
  c.vm.network "private_network", type: "dhcp", auto_config: false,
      virtualbox__intnet: "port2", nic_type: "82545EM"

  c.vm.provider :virtualbox do |v|
    v.customize ["modifyvm", :id, "--nicpromisc2", "allow-all"]
    v.customize ["modifyvm", :id, "--nicpromisc3", "allow-all"]
    v.customize ["modifyvm", :id, "--nicpromisc4", "allow-all"]
    v.customize ["modifyvm", :id, "--nicpromisc5", "allow-all"]
```

Vagrant VM is populated with the following NIC models:

1. NIC-1: 82545EM Intel.
2. NIC-2: 82545EM Intel.
3. NIC-3: 82545EM Intel.
4. NIC-4: 82545EM Intel.

5.3.4 Containers

It was agreed on TWS (Technical Work Stream) call to continue with Ubuntu 18.04 LTS as a baseline system with OPTIONAL extend to Centos 7 and SuSE per demand[408].

All DCR (Docker container) images are REQUIRED to be hosted on Docker registry available from LF network, publicly available and trackable. For backup, tracking and contributing purposes all Dockerfiles (including files needed for building container) MUST be available and stored in[412]repository under appropriate folders. This allows the peer review process to be done for every change of infrastructure related to scope of this document. Currently only **csit-shim-dcr** and **csit-sut-dcr** containers will be stored and maintained under CSIT repository by CSIT contributors.

At the time of designing solution described in this document the interconnection between[410]and[412] for automated build purposes and image hosting cannot be established with the trust and respectful to security of FD.io project. Unless adressed, DCR images will be placed in custom registry service[414]. Automated Jenkins jobs will be created in align of long term solution for container lifecycle and ability to build new version of docker images.

[408] TWS[409]
[409] https://wiki.fd.io/view/CSIT/TWS
[412] FD.io/CSIT gerrit[413]
[413] https://gerrit.fd.io/r/CSIT
[410] Docker hub[411]
[411] https://hub.docker.com/
[414] FD.io regisy

In parallel, the effort is started to find the outsourced Docker registry service.

Versioning

As of initial version of vpp-device, we do have only single latest version of Docker image hosted on[410]. This will be addressed as further improvement with proper semantic versioning.

jenkins-slave-dcr

This DCR acts as the Jenkins slave (known also as jenkins minion). It can connect over SSH protocol to TCP port 6022 of **csit-shim-dcr** and executes non-interactive reservation script. Nomad is responsible for scheduling this container execution onto specific **1-Node VPP_Device** testbed. It executes CSIT environment including CSIT framework.

All software dependencies including VPP/DPDK that are not present in **csit-sut-dcr** container image and/or needs to be compiled prior running on **csit-sut-dcr** SHOULD be compiled in this container.

- *Container Image Location*: Docker image at snergster/vpp-ubuntu18.
- *Container Definition*: Docker file specified at[415].
- *Initializing*: Container is initialized from within *Consul by HashiCorp* and *Nomad by HashiCorp*.

csit-shim-dcr

This DCR acts as an intermediate layer running script responsible for orchestrating topologies under test and reservation. Responsible for managing VF resources and allocation to DUT (Device Under Test), TG (Traffic Generator) containers. This MUST to be done on **csit-shim-dcr**. This image also acts as the generic reservation mechanics arbiter to make sure that only Y number of simulations are spawned on any given HW node.

- *Container Image Location*: Docker image at snergster/csit-shim.
- *Container Definition*: Docker file specified at[417].
- *Initializing*: Container is initialized from within *Consul by HashiCorp* and *Nomad by HashiCorp*. Required docker parameters, to be able to run nested containers with VF reservation system are: privileged, net=host, pid=host.
- *Connectivity*: Over SSH only, using <host>:6022 format. Currently using *root* user account as primary. From the jenkins slave it will be able to connect via env variable, since the jenkins slave doesn't actually know what host its running on.

```
ssh -p 6022 root@10.30.51.node
```

[415] jenkins-slave-dcr-file[416]
[416] https://github.com/snergfdio/multivppcache/blob/master/ubuntu18/Dockerfile
[417] csit-shim-dcr-file[418]
[418] https://github.com/snergfdio/multivppcache/blob/master/csit-shim/Dockerfile

csit-sut-dcr

This DCR acts as an SUT (System Under Test). Any DUT or TG application is installed there. It is RECOMMENDED to install DUT and all DUT dependencies via commands `rpm -ihv` on RedHat based OS or `dpkg -i` on Debian based OS.

Container is designed to be a very lightweight Docker image that only installs packages and execute binaries (previously built or downloaded on **jenkins-slave-dcr**) and contains libraries necessary to run CSIT framework including those required by DUT/TG.

- *Container Image Location*: Docker image at snergster/csit-sut.
- *Container Definition*: Docker file specified at[419].
- *Initializing*:

```
docker run
# Run the container in the background and print the new container ID.
--detach=true
# Give extended privileges to this container. A "privileged" container is
# given access to all devices and able to run nested containers.
--privileged
# Publish all exposed ports to random ports on the host interfaces.
--publish-all
# Automatically remove the container when it exits.
--rm
# Size of /dev/shm.
dcr_stc_params+="--shm-size 512M "
# Override access to PCI bus by attaching a filesystem mount to the
# container.
dcr_stc_params+="--mount type=tmpfs,destination=/sys/bus/pci/devices "
# Mount vfio to be able to bind to see bound interfaces. We cannot use
# --device=/dev/vfio as this does not see newly bound interfaces.
dcr_stc_params+="--volume /dev/vfio:/dev/vfio "
# Mount docker.sock to be able to use docker deamon of the host.
dcr_stc_params+="--volume /var/run/docker.sock:/var/run/docker.sock "
# Mount /opt/boot/ where VM kernel and initrd are located.
dcr_stc_params+="--volume /opt/boot/:/opt/boot/ "
# Mount host hugepages for VMs.
dcr_stc_params+="--volume /dev/hugepages/:/dev/hugepages/ "
```

  Container name is catenated from **csit-** prefix and uuid generated uniquely for each container instance.

- *Connectivity*: Over SSH only, using <host>[:<port>] format. Currently using *root* user account as primary.

