High Performance Cloud-native Networking

K8s Unleashing FD.io

Maciek Konstantynowicz et al in FD.io CSIT
FD.io CSIT Project Lead, lf-id/iirc: mackonstan
Distinguished Engineer, Cisco, mkonstan@cisco.com

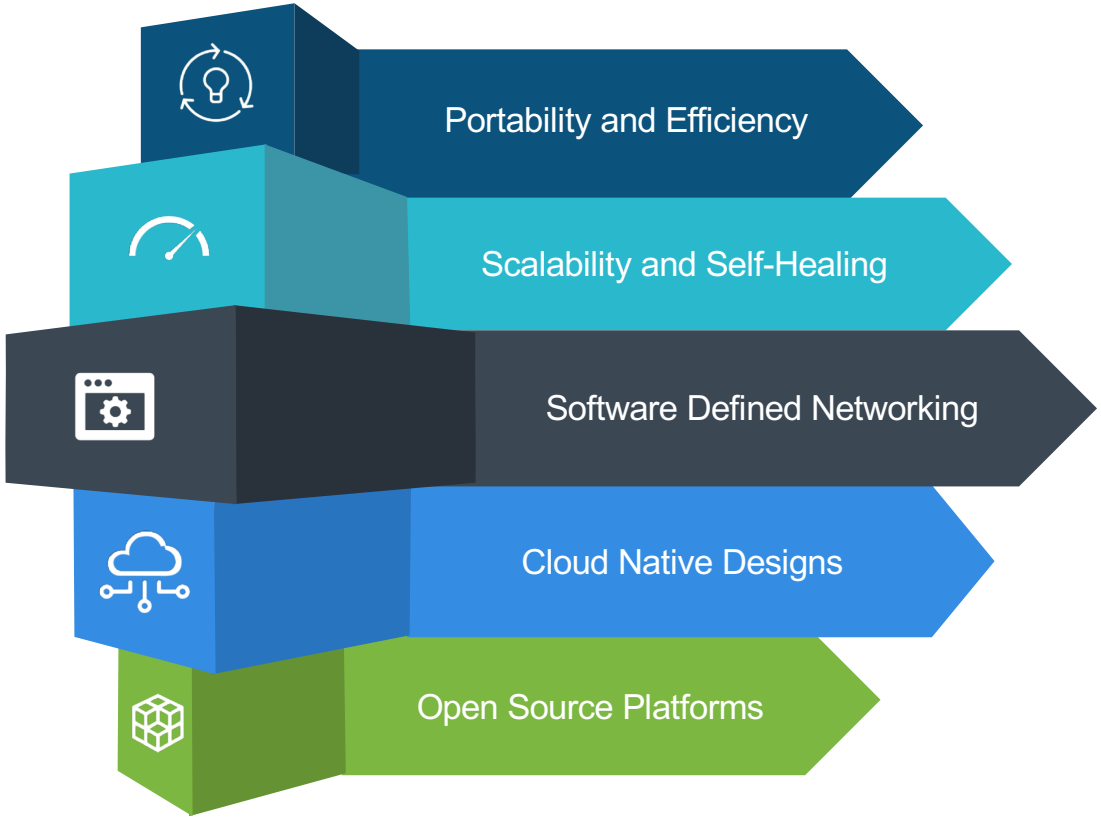
10-Dec-2018 FD.io mini-summit

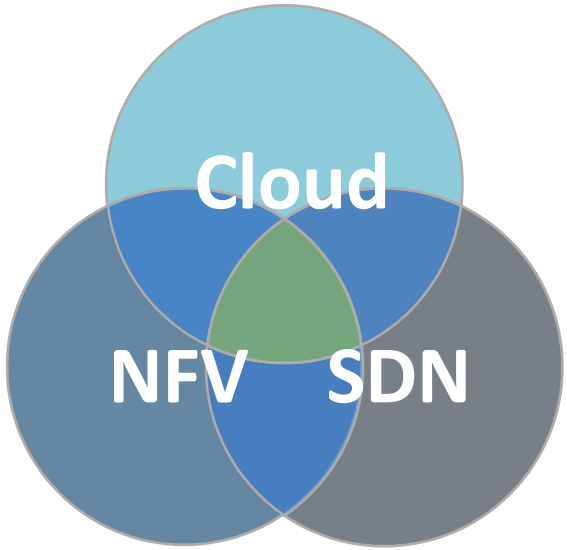
Kubecon Cloud-Native-Con North America, Seattle

