Born Ready for Secure Terabit Internet! Tooling for Benchmarking.

CSIT Project – Automated Open-Source FD.io benchmarking.
csit-dev@lists.fd.io

Maciek Konstantynowicz, CSIT Project Technical Lead

Fast Data I/O

- Importance of Benchmarking
- Processing Packets in Compute
- Open Source Benchmarking
- CSIT Design and Tooling
- Benchmark Methodologies
- Sample Measurements
- Future Work

The Importance of Benchmarking

Observations and measurements underpin all major discoveries of laws of nature.

900s BC, China, "I Ching", yin "0" and yang "1", Fu Xi "Earlier Heaven" binary sequence.

1543, Poland, N.Copernicus, "De Revolutionibus", Earth not the centre of the universe, start of scientific Rennaisance.

1679, Germany, G.Leibniz "Explication de l'Arithmétique Binaire", binary numeral system

1854, Ireland, G.Boole, "An Investigation of the Laws of Thought", Boolean logic

1937, US, C.Shannon, "A Symbolic Analysis of Relay and Switching Circuits", digital circuit design

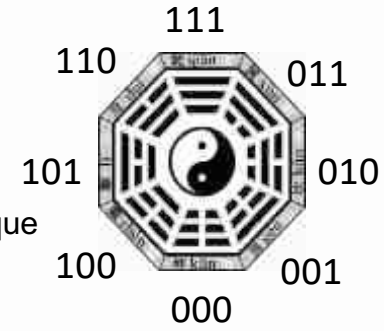
1940, US, G.Stiblitz, "Model K" relay-based computer and remote commands over the telephone line

1945, US, J.Von Neuman, A.Turing et al, EDVAC, ENIAC and birth of computing.

Transistors and Moore's Law … Fast forward to today

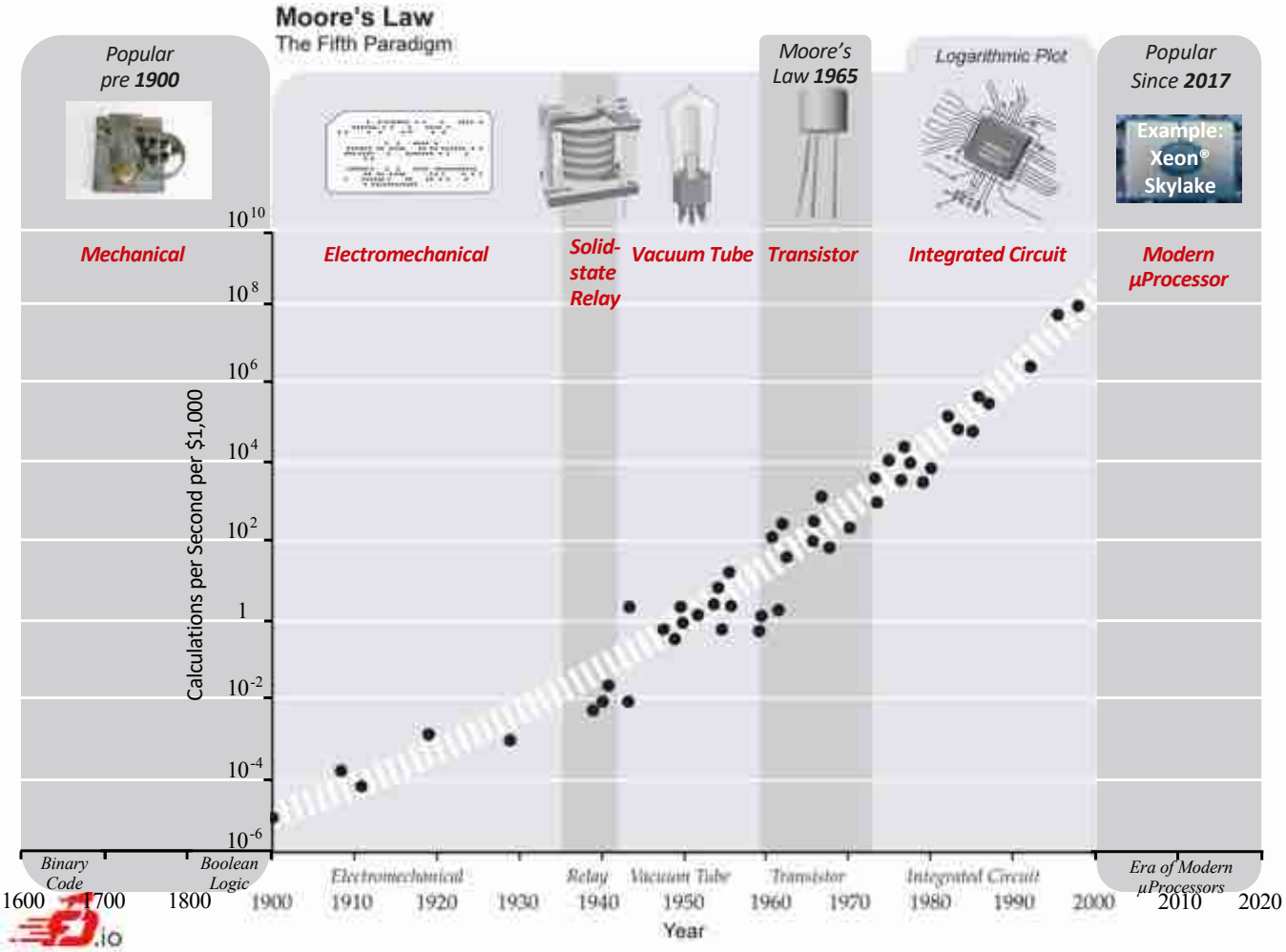
…

What happens next?



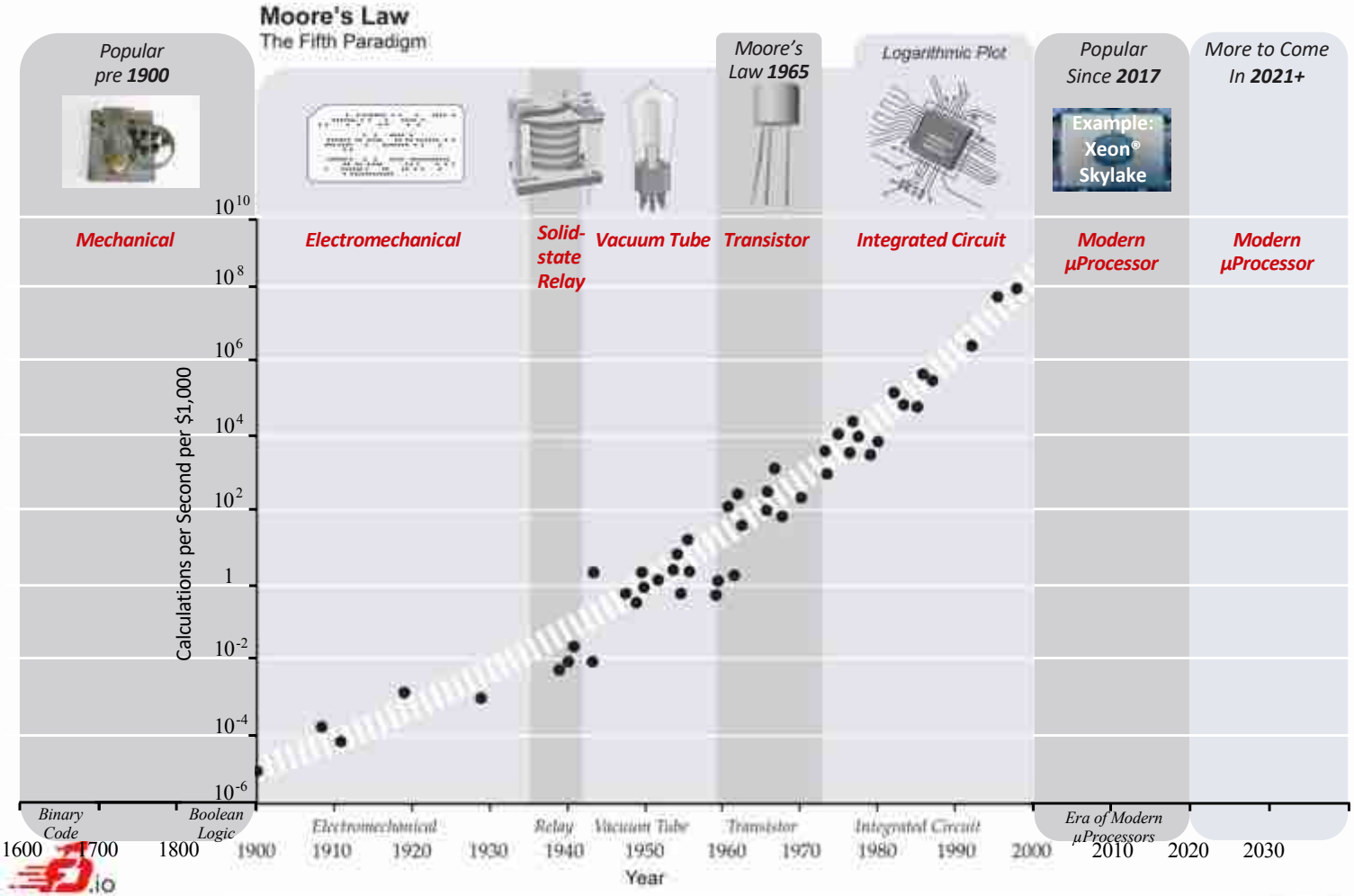
Benchmarking is foundational to scientific and technological progress.

Remember **1965** ”Moore’s Law” – Does It Still Work?