```
ssh -p <port> root@10.30.51.<node>
```

Container required to run as `--privileged` due to ability to create nested containers and have full read/write access to sysfs (for bind/unbind). Docker automatically pick free network port (`--publish-all`) for ability to connect over ssh. To be able to limit access to PCI bus, container is creating tmpfs mount type in PCI bus tree. CSIT reservation script is dynamically linking only PCI devices (NIC cards) that are reserved for particular container. This way it is not colliding with other containers. To make vfio work, access to `/dev/vfio` must be granted.

[419] csit-sut-dcr-filePage 1421, 420
[420] https://github.com/snergfdio/multivppcache/blob/master/csit-sut/Dockerfile

5.3.5 Environment initialization

All 1-node servers are to be managed and provisioned via the[421]set of playbooks with *vpp-device* role. Full playbooks can be found under[423]directory. This way we are able to track all configuration changes of physical servers in gerrit (in structured yaml format) as well as we are able to extend *vpp-device* to additional servers with less effort or re-stage servers in case of failure.

SR-IOV VF initialization is done via `systemd` service during host system boot up. Service with name *csit-initialize-vfs.service* is created under systemd system context (`/etc/systemd/system/`). By default service is calling `/usr/local/bin/csit-initialize-vfs.sh` with single parameter:

- **start**: Creates maximum number of virtual functions (VFs) (detected from `sriov_totalvfs`) for each whitelisted PCI device.
- **stop**: Removes all VFs (Virtual Functions) for all whitelisted PCI device.

Service is considered active even when all of its processes exited successfully. Stopping service will automatically remove VFs.

```
[Unit]
Description=CSIT Initialize SR-IOV VFs
After=network.target

[Service]
Type=one-shot
RemainAfterExit=True
ExecStart=/usr/local/bin/csit-initialize-vfs.sh start
ExecStop=/usr/local/bin/csit-initialize-vfs.sh stop

[Install]
WantedBy=default.target
```

Script is driven by two array variables `pci_blacklist`/`pci_whitelist`. They MUST store all PCI addresses in **<domain>:<bus>:<device>.<func>** format, where:

- **pci_blacklist**: PCI addresses to be skipped from VFs initialization (useful for e.g. excluding management network interfaces).
- **pci_whitelist**: PCI addresses to be included for VFs initialization.

5.3.6 VF reservation

During topology initialization phase of script, mutex is used to avoid multiple instances of script to interact with each other during resources allocation. Mutal exclusion ensure that no two distinct instances of script will get same resource list.

Reservation function reads the list of all available virtual function network devices in system:

```
# Find the first ${device_count} number of available TG Linux network
# VF device names. Only allowed VF PCI IDs are filtered.
for netdev in ${tg_netdev[@]}
do
    for netdev_path in $(grep -l "${pci_id}" \
                         /sys/class/net/${netdev}*/device/device \
                         2> /dev/null)
    do
```

(continues on next page)

[421] ansible[Page 1422, 422]
[422] https://www.ansible.com/
[423] Fd.io/CSIT ansible[424]
[424] https://git.fd.io/csit/tree/fdio.infra.ansible

(continued from previous page)

```
        if [[ ${#TG_NETDEVS[@]} -lt ${device_count} ]]; then
            tg_netdev_name=$(dirname ${netdev_path})
            tg_netdev_name=$(dirname ${tg_netdev_name})
            TG_NETDEVS+=($(basename ${tg_netdev_name}))
        else
            break
        fi
    done
    if [[ ${#TG_NETDEVS[@]} -eq ${device_count} ]]; then
        break
    fi
done
```

Where `${pci_id}` is ID of white-listed VF PCI ID. For more information please see[427]. This act as security constraint to prevent taking other unwanted interfaces. The output list of all VF network devices is split into two lists for TG and SUT side of connection. First two items from each TG or SUT network devices list are taken to expose directly to namespace of container. This can be done via commands:

```
$ ip link set ${netdev} netns ${DCR_CPIDS[tg]}
$ ip link set ${netdev} netns ${DCR_CPIDS[dut1]}
```

In this stage also symbolic links to PCI devices under sysfs bus directory tree are created in running containers. Once VF devices are assigned to container namespace and PCI devices are linked to running containers and mutex is exited. Selected VF network device automatically disappear from parent container namespace, so another instance of script will not find device under that namespace.

Once Docker container exits, network device is returned back into parent namespace and can be reused.

5.3.7 Network traffic isolation - Intel i40evf

In a virtualized environment, on Intel(R) Server Adapters that support SR-IOV, the virtual function (VF) may be subject to malicious behavior. Software- generated layer two frames, like IEEE 802.3x (link flow control), IEEE 802.1Qbb (priority based flow-control), and others of this type, are not expected and can throttle traffic between the host and the virtual switch, reducing performance. To resolve this issue, configure all SR-IOV enabled ports for VLAN tagging. This configuration allows unexpected, and potentially malicious, frames to be dropped.[425]

To configure VLAN tagging for the ports on an SR-IOV enabled adapter, use the following command. The VLAN configuration SHOULD be done before the VF driver is loaded or the VM is booted.[425]

```
$ ip link set dev <PF netdev id> vf <id> vlan <vlan id>
```

For example, the following instructions will configure PF eth0 and the first VF on VLAN 10.