DISCLAIMERs

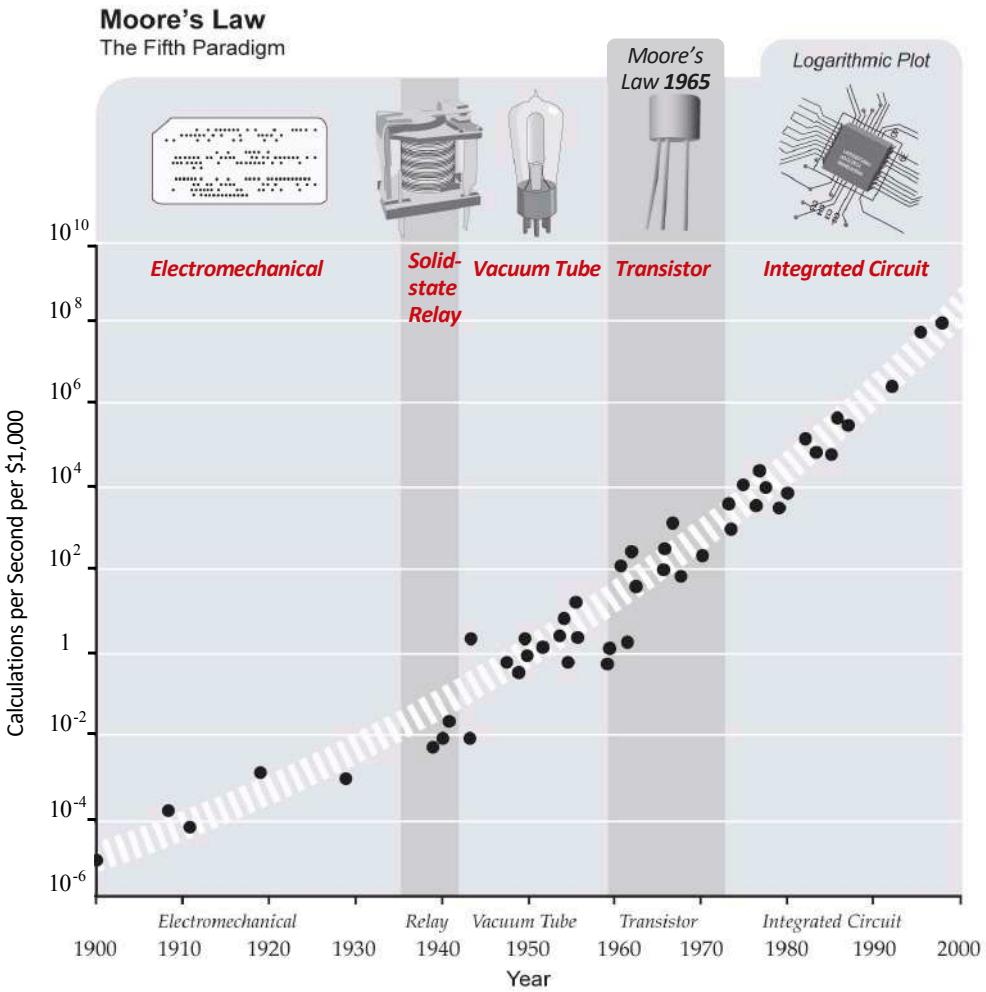
- **'Mileage May Vary'**
  - Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your opinion and investment of any resources. For more complete information about open source performance and benchmark results referred in this material, visit https://wiki.fd.io/view/CSIT and/or https://docs.fd.io/csit/rls1810/report/.
- **Trademarks and Branding**
  - This is an open-source material. Commercial names and brands may be claimed as the property of others.

Internet Mega Trends – ..



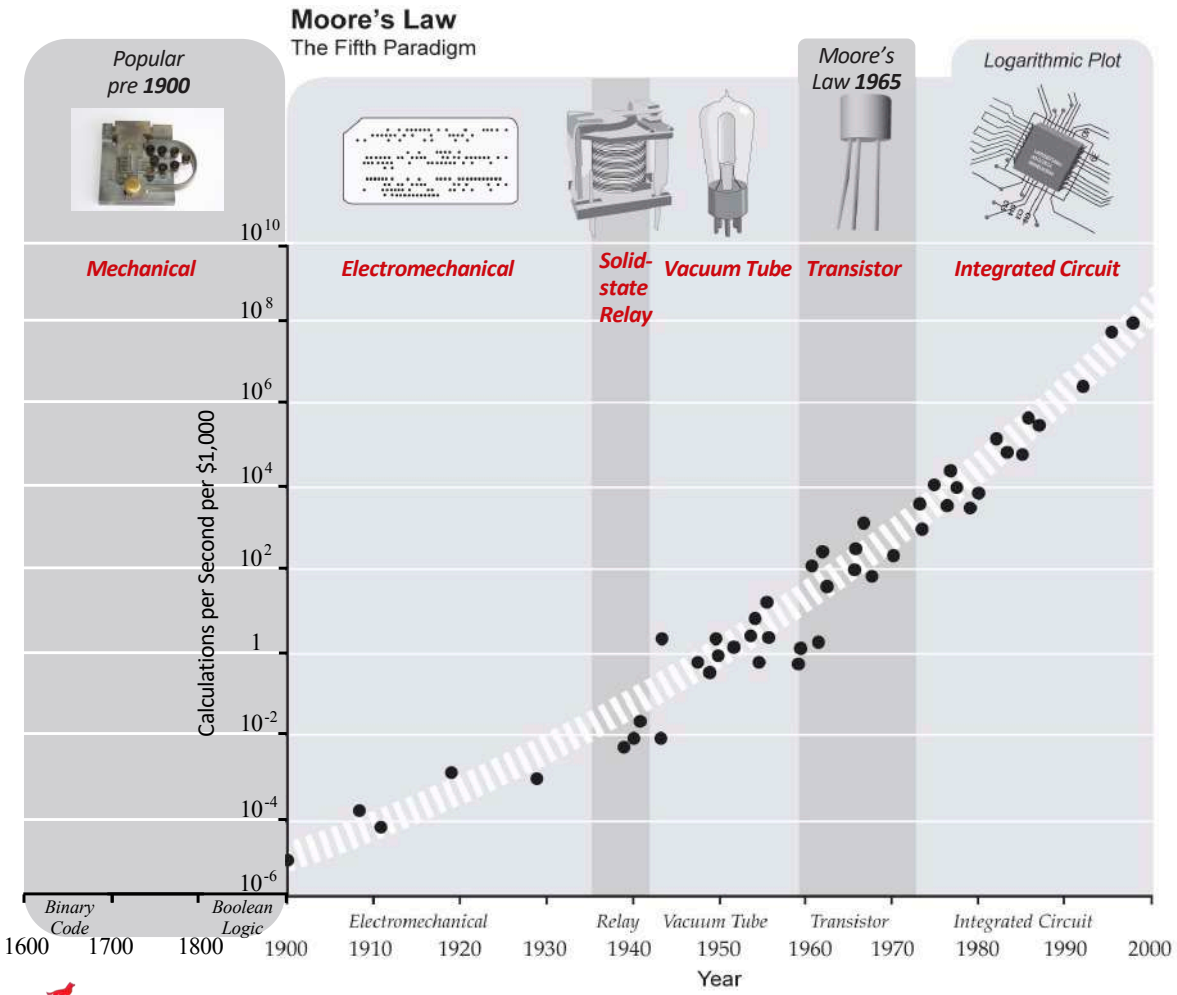


Remember **1965** ”Moore’s Law” – ..