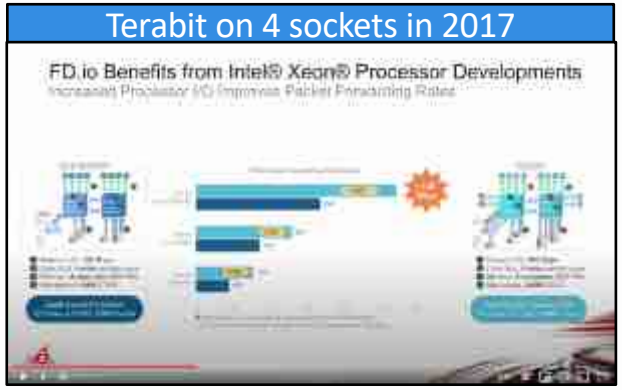
Source: Ray Kurzweil, "The Singularity Is Near: When Humans Transcend Biology", Page 67, The Duckworth Publishers 2009. Data points between 1600 and 1900, and after 2000 represent presenter’s perspective and approximations.

Remember **1965** ”Moore’s Law” – Well, It Surely Does Ramble on …



…

Terabit on 4 sockets in 2017



https://www.youtube.com/watch?v=aLJ0XLeV3V4

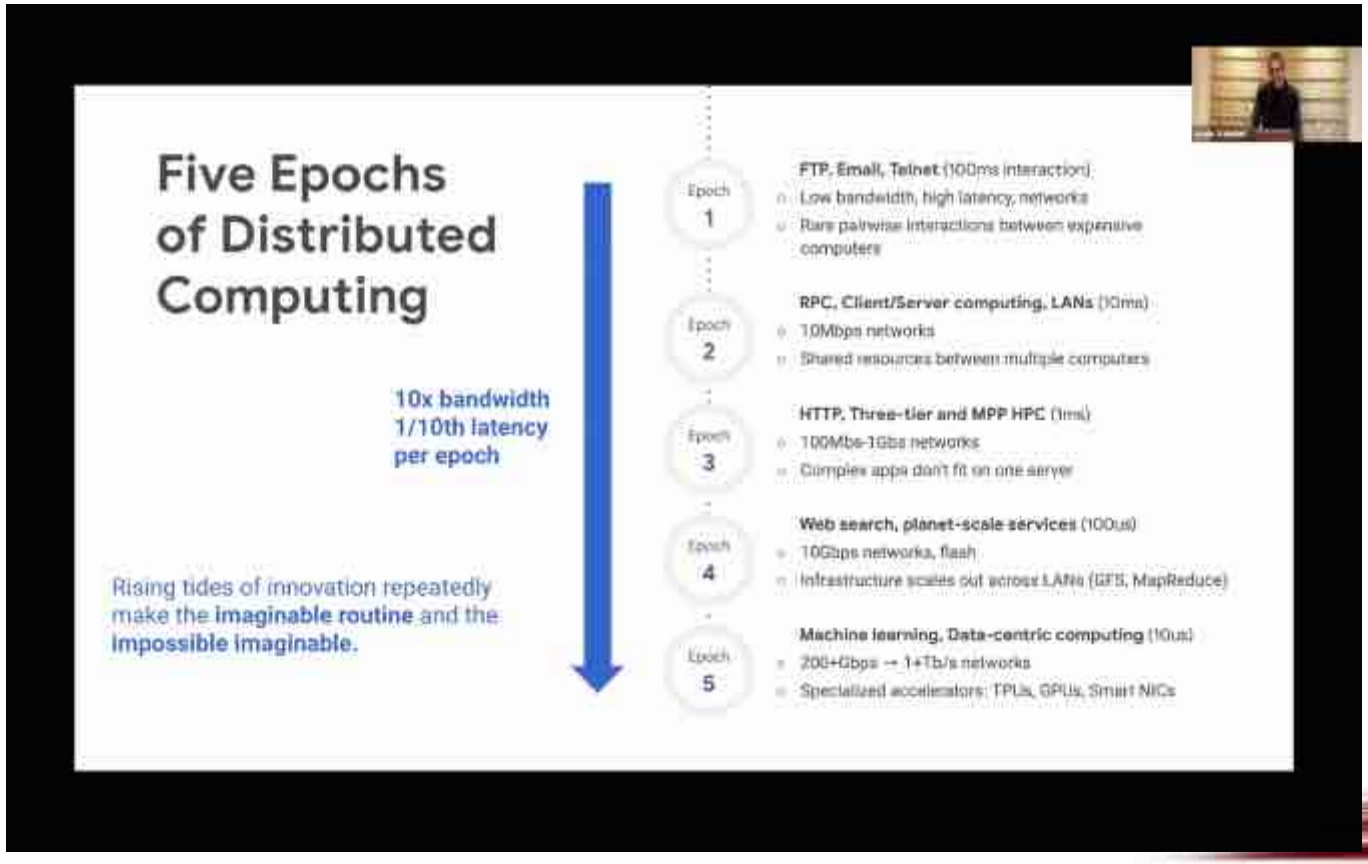
How long before Secure Terabit on 2 sockets … ?

Source: Ray Kurzweil, "The Singularity Is Near: When Humans Transcend Biology", Page 67, The Duckworth Publishers 2009. Data points between 1600 and 1900, and after 2000 represent presenter’s perspective and approximations.

We seem to be entering the Epoch 5 of distributed computing.

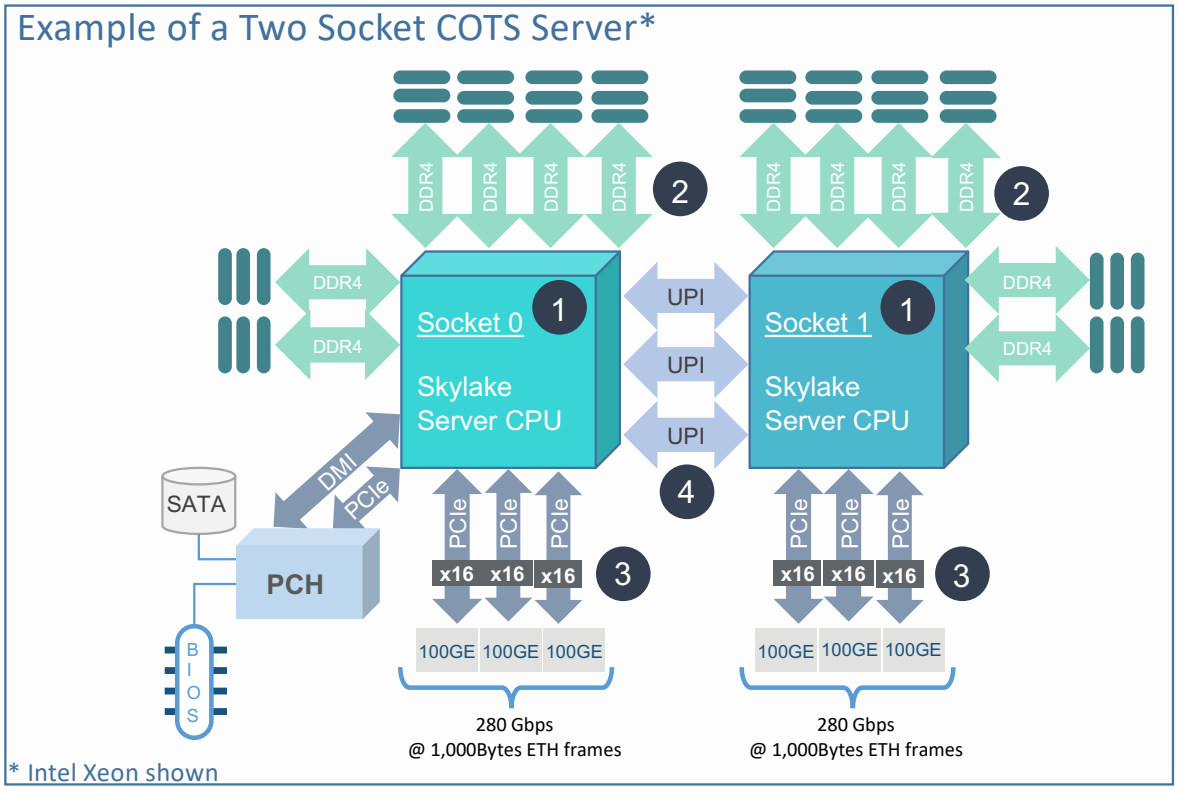
And moving to 10 usec RPC and 1+ Tbps networking.

What does it mean for software networking?



SIGCOMM 2020 Keynote: **Amin Vadhat**: Coming of Age in the Fifth Epoch of Distributed Computing
https://youtu.be/27zuReojDVw?t=601

Processing Packets: How to Use Compute ..