```
$ ip link set dev eth0 vf 0 vlan 10
```

VLAN Tag Packet Steering allows to send all packets with a specific VLAN tag to a particular SR-IOV virtual function (VF). Further, this feature allows to designate a particular VF as trusted, and allows that trusted VF to request selective promiscuous mode on the Physical Function (PF).[425]

To set a VF as trusted or untrusted, enter the following command in the Hypervisor:

```
$ ip link set dev eth0 vf 1 trust [on|off]
```

[427] pci idsPage 1423, 428
[428] http://pci-ids.ucw.cz/v2.2/pci.ids
[425] Intel i40e[426]
[426] https://downloadmirror.intel.com/26370/eng/readme.txt

Once the VF is designated as trusted, use the following commands in the VM to set the VF to promiscuous mode.[425]

- For promiscuous all:

```
$ ip link set eth2 promisc on
```

- For promiscuous Multicast:

```
$ ip link set eth2 allmulti on
```

Note: By default, the ethtool priv-flag vf-true-promisc-support is set to *off*, meaning that promiscuous mode for the VF will be limited. To set the promiscuous mode for the VF to true promiscuous and allow the VF to see all ingress traffic, use the following command. $ ethtool set-priv-flags p261p1 vf-true-promisc-support on The vf-true-promisc-support priv-flag does not enable promiscuous mode; rather, it designates which type of promiscuous mode (limited or true) you will get when you enable promiscuous mode using the ip link commands above. Note that this is a global setting that affects the entire device. However,the vf-true-promisc-support priv-flag is only exposed to the first PF of the device. The PF remains in limited promiscuous mode (unless it is in MFP mode) regardless of the vf-true-promisc-support setting.[Page 1423, 425]

Service described earlier *csit-initialize-vfs.service* is responsible for assigning 802.1Q vlan tagging to each virtual function via physical function from list of white-listed PCI addresses by following (simplified) code.

```
SCRIPT_DIR="$(dirname $(readlink -e "${BASH_SOURCE[0]}"))"
source "${SCRIPT_DIR}/csit-initialize-vfs-data.sh"

# Initilize whitelisted NICs with maximum number of VFs.
pci_idx=0
for pci_addr in ${PCI_WHITELIST[@]}; do
    if ! [[ ${PCI_BLACKLIST[*]} =~ "${pci_addr}" ]]; then
        pci_path="/sys/bus/pci/devices/${pci_addr}"
        # SR-IOV initialization
        case "${1:-start}" in
            "start" )
                sriov_totalvfs=$(< "${pci_path}"/sriov_totalvfs)
                ;;
            "stop" )
                sriov_totalvfs=0
                ;;
        esac
        echo ${sriov_totalvfs} > "${pci_path}"/sriov_numvfs
        # SR-IOV 802.1Q isolation
        case "${1:-start}" in
            "start" )
                pf=$(basename "${pci_path}"/net/*)
                for vf in $(seq "${sriov_totalvfs}"); do
                    # PCI address index in array (pairing siblings).
                    if [[ -n ${PF_INDICES[@]} ]]
                    then
                        vlan_pf_idx=${PF_INDICES[$pci_addr]}
                    else
                        vlan_pf_idx=$((pci_idx % (${#PCI_WHITELIST[@]}/2)))
                    fi
                    # 802.1Q base offset.
                    vlan_bs_off=1100
                    # 802.1Q PF PCI address offset.
```

(continues on next page)

(continued from previous page)