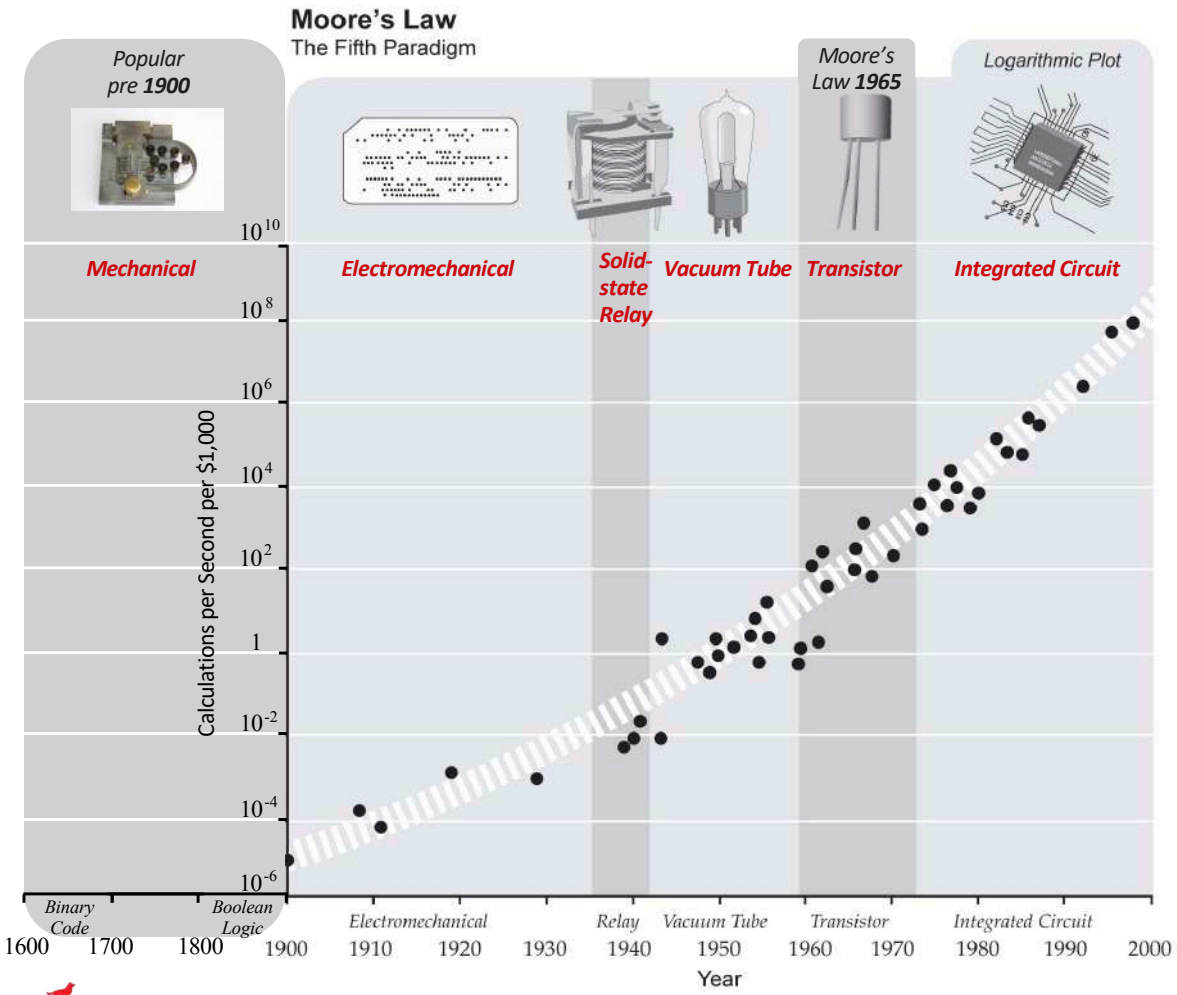
Source: Ray Kurzweil, "The Singularity Is Near: When Humans Transcend Biology", Page 67, The Duckworth Publishers 2009. Data points between 1600 and 1900, and after 2000 represent Presenter’s perspective and approximations.

Remember **1965** ”Moore’s Law” – ..