Resources to Get Performance

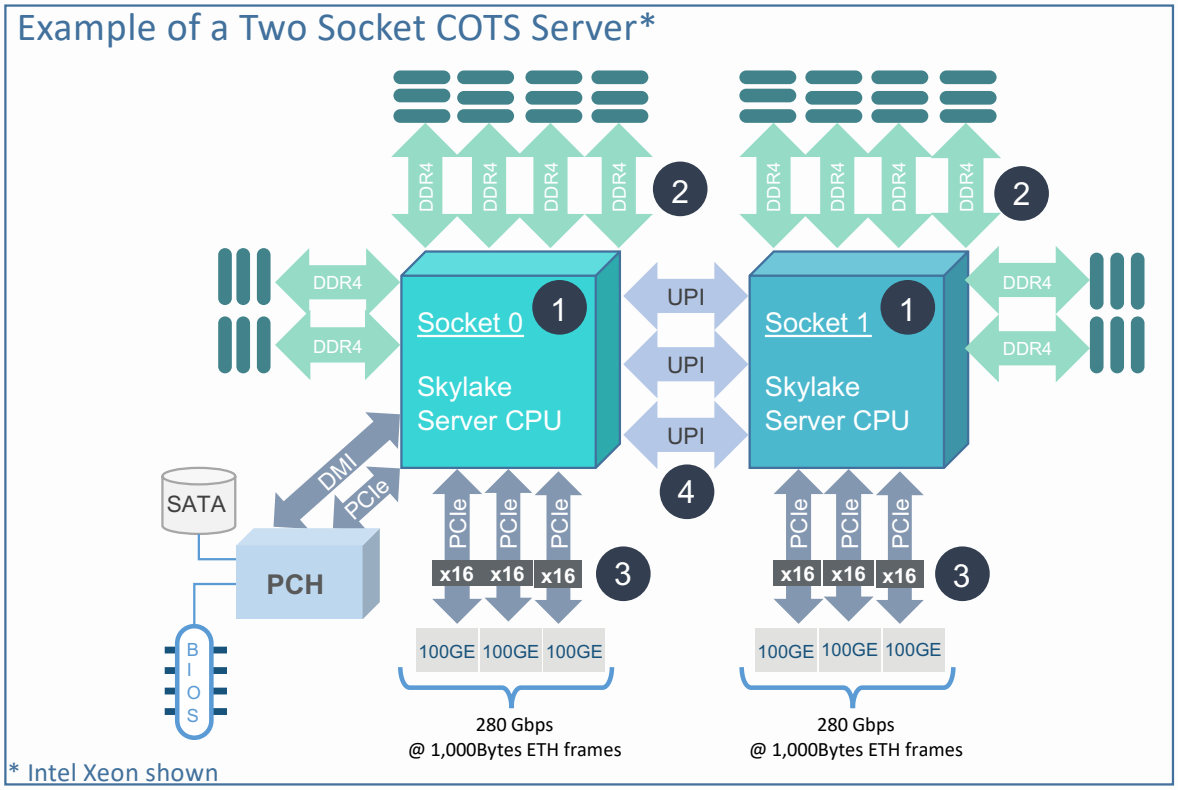
1. **Processor and CPU cores**
   for performing packet processing operations
2. **Memory bandwidth**
   for moving data (packets, lookup) and instructions (packet processing)
3. **I/O bandwidth**
   for moving packets to/from NIC interfaces
4. **Inter-socket bandwidth**
   for handling inter-socket operations

$$CyclesPerPacket\ [ClockCycles] = \frac{\#Instructions}{Packet} * \frac{\#Cycles}{Instruction}$$

$$Throughput\ [pps] = \frac{1}{Packet\_Processing\_Time[sec]} = \frac{CPU\_freq[Hz]}{Cycles\_per\_Packet}$$

$$Throughput\ [bps] = Throughput[pps] * Packet\_Size[pps]$$

Processing Packets: What Improves in Compute …



Resources to Get Performance

1. **Processor and CPU cores**

   FrontEnd:* faster instr. decoder (4- to 5-wide).

   BackEnd:* faster L1 cache, bigger L2 cache, deeper OOO* execution.

   Uncore: move from ring to X-Y fabric mesh.

2. **Memory bandwidth**

   ***~50% increase**:* channels (4 to 6), speed (DDR-2666).

3. **I/O bandwidth**

   >50% increase: PCIe Gen3 lanes (40 to 48), re-designed IO blocks.

4. **Inter-socket bandwidth**

   ~60% increase: QPI to UPI (2x to 3x), interface speed (9.6 to 10.4 GigTrans/sec).

*** Changes in Xeon® from BDX to SKX.**

$$CyclesPerPacket\ [ClockCycles] = \frac{\#Instructions}{Packet} * \frac{\#Cycles}{Instruction}$$

$$Throughput\ [pps] = \frac{1}{Packet\_Processing\_Time[sec]} = \frac{CPU\_freq[Hz]}{Cycles\_per\_Packet}$$

$$Throughput\ [bps] = Throughput[pps] * Packet\_Size[pps]$$

Open Source Benchmarking – Guiding Principles

- Discover the physical system **limits** and **know** them.
- Evaluate based on **externally measured data** and behaviour (black-box).
- Guide benchmarking by **good understanding** of the whole system (white-box).
- Provide **feedback loop** to developers and users.
- Be the good guys, **accelerate progress**.

> "For better or worse, benchmarks shape a field."
> – David A. Patterson, John L. Hennessy

Introducing FastData.io Sub-Project: **VPP**

Vector Packet Processing

Fast packet processing software platform that runs in Linux user space

Performance

VPP on x86 servers **outperforms** specialized packet processing HW.

Portability

VPP runs in any hardware

VPP runs in any environment, **bare-metal, VM, containers**.

Platforms

Platform for building forwarding applications, network functions, plug-ins as first class citizens, and configurable graph architecture.

LFNETWORKING

Open source collaborative project (FD.io) in Linux Foundation.

SDN

Software **programmable**, **extendable** and **flexible.**

Efficiency

Ability to **upscale** and **downscale.**

Introducing FastData.io Sub-Project: **CSIT**

Continous System Integration Testing

Open Source Continuous Benchmarking for evaluating data plane code

Set Goals

- Drive good practice and engineering discipline into FD.io dev community.
- Drive innovative optimizations into the source code, verify they work.
- Provide conformance, performance and efficiency metrics to track.
- Help to ensure code quality.

Then Execute

1. **Fully automated testing**
   a. FD.io: VPP, DPDK L3fwd, DPDK Testpmd.
   b. Functionality, drivers, performance.

f(x)

2. **Functionality tests**
   a. Network data, control and management planes.
   b. Conformance against IETF RFC specs.

3. **Performance tests**
   a. Data plane throughput and latency.
   b. Trending of data plane throughput limits.

4. **Use case driven test definitions**
   a. Uni-dimensional: data, control, management.
   b. Multi-dimensional tests: use case driven.

5. **Test execution in LFN FD.io environment**
   a. Performance tests on physical servers in LFN DCs.
   b. Functional tests on Containers hosted in LFN DCs.

6. **Integration with LFN FD.io CI system**
   a. Gerrit and Jenkins CI.
   b. Automated test execution triggers.

FD.io CSIT Continuous Benchmarking

Open Source Benchmarking for evaluating data plane code

Benchmark Test Areas

- L2 Ethernet Switching (up to 1mil MACs)
- IPv4, IPv6 Routing (up to 2mil routes)
- IPsec IPv4 Routing (up to 60k IPsec tunnels)
- SRv6 Routing
- Features: ACLs, NAT44-EI/ED, Policer, …
- IPv4, IPv6 Tunnels
- KVM VMs vHost-user (up to box-full)
- LXC.DRC Container Memif (up to box-full)

Test Methodology

- Packet Throughput
- Stateless and stateful
- Speedup Multi-Core
- Packet Latency
- Soak Tests
- Reconfiguration Tests
- NFV Service Density
- Host-stack Testing

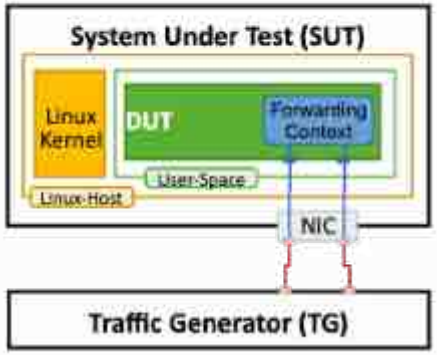
Logical Topologies

A. NIC-to-NIC Routing/Switching

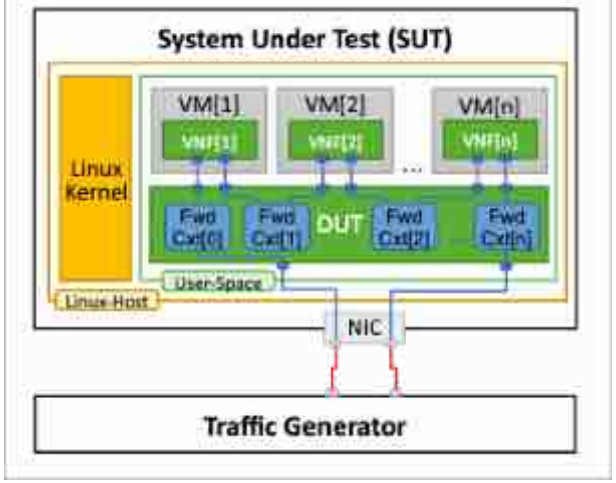
B. VM Service Routing/Switching

C. Container Service Routing/Switching

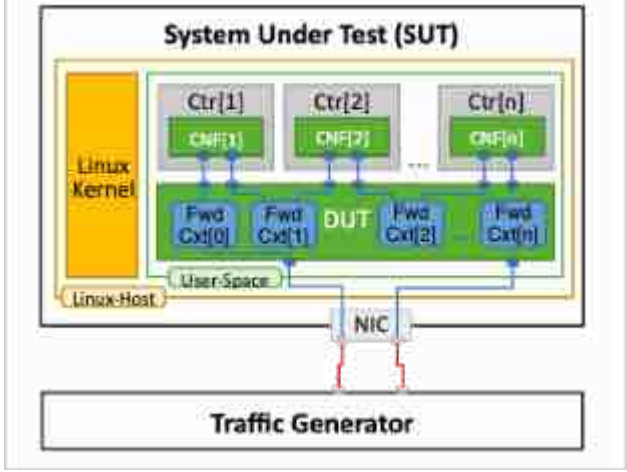
A. 2-Node Topology: NIC-to-NIC Switching