```
                    vlan_pf_off=$(( vlan_pf_idx * 100 + vlan_bs_off ))
                    # 802.1Q VF PCI address offset.
                    vlan_vf_off=$(( vlan_pf_off + vf - 1 ))
                    # VLAN string.
                    vlan_str="vlan ${vlan_vf_off}"
                    # MAC string.
                    mac5="$(printf '%x' ${pci_idx})"
                    mac6="$(printf '%x' $(( vf - 1 )))"
                    mac_str="mac ba:dc:0f:fe:${mac5}:${mac6}"
                    # Set 802.1Q VLAN id and MAC address
                    ip link set ${pf} vf $(( vf - 1)) ${mac_str} ${vlan_str}
                    ip link set ${pf} vf $(( vf - 1)) trust on
                    ip link set ${pf} vf $(( vf - 1)) spoof off
                done
                pci_idx=$(( pci_idx + 1 ))
                ;;
        esac
        rmmod i40evf
        modprobe i40evf
    fi
done
```

Assignment starts at VLAN 1100 and incrementing by 1 for each VF and by 100 for each white-listed PCI address up to the middle of the PCI list. Second half of the lists is assumed to be directly (cable) paired siblings and assigned with same 802.1Q VLANs as its siblings.

5.3.8 Open tasks

Security

Note: Switch to non-privileged containers: As of now all three container flavors are using privileged containers to make it working. Explore options to switch containers to non-privileged with explicit rather implicit privileges.

Note: Switch to testuser account instead of root.

Maintainability

Note: Docker image distribution: Create jenkins jobs with full pipeline of CI/CD for CSIT Docker images.

Stability

Note: Implement queueing mechanism: Currently there is no mechanics that would place starving jobs in queue in case of no resources available.

Note: Replace reservation script with Docker network plugin written in GOLANG/SH/Python - platform independent.

5.3.9 Links

CHAPTER

SIX

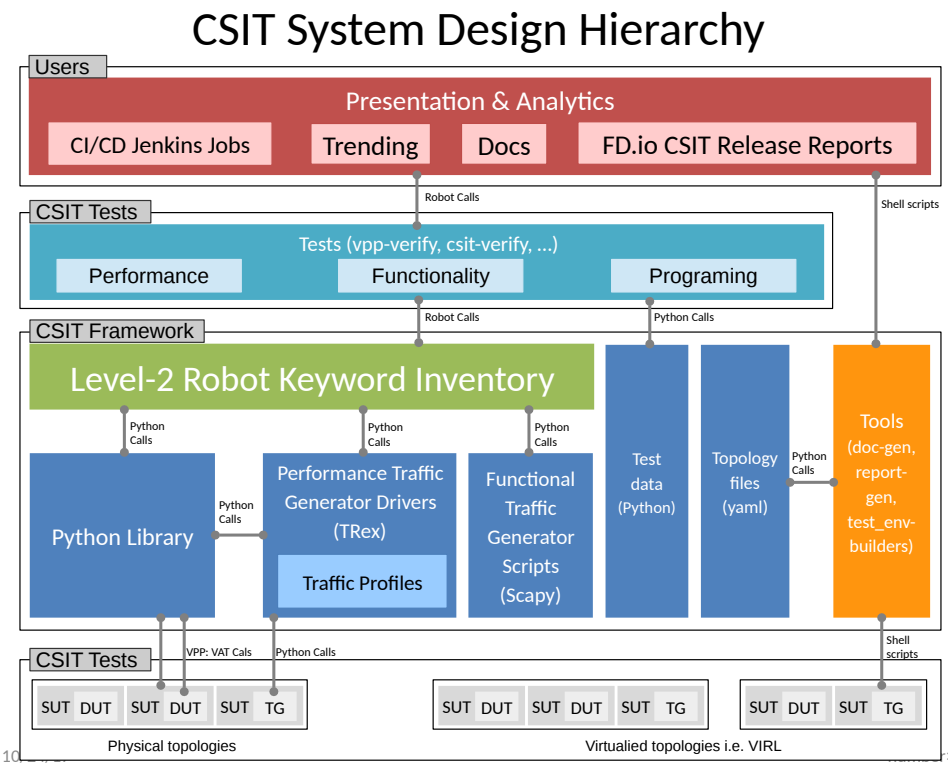
CSIT FRAMEWORK

6.1 Design

FD.io CSIT system design needs to meet continuously expanding requirements of FD.io projects including VPP, related sub-systems (e.g. plugin applications, DPDK drivers) and FD.io applications (e.g. DPDK applications), as well as growing number of compute platforms running those applications. With CSIT project scope and charter including both FD.io continuous testing AND performance trending/comparisons, those evolving requirements further amplify the need for CSIT framework modularity, flexibility and usability.

6.1.1 Design Hierarchy

CSIT follows a hierarchical system design with SUTs and DUTs at the bottom level of the hierarchy, presentation level at the top level and a number of functional layers in-between. The current CSIT system design including CSIT framework is depicted in the figure below.



A brief bottom-up description is provided here:

1. SUTs, DUTs, TGs
    - SUTs - Systems Under Test;
    - DUTs - Devices Under Test;
    - TGs - Traffic Generators;
2. Level-1 libraries - Robot and Python
    - Lowest level CSIT libraries abstracting underlying test environment, SUT, DUT and TG specifics;
    - Used commonly across multiple L2 KWs;
    - Performance and functional tests:
        - L1 KWs (KeyWords) are implemented as RF libraries and Python libraries;
    - Performance TG L1 KWs:
        - All L1 KWs are implemented as Python libraries:
            * Support for TRex only today;
            * CSIT IXIA drivers in progress;
    - Performance data plane traffic profiles:
        - TG-specific stream profiles provide full control of:
            * Packet definition - layers, MACs, IPs, ports, combinations thereof e.g. IPs and UDP ports;
            * Stream definitions - different streams can run together, delayed, one after each other;
            * Stream profiles are independent of CSIT framework and can be used in any T-rex setup, can be sent anywhere to repeat tests with exactly the same setup;
            * Easily extensible - one can create a new stream profile that meets tests requirements;
            * Same stream profile can be used for different tests with the same traffic needs;
    - Functional data plane traffic scripts:
        - Scapy specific traffic scripts;
3. Level-2 libraries - Robot resource files:
    - Higher level CSIT libraries abstracting required functions for executing tests;
    - L2 KWs are classified into the following functional categories:
        - Configuration, test, verification, state report;
        - Suite setup, suite teardown;
        - Test setup, test teardown;
4. Tests - Robot:
    - Test suites with test cases;
    - Performance tests using physical testbed environment:
        - VPP;
        - DPDK-Testpmd;
        - DPDK-L3Fwd;
    - Tools:
        - Documentation generator;
        - Report generator;

- Testbed environment setup ansible playbooks;
- Operational debugging scripts;

6.1.2 Test Lifecycle Abstraction

A well coded test must follow a disciplined abstraction of the test lifecycles that includes setup, configuration, test and verification. In addition to improve test execution efficiency, the commmon aspects of test setup and configuration shared across multiple test cases should be done only once. Translating these high-level guidelines into the Robot Framework one arrives to definition of a well coded RF tests for FD.io CSIT. Anatomy of Good Tests for CSIT:

1. Suite Setup - Suite startup Configuration common to all Test Cases in suite: uses Configuration KWs, Verification KWs, StateReport KWs;
2. Test Setup - Test startup Configuration common to multiple Test Cases: uses Configuration KWs, StateReport KWs;
3. Test Case - uses L2 KWs with RF Gherkin style:
   - prefixed with {Given} - Verification of Test setup, reading state: uses Configuration KWs, Verification KWs, StateReport KWs;
   - prefixed with {When} - Test execution: Configuration KWs, Test KWs;