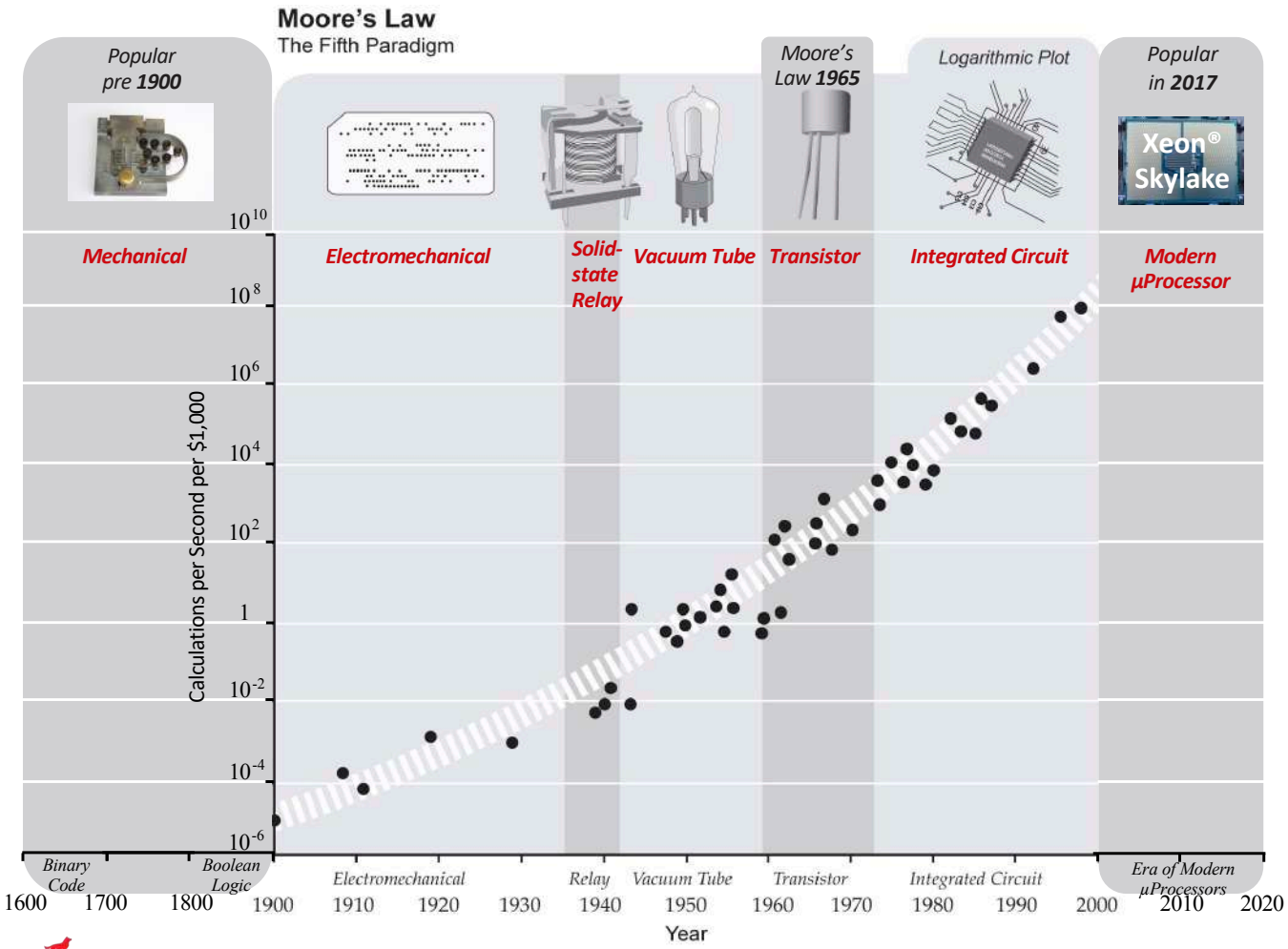
Source: Ray Kurzweil, "The Singularity Is Near: When Humans Transcend Biology", Page 67, The Duckworth Publishers 2009. Data points between 1600 and 1900, and after 2000 represent Presenter’s perspective and approximations.

Remember **1965** ”Moore’s Law” – 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.

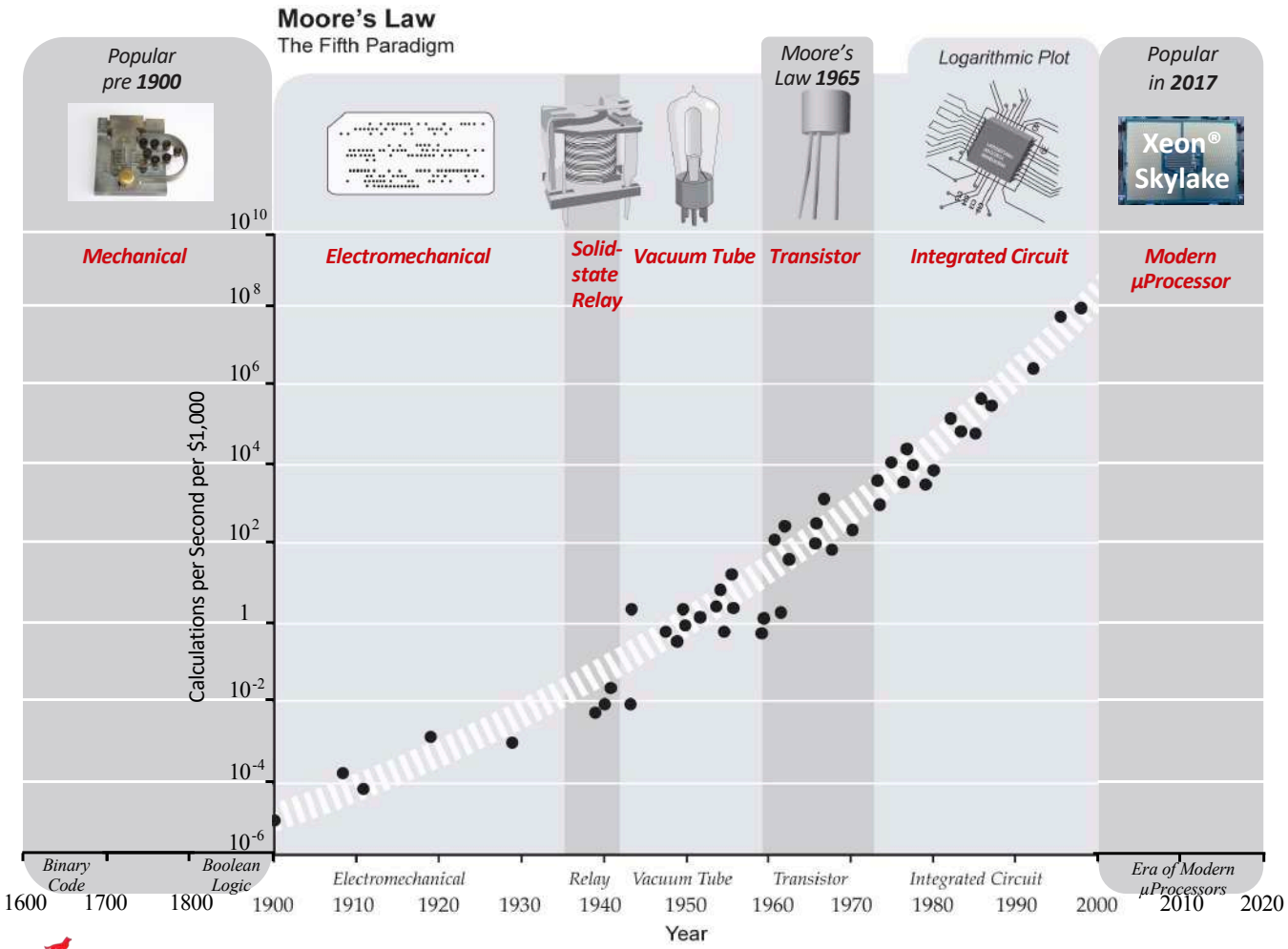
Remember **1965** ”Moore’s Law” – Does It Still Work?