B. 2-Node Topology: VM Service Switching



C. 2-Node Topology: Container Service Switching



FD.io CSIT Continuous Benchmarking

Servers, Processors, Cloud Instances

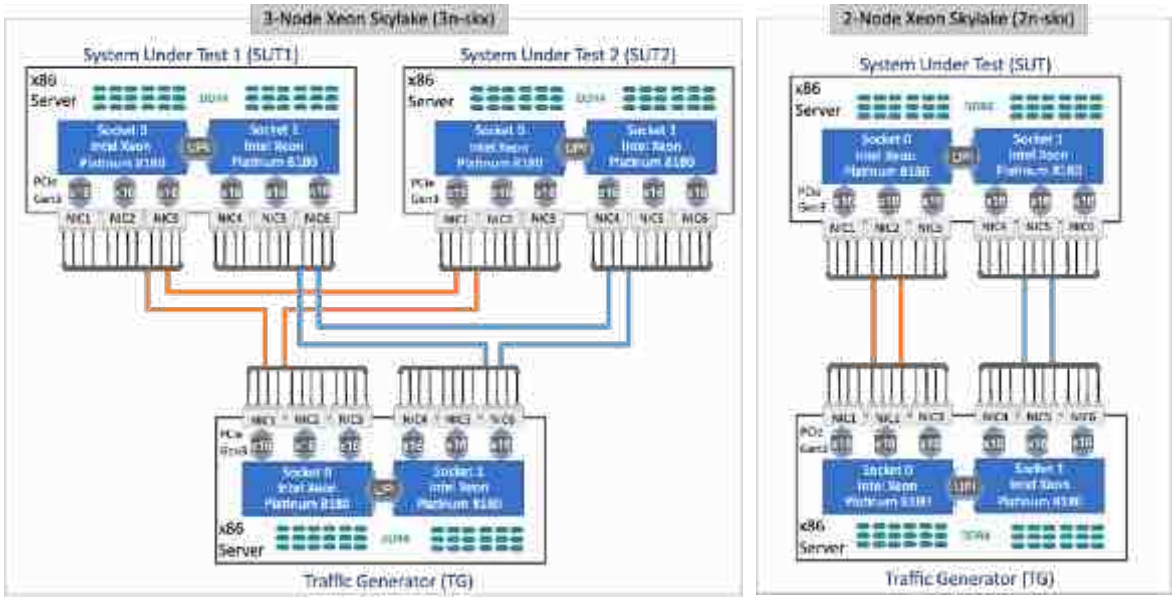
- UCS c240, SuperMicro, Intel, Taishan, AMD, ThunderX2.
- Intel Xeon Haswell, Skylake, Cascadelake (Icelake coming).
- Intel Atom Denverton.
- Arm Cortex-A72 Taishan, CN9900 ThunderX2, (N1 Ampere coming).
- AMD EPYC Zen2.
- (WIP) AWS EC2 c5n instances (Skylake[2]), AWS m6g instances Graviton2[2] with Arm Neoverse N1 cores.

Drivers, NICs

- AVF native[1], DPDK: Intel FVL 4p10GE, 2p25GE, 2p40GE.
- DPDK: Cisco VIC 1227 2p10GE, VIC 1385 2p40GE.
- Rdma_core native[1]: Mellanox CX5 cx556a 2p100GE.
- ENA DPDK[2]: AWS Nitro.
- MLX DPDK[2]: Azure Custom MLX4/5.

Physical Test Environments

- **3-Node**, 2 x System Under Test (SUT), 1 x Traffic Generator (TG).
- **2-Node**, 1 x SUT (Server), 1 x TG (TRex).

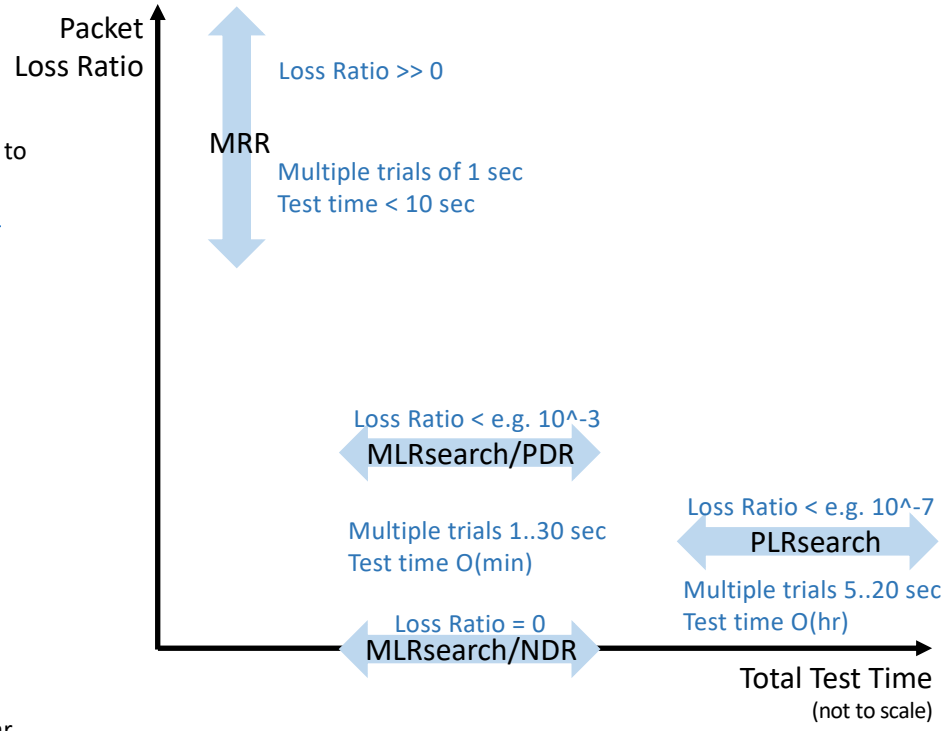


- **3-Node** used for complex encapsulations (e.g. IPsec) and cloud scenarios.
- **2-Node** for everything else.

[1] VPP native drivers.
[2] Experimental work.

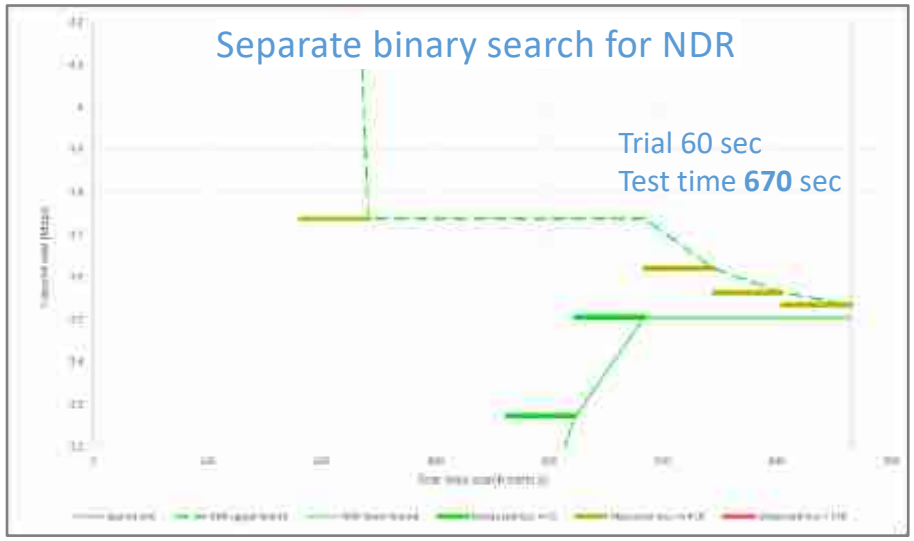
Throughput Test Methodologies: MRR, MLRsearch, PLRsearch

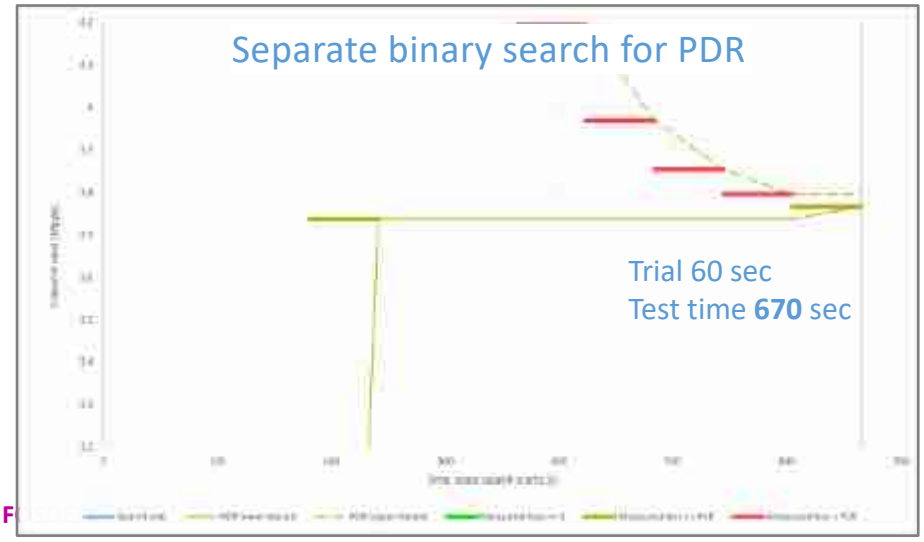
- **Maximum Receive Rate (MRR)**
  - Measures packet throughput under the maximum offered load regardless of packet loss.
  - In CSIT used for daily trending and automated anomaly detection.
- **Multiple Loss Ratio search (MLRsearch)**
  - Discovers multiple packet throughput rates (NDR, PDR[1]) in a single search to optimize overall benchmark duration.
  - IETF BMWG ref: https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch.
- **Probabilistic Loss Ratio search (PLRsearch)**
  - "Soak[2] testing" methodology for data planes over an extended period of time.
  - IETF BMWG ref: https://tools.ietf.org/html/draft-vpolak-bmwg-plrsearch.
- **All throughput measured today with TRex**
  - Majority of tests are stateless packet flow.
  - New stateful UDP and TCP/IP tests introduced recently using the same algorithms for throughput and CPS (NAT-44 Endpoint Dependent tests).
- **Why does one need MRR, PDR, NDR, Soak?**
  - A well behaving system should have MRR, PDR, NDR and Soak rates not far from each other!