   - prefixed with {Then} - Verification of Test execution, reading state: uses Verification KWs, StateReport KWs;
4. Test Teardown - post Test teardown with Configuration cleanup and Verification common to multiple Test Cases - uses: Configuration KWs, Verification KWs, StateReport KWs;
5. Suite Teardown - Suite post-test Configuration cleanup: uses Configuration KWs, Verification KWs, StateReport KWs;

6.1.3 RF Keywords Functional Classification

CSIT RF KWs are classified into the functional categories matching the test lifecycle events described earlier. All CSIT RF L2 and L1 KWs have been grouped into the following functional categories:

1. Configuration;
2. Test;
3. Verification;
4. StateReport;
5. SuiteSetup;
6. TestSetup;
7. SuiteTeardown;
8. TestTeardown;

6.1.4 RF Keywords Naming Guidelines

Readability counts: "..code is read much more often than it is written." Hence following a good and consistent grammar practice is important when writing RF KeyWords and Tests. All CSIT test cases are coded using Gherkin style and include only L2 KWs references. L2 KWs are coded using simple style and include L2 KWs, L1 KWs, and L1 python references. To improve readability, the proposal is to use the same grammar for both RF KW styles, and to formalize the grammar of English sentences used for naming the RF KWs. RF KWs names are short sentences expressing functional description of the command. They must follow English sentence grammar in one of the following forms:

1. **Imperative** - verb-object(s): *"Do something"*, verb in base form.
2. **Declarative** - subject-verb-object(s): *"Subject does something"*, verb in a third-person singular present tense form.
3. **Affirmative** - modal_verb-verb-object(s): *"Subject should be something"*, *"Object should exist"*, verb in base form.
4. **Negative** - modal_verb-Not-verb-object(s): *"Subject should not be something"*, *"Object should not exist"*, verb in base form.

Passive form MUST NOT be used. However a usage of past participle as an adjective is okay. See usage examples provided in the Coding guidelines section below. Following sections list applicability of the above grammar forms to different RF KW categories. Usage examples are provided, both good and bad.

6.1.5 Coding Guidelines

Coding guidelines can be found on Design optimizations wiki page[429].

6.2 Test Naming

6.2.1 Background

CSIT-2210 follows a common structured naming convention for all performance and system functional tests, introduced in CSIT-1701.

The naming should be intuitive for majority of the tests. Complete description of CSIT test naming convention is provided on CSIT test naming wiki page[430]. Below few illustrative examples of the naming usage for test suites across CSIT performance, functional and Honeycomb management test areas.

6.2.2 Naming Convention

The CSIT approach is to use tree naming convention and to encode following testing information into test suite and test case names:

1. packet network port configuration
   - port type, physical or virtual;
   - number of ports;
   - NIC model, if applicable;
   - port-NIC locality, if applicable;
2. packet encapsulations;
3. VPP packet processing

[429] https://wiki.fd.io/view/CSIT/Design_Optimizations
[430] https://wiki.fd.io/view/CSIT/csit-test-naming

- packet forwarding mode;
- packet processing function(s);

4. packet forwarding path
   - if present, network functions (processes, containers, VMs) and their topology within the computer;
5. main measured variable, type of test.

Proposed convention is to encode ports and NICs on the left (underlay), followed by outer-most frame header, then other stacked headers up to the header processed by vSwitch-VPP, then VPP forwarding function, then encap on vhost interface, number of vhost interfaces, number of VMs. If chained VMs present, they get added on the right. Test topology is expected to be symmetric, in other words packets enter and leave SUT through ports specified on the left of the test name. Here some examples to illustrate the convention followed by the complete legend, and tables mapping the new test filenames to old ones.

6.2.3 Naming Examples

CSIT test suite naming examples (filename.robot) for common tested VPP topologies:

1. **Physical port to physical port - a.k.a. NIC-to-NIC, Phy-to-Phy, P2P**
   - *PortNICConfig-WireEncapsulation-PacketForwardingFunction- PacketProcessingFunction1-…-PacketProcessingFunctionN-TestType*
   - *10ge2p1x520-dot1q-l2bdbasemaclrn-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, dot1q tagged Ethernet, L2 bridge-domain baseline switching with MAC learning, NDR throughput discovery.
   - *10ge2p1x520-ethip4vxlan-l2bdbasemaclrn-ndrchk.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 VXLAN Ethernet, L2 bridge-domain baseline switching with MAC learning, NDR throughput discovery.
   - *10ge2p1x520-ethip4-ip4base-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 baseline routed forwarding, NDR throughput discovery.
   - *10ge2p1x520-ethip6-ip6scale200k-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv6 scaled up routed forwarding, NDR throughput discovery.
   - *10ge2p1x520-ethip4-ip4base-iacldstbase-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 baseline routed forwarding, ingress Access Control Lists baseline matching on destination, NDR throughput discovery.
   - *40ge2p1vic1385-ethip4-ip4base-ndrdisc.robot* => 2 ports of 40GE on Cisco vic1385 NIC, IPv4 baseline routed forwarding, NDR throughput discovery.
   - *eth2p-ethip4-ip4base-func.robot* => 2 ports of Ethernet, IPv4 baseline routed forwarding, functional tests.
2. **Physical port to VM (or VM chain) to physical port - a.k.a. NIC2VM2NIC, P2V2P, NIC2VMchain2NIC, P2V2V2P**
   - *PortNICConfig-WireEncapsulation-PacketForwardingFunction- PacketProcessingFunction1-…-PacketProcessingFunctionN-VirtEncapsulation- VirtPortConfig-VMconfig-TestType*