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” – Yes, It Surely Does ..

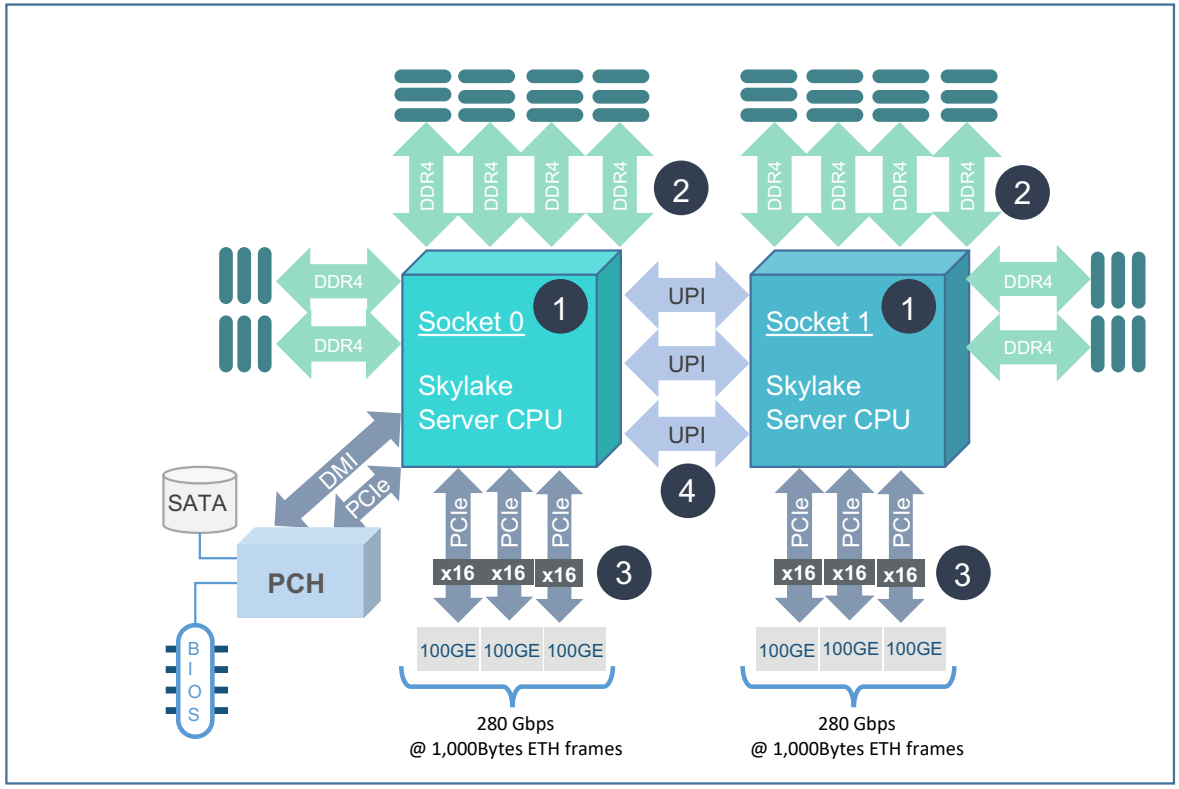
“Ramble On ..”



...

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.

Processing Packets: How to Use Compute ..



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

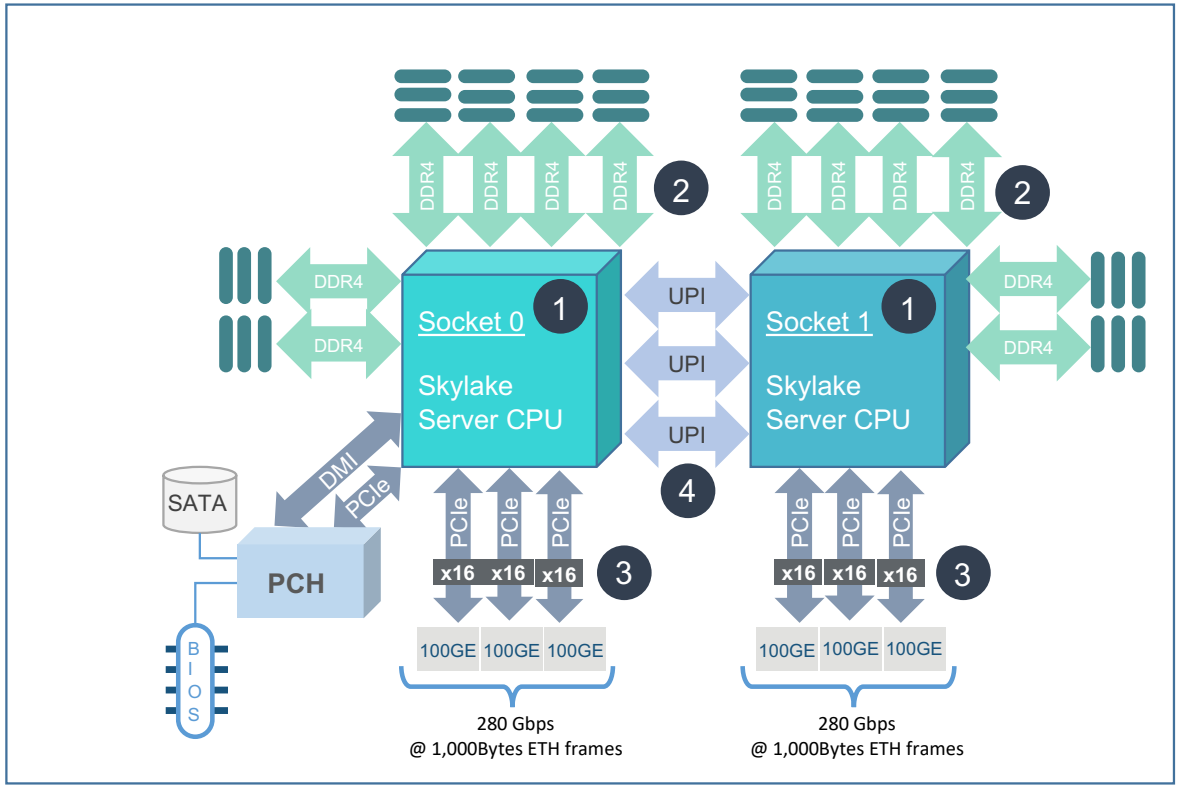
$$Throughput\ [pps] = \frac{1}{Packet\_Processing\_Time[sec]} = \frac{CPU\_freq[Hz]}{Cycles\_per\_Packet}$$