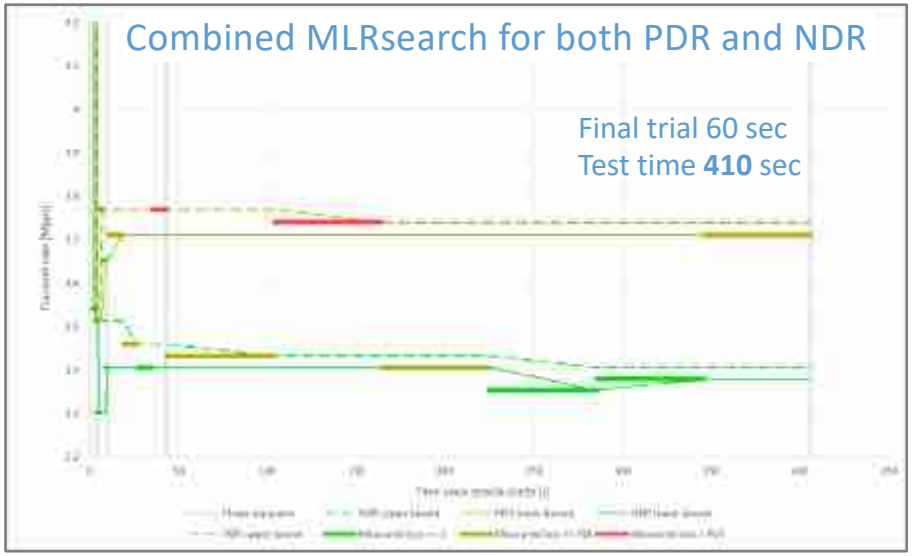


[1] Packet rate and bandwidth measurements: Non-Drop Rate (NDR, with zero packet loss), ii) Partial Drop Rate (PDR, with packet loss rate not greater than configured non-zero packet-loss-ratio).
[2] CSIT Soak tests use PLRsearch with packet-loss-ratio set to 10^-7.

MLRsearch vs. Binary Search – Test Time Comparisons*







MLRsearch yields > 3 times shorter test duration vs. two separate binary searches.

*Source: https://docs.fd.io/csit/rls1804/report/vpp_performance_tests/mdr_search.html

MLRsearch vs. Binary Search – Test Time Comparisons*

Search part of test duration.

Duration+-avgdev [s]	IP4 10s	IP4 30s	IP4 60s	Vhost 10s	Vhost 30s	Vhost 60s
MLRsearch (NDR, PDR)	50.8+-1.2	109.0+-10.0	202.8+-11.7	80.5+-9.0	201.9+-20.6	474.9+-58.2
NDR binary	98.9+-0.1	278.6+-0.1	548.8+-0.1	119.8+-0.1	339.3+-0.1	669.6+-0.2
PDR binary	98.9+-0.1	278.6+-0.1	548.8+-0.1	119.7+-0.1	339.3+-0.1	669.5+-0.1
NDR+PDR sum	197.8+-0.1	557.2+-0.2	1097.6+-0.1	239.5+-0.1	678.7+-0.1	1339.2+-0.1

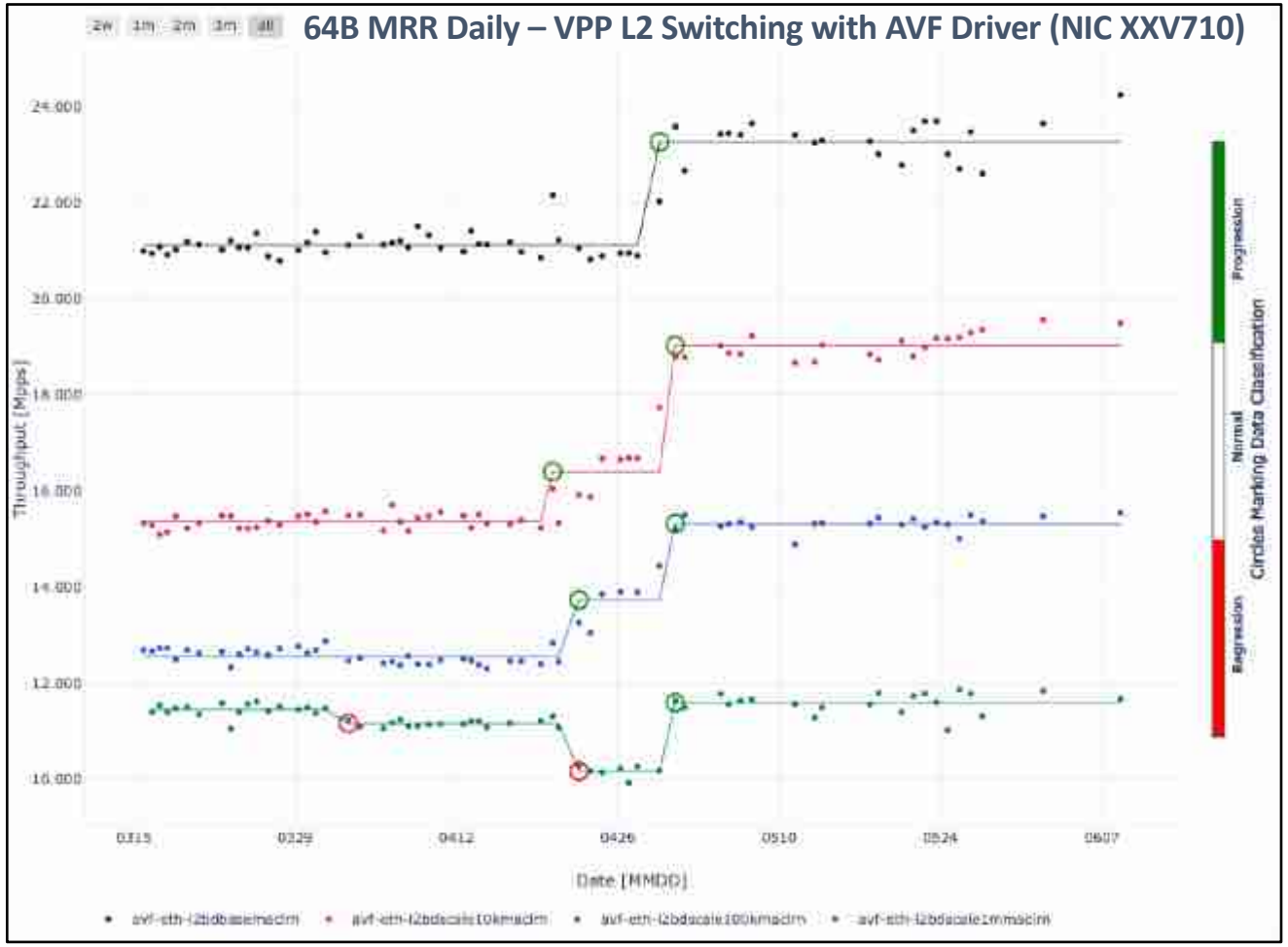
MLRsearch Gain:
Total test time
3 to 5x shorter

DUT: VPP release 18.04
IP4: VPP test - IPv4 routing baseline
Vhost: VPP test - vhostuser VM L2 bridging baseline

*Source: https://docs.fd.io/csit/rls1804/report/vpp_performance_tests/mdr_search.html

VPP Throughput Trending – Daily and Weekly

Example: auto-detection of progressions and regressions in v20.05 release cycle



Performance Trending

- *Daily*: MRR throughput measurements
- *Weekly*: NDRPDR throughput measurements
- Machine based anomaly detection
- Correlation to VPP and CSIT builds
- Results in graphs and notification emails