   - *10ge2p1x520-dot1q-l2bdbasemaclrn-eth-2vhost-1vm-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, dot1q tagged Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, NDR throughput discovery.
   - *10ge2p1x520-ethip4vxlan-l2bdbasemaclrn-eth-2vhost-1vm-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, NDR throughput discovery.

- *10ge2p1x520-ethip4vxlan-l2bdbasemaclrn-eth-4vhost-2vm-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from four vhost interfaces and two VMs, NDR throughput discovery.
- *eth2p-ethip4vxlan-l2bdbasemaclrn-eth-2vhost-1vm-func.robot* => 2 ports of Ethernet, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, functional tests.

3. **API CRUD tests - Create (Write), Read (Retrieve), Update (Modify), Delete (Destroy) operations for configuration and operational data**
   - *ManagementTestKeyword-ManagementOperation-ManagedFunction1-...- ManagedFunctionN-ManagementAPI1-ManagementAPIN-TestType*
   - *mgmt-cfg-lisp-apivat-func* => configuration of LISP with VAT API calls, functional tests.
   - *mgmt-cfg-l2bd-apihc-apivat-func* => configuration of L2 Bridge-Domain with Honeycomb API and VAT API calls, functional tests.
   - *mgmt-oper-int-apihcnc-func* => reading status and operational data of interface with Honeycomb NetConf API calls, functional tests.
   - *mgmt-cfg-int-tap-apihcnc-func* => configuration of tap interfaces with Honeycomb NetConf API calls, functional tests.
   - *mgmt-notif-int-subint-apihcnc-func* => notifications of interface and sub-interface events with Honeycomb NetConf Notifications, functional tests.

For complete description of CSIT test naming convention please refer to CSIT test naming wiki page[431].

6.3 CSIT RF Tags Descriptions

All CSIT test cases are labelled with Robot Framework tags used to allow for easy test case type identification, test case grouping and selection for execution. Following sections list currently used CSIT TAGs and their documentation based on the content of tag documentation rst file[432].

6.3.1 Testbed Topology Tags

2_NODE_DOUBLE_LINK_TOPO

2 nodes connected in a circular topology with two links interconnecting the devices.

2_NODE_SINGLE_LINK_TOPO

2 nodes connected in a circular topology with at least one link interconnecting devices.

3_NODE_DOUBLE_LINK_TOPO

3 nodes connected in a circular topology with two links interconnecting the devices.

3_NODE_SINGLE_LINK_TOPO

3 nodes connected in a circular topology with at least one link interconnecting devices.

[431] https://wiki.fd.io/view/CSIT/csit-test-naming
[432] https://git.fd.io/csit/tree/docs/tag_documentation.rst?h=rls2210

6.3.2 Objective Tags

SKIP_PATCH

Test case(s) marked to not run in case of vpp-csit-verify (i.e. VPP patch) and csit-vpp-verify jobs (i.e. CSIT patch).

SKIP_VPP_PATCH

Test case(s) marked to not run in case of vpp-csit-verify (i.e. VPP patch).

6.3.3 Environment Tags

HW_ENV

DUTs and TGs are running on bare metal.

VM_ENV

DUTs and TGs are running in virtual environment.

VPP_VM_ENV

DUTs with VPP and capable of running Virtual Machine.

6.3.4 NIC Model Tags

NIC_Intel-X520-DA2

Intel X520-DA2 NIC.

NIC_Intel-XL710

Intel XL710 NIC.

NIC_Intel-X710

Intel X710 NIC.

NIC_Intel-XXV710

Intel XXV710 NIC.

NIC_Cisco-VIC-1227

VIC-1227 by Cisco.

NIC_Cisco-VIC-1385

VIC-1385 by Cisco.

NIC_Amazon-Nitro-50G

Amazon EC2 ENA NIC.

6.3.5 Scaling Tags

FIB_20K

2x10,000 entries in single fib table

FIB_200K

2x100,000 entries in single fib table

FIB_2M

2x1,000,000 entries in single fib table

L2BD_1

Test with 1 L2 bridge domain.

L2BD_10

Test with 10 L2 bridge domains.

L2BD_100

Test with 100 L2 bridge domains.

L2BD_1K

Test with 1000 L2 bridge domains.

VLAN_1

Test with 1 VLAN sub-interface.

VLAN_10

Test with 10 VLAN sub-interfaces.

VLAN_100

Test with 100 VLAN sub-interfaces.

VLAN_1K

Test with 1000 VLAN sub-interfaces.

VXLAN_1

Test with 1 VXLAN tunnel.

VXLAN_10

Test with 10 VXLAN tunnels.

VXLAN_100

Test with 100 VXLAN tunnels.

VXLAN_1K

Test with 1000 VXLAN tunnels.

TNL_{t}

IPSec in tunnel mode - {t} tunnels.

SRC_USER_{u}

Traffic flow with {u} unique IPs (users) in one direction. {u}=(1,10,100,1000,2000,4000).

100_FLOWS

Traffic stream with 100 unique flows (10 IPs/users x 10 UDP ports) in one direction.

10k_FLOWS

Traffic stream with 10 000 unique flows (10 IPs/users x 1000 UDP ports) in one direction.

100k_FLOWS

Traffic stream with 100 000 unique flows (100 IPs/users x 1000 UDP ports) in one direction.

HOSTS_{h}

Stateless or stateful traffic stream with {h} client source IP4 addresses, usually with 63 flow differing in source port number. Could be UDP or TCP. If NAT is used, the clients are inside. Outside IP range can differ. {h}=(1024,4096,16384,65536,262144).

GENEVE4_{t}TUN

Test with {t} GENEVE IPv4 tunnel. {t}=(1,4,16,64,256,1024)

6.3.6 Test Category Tags

DEVICETEST

All vpp_device functional test cases.

PERFTEST

All performance test cases.

6.3.7 VPP Device Type Tags

SCAPY

All test cases that uses Scapy for packet generation and validation.

6.3.8 Performance Type Tags

NDRPDR

Single test finding both No Drop Rate and Partial Drop Rate simultaneously. The search is done by optimized algorithm which performs multiple trial runs at different durations and transmit rates. The results come from the final trials, which have duration of 30 seconds.

MRR

Performance tests where TG sends the traffic at maximum rate (line rate) and reports total sent/received packets over trial duration. The result is an average of 10 trials of 1 second duration.

SOAK

Performance tests using PLRsearch to find the critical load.

RECONF

Performance tests aimed to measure lost packets (time) when performing reconfiguration while full throughput offered load is applied.

6.3.9 Ethernet Frame Size Tags