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

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

Processing Packets: What Improves in Compute ..



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

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

Moore's Law in Action

Generation to Generation – BDX to SKX example

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 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)

Open Source Benchmarking – Guiding Principles

- Discover the **limits** and **know** them
- Assess based on **externally measured data** and behaviour (black-box)
- Guide benchmarking by **good understanding** of the whole system (white-box)
- Provide a **feedback loop** to hardware and software engineering

"One should understand the laws of physics, and use them efficiently!"

“Without data, you're just another person with an opinion.” — W. Edwards Deming

FD.io CSIT-1810 Open-Source Benchmarking Report

VPP Performance:

1. **VPP release notes**
   a. Changes in CSIT-1810: http://bit.ly/2OxSUA6
   b. Known issues: http://bit.ly/2REvB9D

2. **VPP performance graphs**
   a. Throughput: http://bit.ly/2OwQPEn
   b. Speedup Multi-Core: http://bit.ly/2zztqgi
   c. Latency: http://bit.ly/2F9j4K6

3. **VPP performance comparisons**
   a. VPP-18.10 vs. VPP-18.04: http://bit.ly/2RGBvXM
   b. 3-Node Skylake vs. 3-Node Haswell testbeds: http://bit.ly/2DgjrOJ
   c. 2-Node Skylake vs. 3-Node Skylake testbeds: http://bit.ly/2FzC84q

4. **VPP performance test details**
   a. Detailed results: http://bit.ly/2yVWMGf
   b. Configuration: http://bit.ly/2qy7w8U
   c. Run-time telemetry: http://bit.ly/2PQlkKw

DPDK Testpmd and L3fwd Performance:

1. **DPDK release notes**
   a. Changes in CSIT-1810: http://bit.ly/2Oy7E1U
   b. Known issues: http://bit.ly/2RGGE22

2. **DPDK Testpmd and L3fwd performance graphs**
   a. Throughput: http://bit.ly/2PeSlAz
   b. Latency: http://bit.ly/2z5OSdx

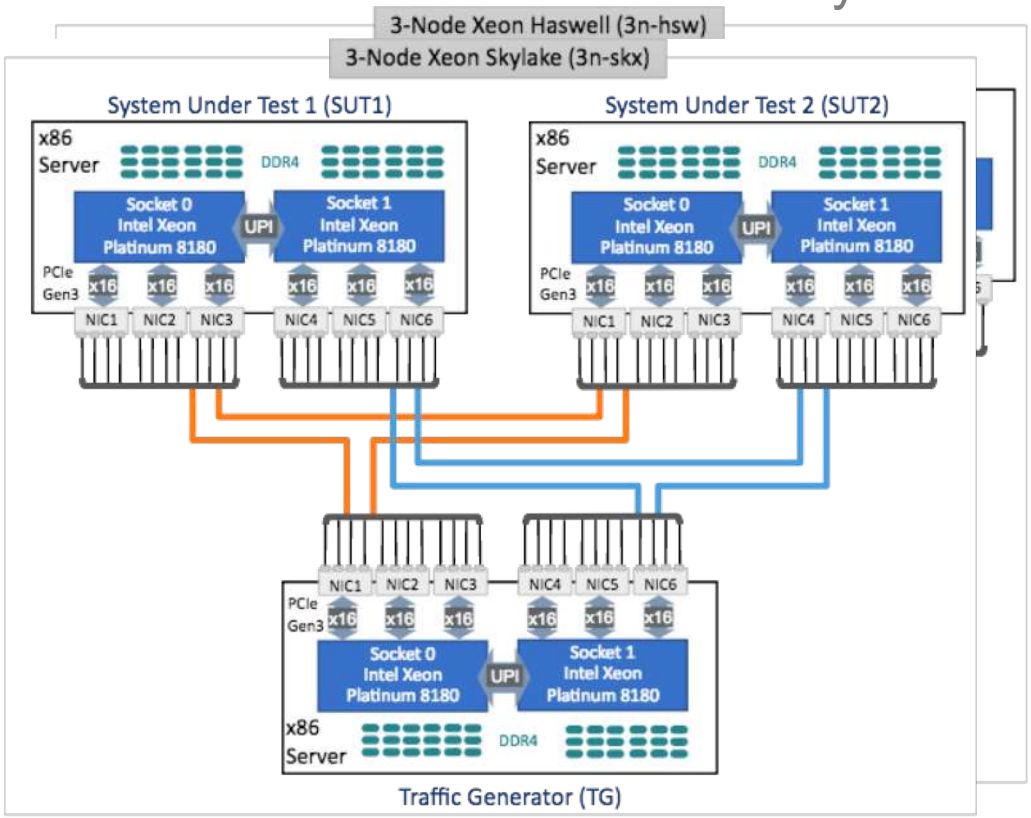
3. **DPDK performance comparisons**
   a. DPDK-18.08 vs. DPDK-18.05: http://bit.ly/2Q6QVrp