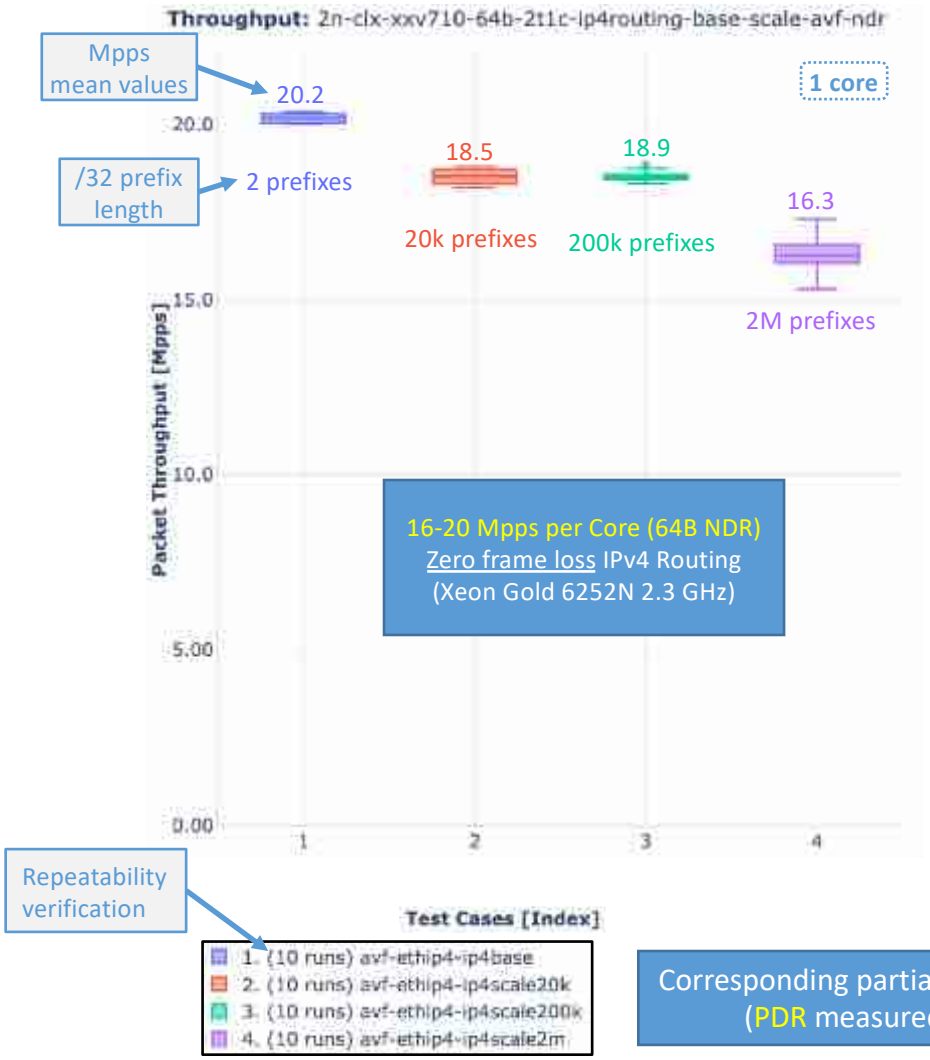
Automated Anomaly Bisecting

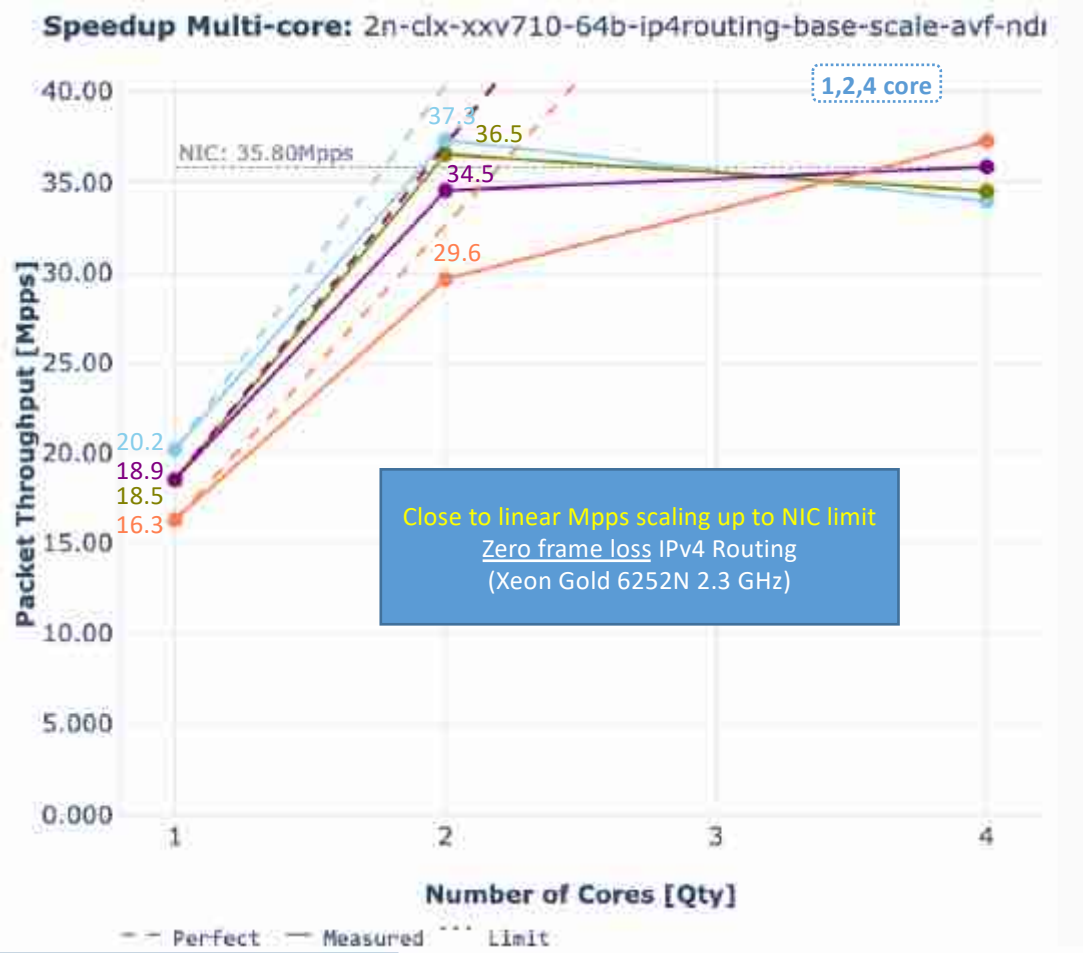
- Jobs for bisecting of regressions, progressions

For more VPP trending data see FD.io CSIT trending pages:
https://docs.fd.io/csit/master/trending/index.html

Link to graph: https://docs.fd.io/csit/master/trending/trending/l2-2n-clx-xxv710.html#b-l2switching-base-scale-avf

Benchmark Results: IPv4 Routing at Scale (64B NDR Throughput in Mpps)