These are describing the traffic offered by Traffic Generator, "primary" traffic in case of asymmetric load. For traffic between DUTs, or for "secondary" traffic, see ${overhead} value.

{b}B

{b} Bytes frames used for test.

IMIX

IMIX frame sequence (28x 64B, 16x 570B, 4x 1518B) used for test.

6.3.10 Test Type Tags

BASE

Baseline test cases, no encapsulation, no feature(s) configured in tests. No scaling whatsoever, beyond minimum needed for RSS.

IP4BASE

IPv4 baseline test cases, no encapsulation, no feature(s) configured in tests. Minimal number of routes. Other quantities may be scaled.

IP6BASE

IPv6 baseline test cases, no encapsulation, no feature(s) configured in tests.

L2XCBASE

L2XC baseline test cases, no encapsulation, no feature(s) configured in tests.

L2BDBASE

L2BD baseline test cases, no encapsulation, no feature(s) configured in tests.

L2PATCH

L2PATCH baseline test cases, no encapsulation, no feature(s) configured in tests.

SCALE

Scale test cases. Other tags specify which quantities are scaled. Also applies if scaling is set on TG only (e.g. DUT works as IP4BASE).

ENCAP

Test cases where encapsulation is used. Use also encapsulation tag(s).

FEATURE

At least one feature is configured in test cases. Use also feature tag(s).

UDP

Tests which use any kind of UDP traffic (STL or ASTF profile).

TCP

Tests which use any kind of TCP traffic (STL or ASTF profile).

TREX

Tests which test trex traffic without any software DUTs in the traffic path.

UDP_UDIR

Tests which use unidirectional UDP traffic (STL profile only).

UDP_BIDIR

Tests which use bidirectional UDP traffic (STL profile only).

UDP_CPS

Tests which measure connections per second on minimal UDP pseudoconnections. This implies ASTF traffic profile is used. This tag selects specific output processing in PAL.

TCP_CPS

Tests which measure connections per second on empty TCP connections. This implies ASTF traffic profile is used. This tag selects specific output processing in PAL.

TCP_RPS

Tests which measure requests per second on empty TCP connections. This implies ASTF traffic profile is used. This tag selects specific output processing in PAL.

UDP_PPS

Tests which measure packets per second on lightweight UDP transactions. This implies ASTF traffic profile is used. This tag selects specific output processing in PAL.

TCP_PPS

Tests which measure packets per second on lightweight TCP transactions. This implies ASTF traffic profile is used. This tag selects specific output processing in PAL.

HTTP

Tests which use traffic formed of valid HTTP requests (and responses).

LDP_NGINX

LDP NGINX is un-modified NGINX with VPP via LD_PRELOAD.

NF_DENSITY

Performance tests that measure throughput of multiple VNF and CNF service topologies at different service densities.

6.3.11 NF Service Density Tags

CHAIN

NF service density tests with VNF or CNF service chain topology(ies).

PIPE

NF service density tests with CNF service pipeline topology(ies).

NF_L3FWDIP4

NF service density tests with DPDK l3fwd IPv4 routing as NF workload.

NF_VPPIP4

NF service density tests with VPP IPv4 routing as NF workload.

{r}R{c}C

Service density matrix locator {r}R{c}C, {r}Row denoting number of service instances, {c}Column denoting number of NFs per service instance. {r}=(1,2,4,6,8,10), {c}=(1,2,4,6,8,10).

{n}VM{t}T

Service density {n}VM{t}T, {n}Number of NF Qemu VMs, {t}Number of threads per NF.

{n}DCRt}T

Service density {n}DCR{t}T, {n}Number of NF Docker containers, {t}Number of threads per NF.

{n}_ADDED_CHAINS

{n}Number of chains (or pipelines) added (and/or removed) during RECONF test.

6.3.12 Forwarding Mode Tags

L2BDMACSTAT

VPP L2 bridge-domain, L2 MAC static.

L2BDMACLRN

VPP L2 bridge-domain, L2 MAC learning.

L2XCFWD

VPP L2 point-to-point cross-connect.

IP4FWD

VPP IPv4 routed forwarding.

IP6FWD

VPP IPv6 routed forwarding.

LOADBALANCER_MAGLEV

VPP Load balancer maglev mode.

LOADBALANCER_L3DSR

VPP Load balancer l3dsr mode.

LOADBALANCER_NAT4

VPP Load balancer nat4 mode.

N2N

Mode, where NICs from the same physical server are directly connected with a cable.

6.3.13 Underlay Tags

IP4UNRLAY

IPv4 underlay.

IP6UNRLAY

IPv6 underlay.

MPLSUNRLAY

MPLS underlay.

6.3.14 Overlay Tags

L2OVRLAY

L2 overlay.

IP4OVRLAY

IPv4 overlay (IPv4 payload).

IP6OVRLAY

IPv6 overlay (IPv6 payload).

6.3.15 Tagging Tags

DOT1Q

All test cases with dot1q.

DOT1AD

All test cases with dot1ad.

6.3.16 Encapsulation Tags

ETH

All test cases with base Ethernet (no encapsulation).

LISP

All test cases with LISP.

LISPGPE

All test cases with LISP-GPE.

LISP_IP4o4

All test cases with LISP_IP4o4.

LISPGPE_IP4o4

All test cases with LISPGPE_IP4o4.

LISPGPE_IP6o4

All test cases with LISPGPE_IP6o4.

LISPGPE_IP4o6

All test cases with LISPGPE_IP4o6.

LISPGPE_IP6o6

All test cases with LISPGPE_IP6o6.

VXLAN

All test cases with Vxlan.

VXLANGPE

All test cases with VXLAN-GPE.

GRE

All test cases with GRE.

GTPU

All test cases with GTPU.

GTPU_HWACCEL

All test cases with GTPU_HWACCEL.

IPSEC

All test cases with IPSEC.

WIREGUARD

All test cases with WIREGUARD.

SRv6

All test cases with Segment routing over IPv6 dataplane.

SRv6_1SID

All SRv6 test cases with single SID.

SRv6_2SID_DECAP

All SRv6 test cases with two SIDs and with decapsulation.

SRv6_2SID_NODECAP

All SRv6 test cases with two SIDs and without decapsulation.

GENEVE

All test cases with GENEVE.

GENEVE_L3MODE

All test cases with GENEVE tunnel in L3 mode.

FLOW

All test cases with FLOW.

FLOW_DIR

All test cases with FLOW_DIR.

FLOW_RSS

All test cases with FLOW_RSS.

NTUPLE

All test cases with NTUPLE.

L2TPV3

All test cases with L2TPV3.

6.3.17 Interface Tags

PHY

All test cases which use physical interface(s).

GSO

All test cases which uses Generic Segmentation Offload.

VHOST

All test cases which uses VHOST.