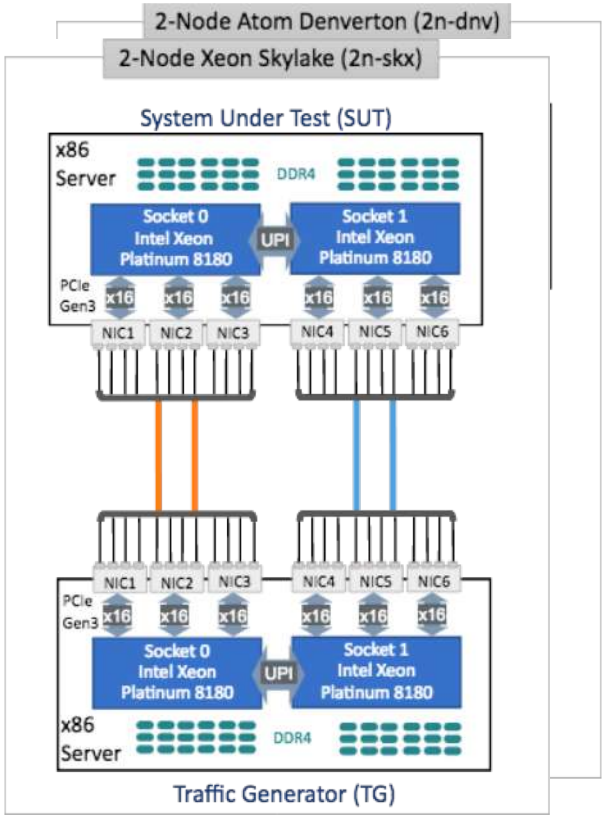
FD.io CSIT-1810.49 report has been published on FD.io docs site:

html: https://docs.fd.io/csit/rls1810/report/
pdf: https://docs.fd.io/csit/rls1810/report/_static/archive/csit_rls1810.pdf

FD.io CSIT-1810 Open-Source Benchmarking Report

Performance Test Environments: Physical Testbeds





SUT1 and SUT2 servers are populated:

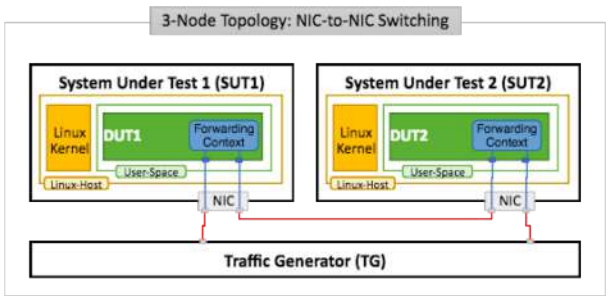
- NIC-1: x710-DA4 4p10GE Intel.
- NIC-2: xxv710-DA2 2p25GE Intel.

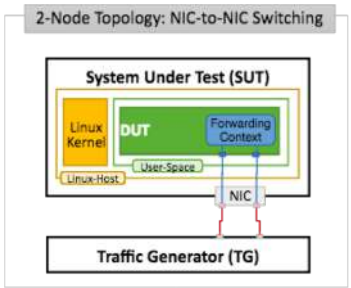
SUT servers are populated with:

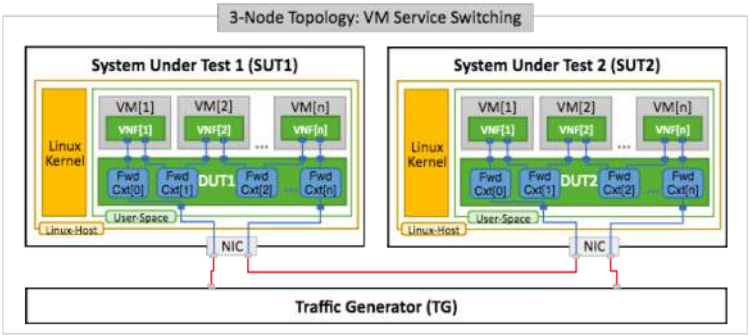
- NIC-1: x710-DA4 4p10GE Intel.
- NIC-2: xxv710-DA2 2p25GE Intel.
- NIC-3: mcx556a-edat ConnectX5 2p100GE Mellanox. (Not used yet.)

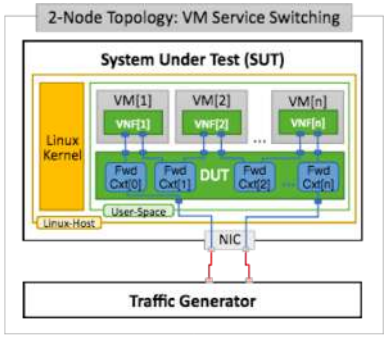
FD.io CSIT-1810 Open-Source Benchmarking Report

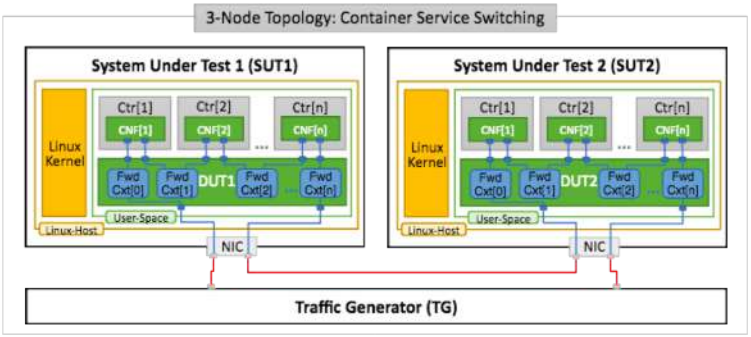
Performance Test Environments: Logical Topologies