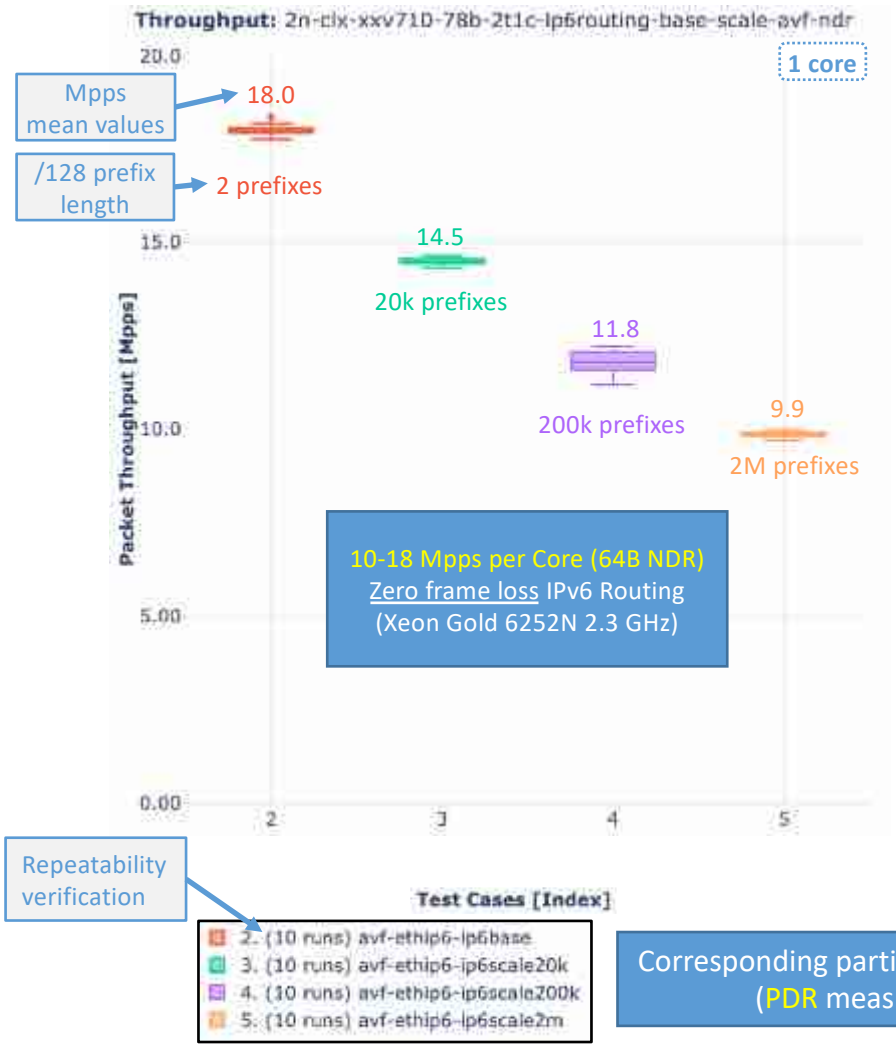


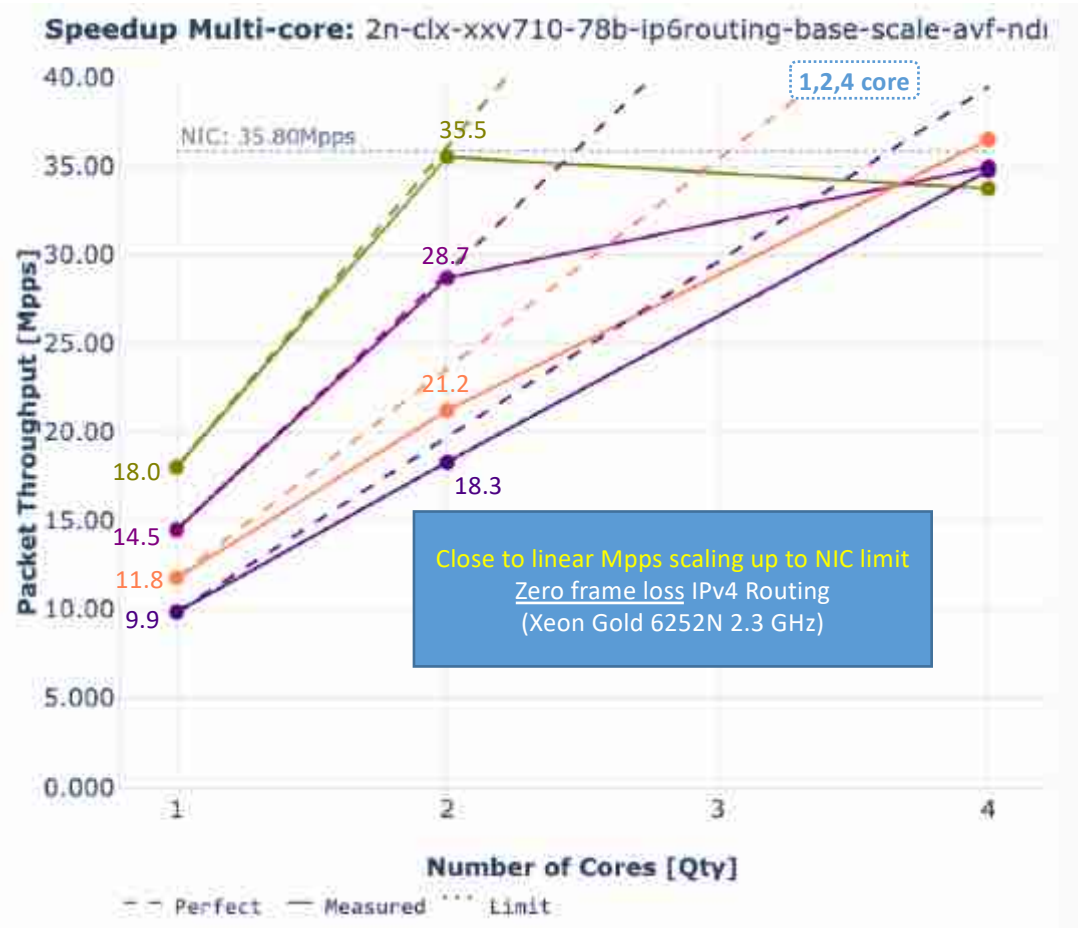
Corresponding partial-drop-rates (PDR) are close to NDR
(PDR measured at packet-loss-ratio = 0.5%)

CSIT-2005: https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/packet_throughput_graphs/ip4-2n-clx-xxv710.html
CSIT-2009: https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/packet_throughput_graphs/ip4-2n-clx-xxv710.html

CSIT-2005: https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/throughput_speedup_multi_core/ip4-2n-clx-xxv710.html
CSIT-2009: https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/throughput_speedup_multi_core/ip4-2n-clx-xxv710.html

Benchmark Results: IPv6 Routing at Scale (78B NDR Throughput in Mpps)





Corresponding partial-drop-rates (PDR) are very close to NDR
(PDR measured at packet-loss-ratio = 0.5%)

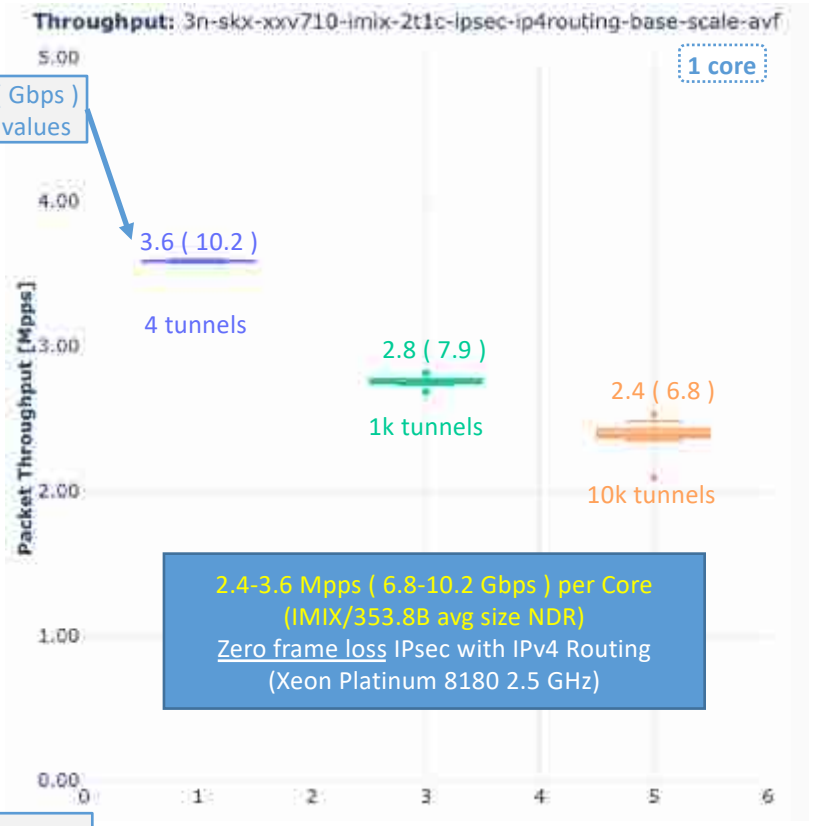
avf-ethip4-ip4base
avf-ethip4-ip4scale20k
avf-ethip4-ip4scale200k
avf-ethip4-ip4scale2m

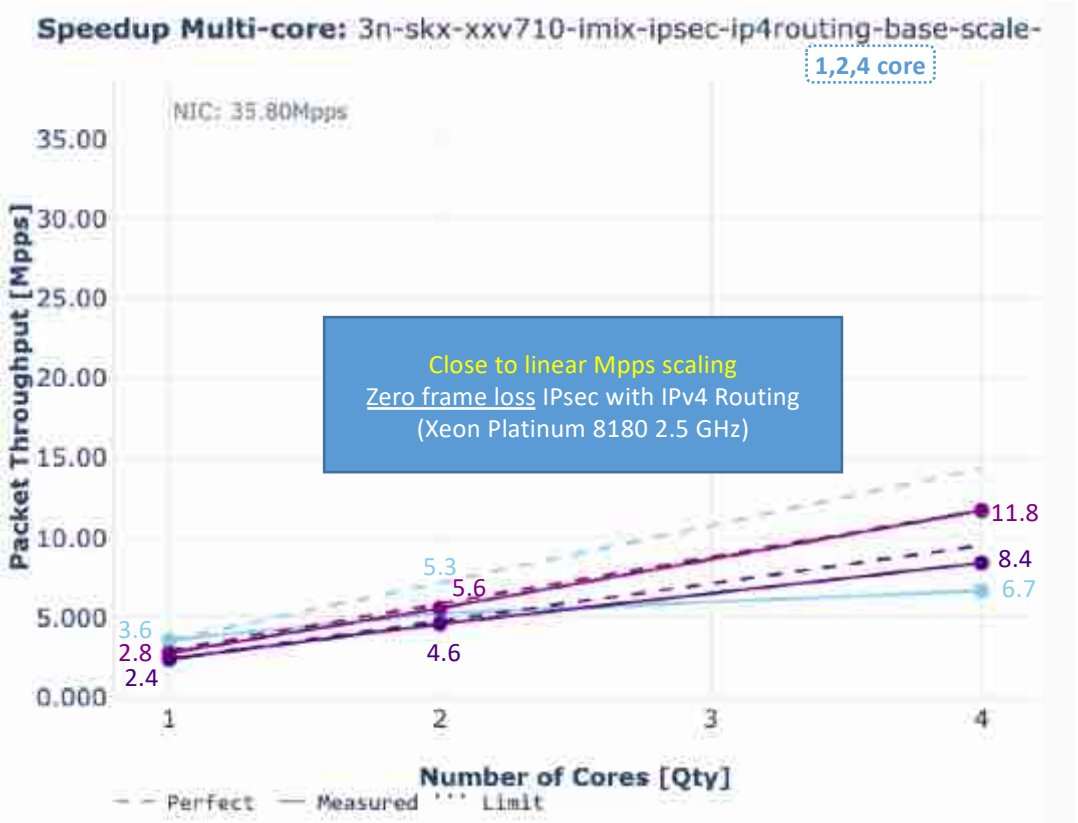
CSIT-2005: https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/packet_throughput_graphs/ip6-2n-clx-xxv710.html
CSIT-2009: https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/packet_throughput_graphs/ip6-2n-clx-xxv710.html

CSIT-2005: https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/throughput_speedup_multi_core/ip6-2n-clx-xxv710.html
CSIT-2009: https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/throughput_speedup_multi_core/ip6-2n-clx-xxv710.html

Benchmark Results: IPsec 4-10k tunnels (IMIX NDR Throughput in Mpps)