VHOST_1024

All test cases which uses VHOST DPDK driver with qemu queue size set to 1024.

VIRTIO

All test cases which uses VIRTIO native VPP driver.

VIRTIO_1024

All test cases which uses VIRTIO native VPP driver with qemu queue size set to 1024.

CFS_OPT

All test cases which uses VM with optimised scheduler policy.

TUNTAP

All test cases which uses TUN and TAP.

AFPKT

All test cases which uses AFPKT.

NETMAP

All test cases which uses Netmap.

MEMIF

All test cases which uses Memif.

SINGLE_MEMIF

All test cases which uses only single Memif connection per DUT. One DUT instance is running in container having one physical interface exposed to container.

LBOND

All test cases which uses link bonding (BondEthernet interface).

LBOND_DPDK

All test cases which uses DPDK link bonding.

LBOND_VPP

All test cases which uses VPP link bonding.

LBOND_MODE_XOR

All test cases which uses link bonding with mode XOR.

LBOND_MODE_LACP

All test cases which uses link bonding with mode LACP.

LBOND_LB_L34

All test cases which uses link bonding with load-balance mode l34.

LBOND_{n}L

All test cases which use {n} link(s) for link bonding.

DRV_{d}

All test cases which NIC Driver for DUT is set to {d}. Default is VFIO_PCI. {d}=(AVF, RDMA_CORE, VFIO_PCI, AF_XDP).

TG_DRV_{d}

All test cases which NIC Driver for TG is set to {d}. Default is IGB_UIO. {d}=(RDMA_CORE, IGB_UIO).

RXQ_SIZE_{n}

All test cases which RXQ size (RX descriptors) are set to {n}. Default is 0, which means VPP (API) default.

TXQ_SIZE_{n}

All test cases which TXQ size (TX descriptors) are set to {n}. Default is 0, which means VPP (API) default.

6.3.18 Feature Tags

IACLDST

iACL destination.

ADLALWLIST

ADL allowlist.

NAT44

NAT44 configured and tested.

NAT64

NAT44 configured and tested.

ACL

ACL plugin configured and tested.

IACL

ACL plugin configured and tested on input path.

OACL

ACL plugin configured and tested on output path.

ACL_STATELESS

ACL plugin configured and tested in stateless mode (permit action).

ACL_STATEFUL

ACL plugin configured and tested in stateful mode (permit+reflect action).

ACL1

ACL plugin configured and tested with 1 not-hitting ACE.

ACL10

ACL plugin configured and tested with 10 not-hitting ACEs.

ACL50

ACL plugin configured and tested with 50 not-hitting ACEs.

SRv6_PROXY

SRv6 endpoint to SR-unaware appliance via proxy.

SRv6_PROXY_STAT

SRv6 endpoint to SR-unaware appliance via static proxy.

SRv6_PROXY_DYN

SRv6 endpoint to SR-unaware appliance via dynamic proxy.

SRv6_PROXY_MASQ

SRv6 endpoint to SR-unaware appliance via masquerading proxy.

6.3.19 Encryption Tags

IPSECSW

Crypto in software.

IPSECHW

Crypto in hardware.

IPSECTRAN

IPSec in transport mode.

IPSECTUN

IPSec in tunnel mode.

IPSECINT

IPSec in interface mode.

AES

IPSec using AES algorithms.

AES_128_CBC

IPSec using AES 128 CBC algorithms.

AES_128_GCM

IPSec using AES 128 GCM algorithms.

AES_256_GCM

IPSec using AES 256 GCM algorithms.

HMAC

IPSec using HMAC integrity algorithms.

HMAC_SHA_256

IPSec using HMAC SHA 256 integrity algorithms.

HMAC_SHA_512

IPSec using HMAC SHA 512 integrity algorithms.

SCHEDULER

IPSec using crypto sw scheduler engine.

6.3.20 Client-Workload Tags

VM

All test cases which use at least one virtual machine.

LXC

All test cases which use Linux container and LXC utils.

DRC

All test cases which use at least one Docker container.

DOCKER

All test cases which use Docker as container manager.

APP

All test cases with specific APP use.

6.3.21 Container Orchestration Tags

{n}VSWITCH

{n} VPP running in {n} Docker container(s) acting as a VSWITCH. {n}=(1).

{n}VNF

{n} VPP running in {n} Docker container(s) acting as a VNF work load. {n}=(1).

6.3.22 Multi-Threading Tags

STHREAD

Dynamic tag. All test cases using single poll mode thread.

MTHREAD

Dynamic tag.
All test cases using more then one poll mode driver thread.

{n}NUMA

All test cases with packet processing on {n} socket(s). {n}=(1,2).

{c}C

{c} worker thread pinned to {c} dedicated physical core; or if HyperThreading is enabled, {c}*2 worker threads each pinned to a separate logical core within 1 dedicated physical core. Main thread pinned to core 1. {t}=(1,2,4).

{t}T{c}C

Dynamic tag.

{t} worker threads pinned to {c} dedicated physical cores. Main thread pinned to core 1. By default CSIT is configuring same amount of receive queues per interface as worker threads. {t}=(1,2,4,8), {t}=(1,2,4).

BIBLIOGRAPHY

[lxc] Linux Containers[38]

[lxcnamespace] Resource management: Linux kernel Namespaces and cgroups[39].

[stgraber] LXC 1.0: Blog post series[40].

[lxcsecurity] Linux Containers Security[41].

[capabilities] Linux manual - capabilities - overview of Linux capabilities[42].

[cgroup1] Linux kernel documentation: cgroups[43].

[cgroup2] Linux kernel documentation: Control Group v2[44].

[selinux] SELinux Project Wiki[45].

[lxcsecfeatures] LXC 1.0: Security features[46].

[lxcsource] Linux Containers source[47].

[apparmor] Ubuntu AppArmor[48].

[seccomp] SECure COMPuting with filters[49].

[docker] Docker[50].

[k8sdoc] Kubernetes documentation[51].

[38] https://linuxcontainers.org/
[39] https://www.cs.ucsb.edu/~rich/class/cs293b-cloud/papers/lxc-namespace.pdf
[40] https://stgraber.org/2013/12/20/lxc-1-0-blog-post-series/
[41] https://linuxcontainers.org/lxc/security/
[42] http://man7.org/linux/man-pages/man7/capabilities.7.html
[43] https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt
[44] https://www.kernel.org/doc/Documentation/cgroup-v2.txt
[45] http://selinuxproject.org/page/Main_Page
[46] https://stgraber.org/2014/01/01/lxc-1-0-security-features/
[47] https://github.com/lxc/lxc
[48] https://wiki.ubuntu.com/AppArmor
[49] https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt
[50] https://www.docker.com/what-docker
[51] https://kubernetes.io/docs/home/