AES-NI instruction set, no hardware acceleration



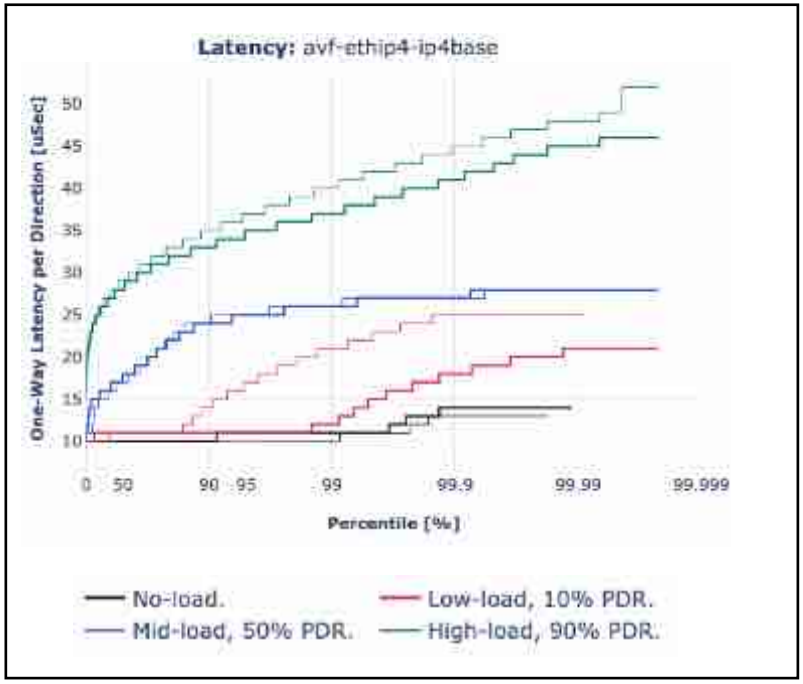


CSIT-2005: https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/packet_throughput_graphs/ipsec-3n-skx-xxv710.html
CSIT-2009: https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/packet_throughput_graphs/ipsec-3n-skx-xxv710.html

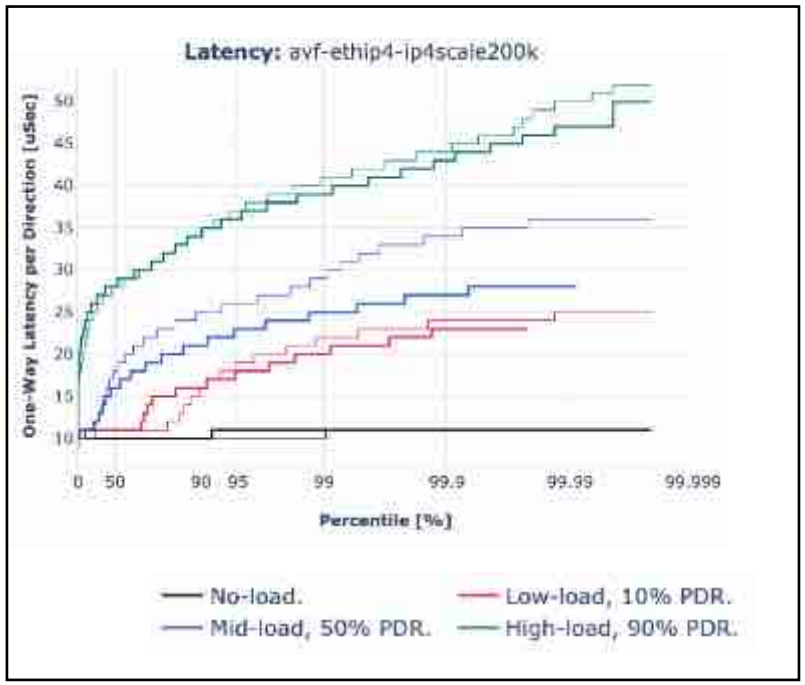
CSIT-2005: https://docs.fd.io/csit/rls2005/report/vpp_performance_tests/throughput_speedup_multi_core/ipsec-3n-skx-xxv710.html
CSIT-2009: https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/throughput_speedup_multi_core/ipsec-3n-skx-xxv710.html

Latency Test Methodology: TRex with HDRhistogram

IPv4 routing baseline with 2 /32 prefixes



IPv4 routing scale with 200k /32 prefixes



- One-way latency measurements with dedicated latency packet streams.
- Varying background packet load as % of PDR to verify impact on latency.
- HDRhistogram* integrated into TRex enables recording of a High Dynamic Range packet latency histogram.
- Zig-zagged lines artefact of TRex 1 usec latency measurement resolution (SW time-stamping today, future work to improve this).

* http://hdrhistogram.org/ , https://pypi.org/project/hdrhistogram/

CSIT Master: https://docs.fd.io/csit/rls2009/report/vpp_performance_tests/packet_throughput_graphs/ipsec-3n-skx-xxv710.html

CSIT System Design

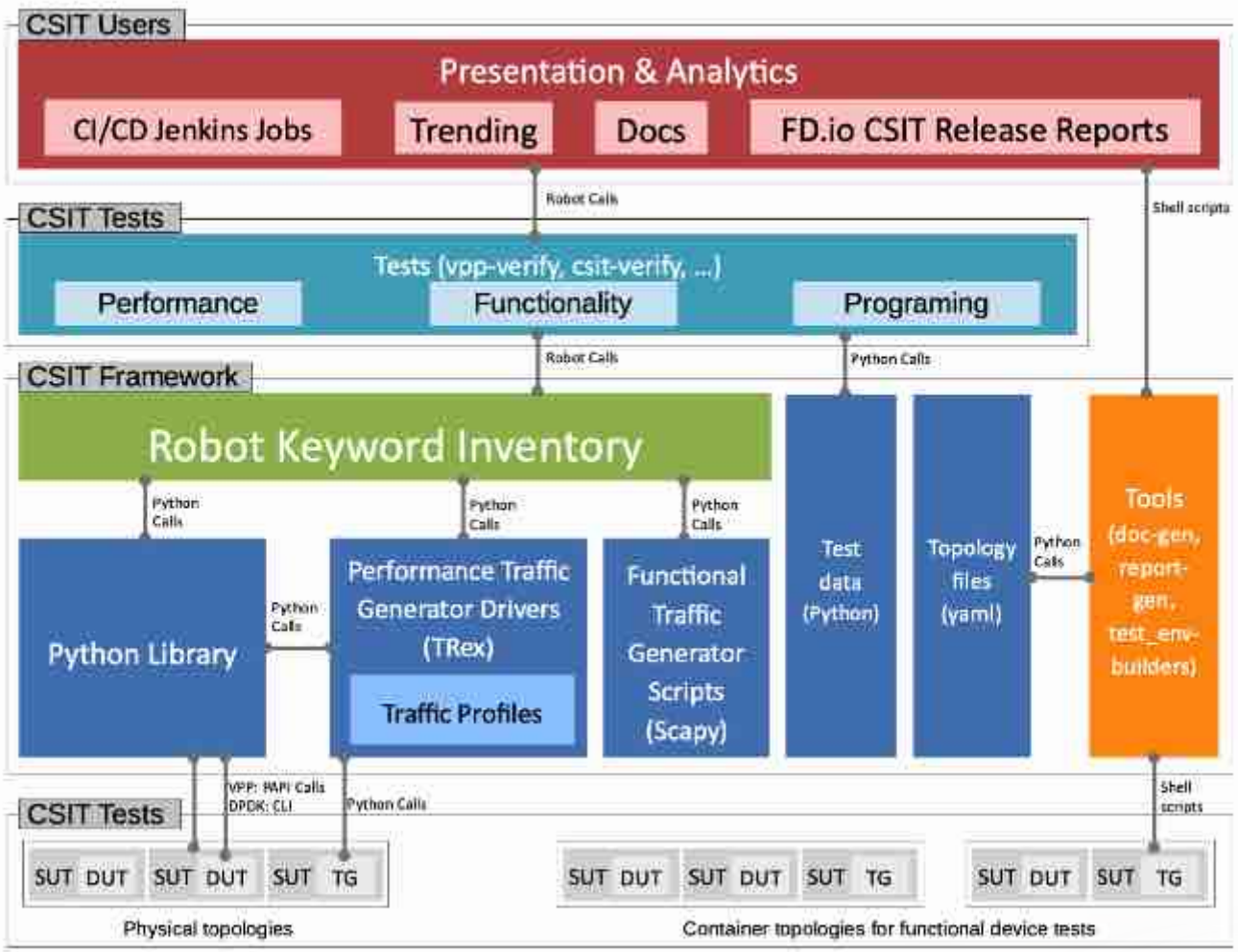
https://git.fd.io/csit/tree/

./docs
./resources/tools/presentation

./tests/vpp/perf
./tests/vpp/device

./resources/libraries

./topologies



https://docs.fd.io/csit/rls2009/report/csit_framework_documentation/csit_design.html

Main Work Areas

- TRex and TG tooling improvements
  - Increasing performance for stateful tests.
  - Improving precision of latency tests.
- Evolve test methodologies across the board
  - Stateless and stateful.
  - Further parameterization of traffic profiles.
- White-box performance analytics
  - Focused leverage of platform PMUs.
  - Per network function counter analytics.
- Benchmarks in the CSPs (AWS, Azure, GCP)
  - Onboarding and execution of benchmarks.
  - Analytics.
- Adding new and upcoming CPU platforms
  - Arm N1, Atoms, EPYCs and Xeons.

IETF BMWG Work

- Two drafts
  - MLRsearch, https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch
  - PLRsearch, https://tools.ietf.org/html/draft-vpolak-bmwg-plrsearch
- Markdown source files in FD.io CSIT repo
  - https://git.fd.io/csit/tree/docs/ietf

FD.io CSIT Resources

- Project:
  - Wiki pages: https://wiki.fd.io/view/CSIT
  - Mailing list: csit-dev@lists.fd.io
- CSIT Release Reports
  - Latest CSIT-2009: https://docs.fd.io/csit/rls2009/report/
  - Previous: https://docs.fd.io/csit/
- Benchmark Trending Pages
  - Daily and Weekly: https://docs.fd.io/csit/master/trending/index.html
- Source Code:
  - Main repo: https://git.fd.io/csit
  - Github mirror: https://github.com/FDio/csit
  - Gerrit reviews: https://gerrit.fd.io

And, of course, we are on continuous lookout for growing the community of contributors, users and other interested parties!

Born Ready for Secure Terabit Internet!
Tooling for Benchmarking.

"For better or worse, benchmarks shape a field."
– David A. Patterson, John L. Hennessy

Good ones accelerate progress.

THANK YOU !