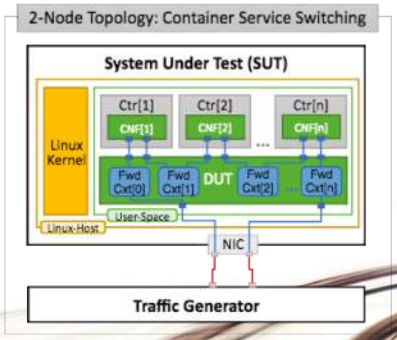










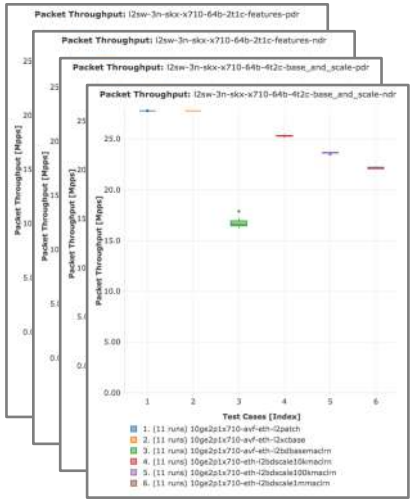


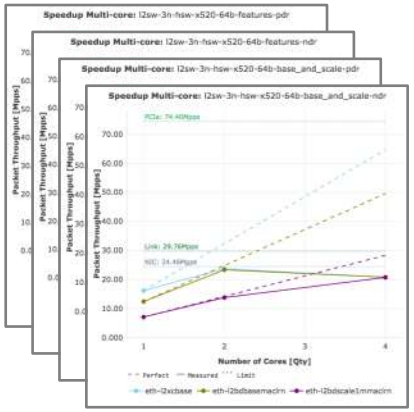
Benchmarking Data and Public References

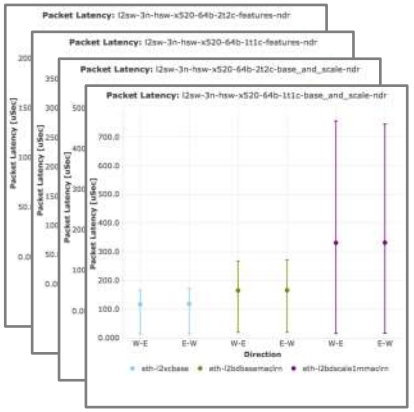
FD.io CSIT

Per release test and performance reports

- **Multi-Platform/-Vendor**
  - Intel Xeon and Atom, Arm (WIP)
- **Packet Throughput & Latency**
  - Zero-Drop & Partial Drop Rates
- **Data Plane Workloads**
  - FD.io VPP
  - DPDK L3fwd, Testpmd
- **Scaling**
  - Single-, Multi-Core
  - MACs, IPs, Flows, ACLs etc.
  - K8s
- **Performance Test Suites (unique #s)**
  - L2: 106
  - L3 (IPv4 / IPv6): 136
  - VM vhostuser: 56
  - Containers memif: 12
  - Crypto: 26
  - SRv6: 12
  - TCP/IP: 02
  - **Total: 350**







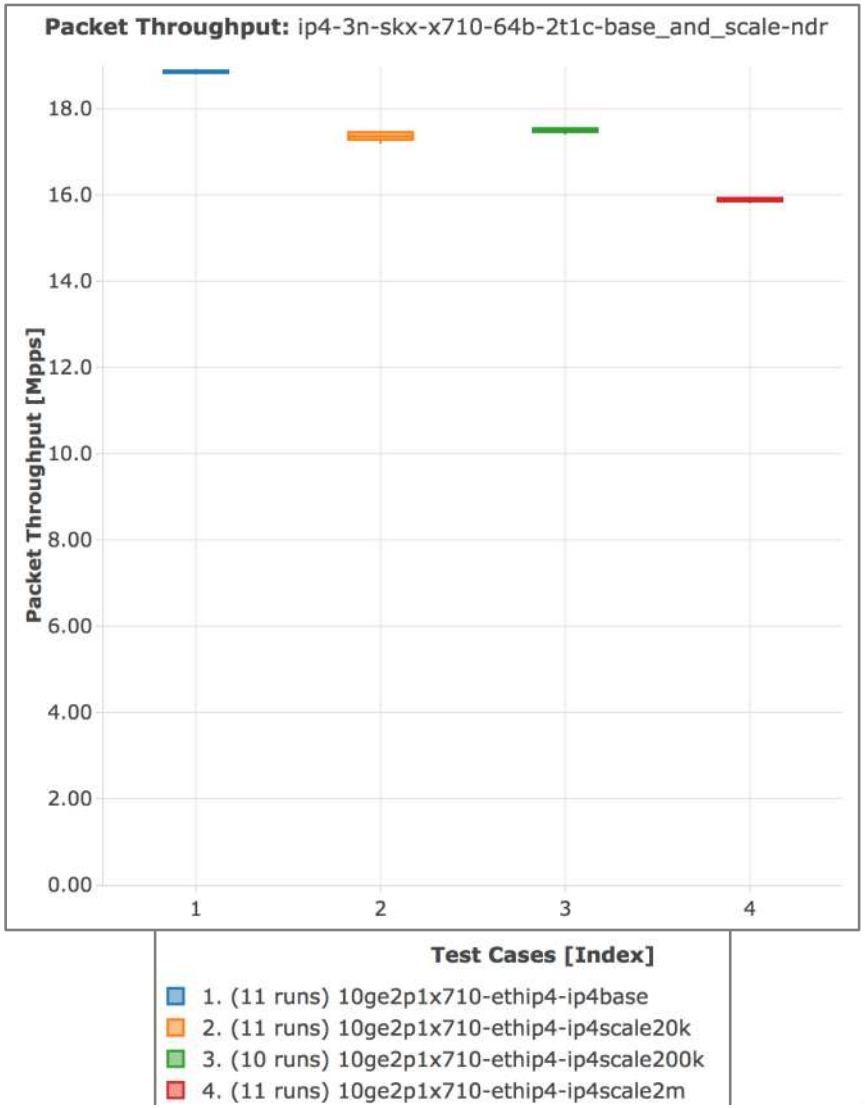
https://docs.fd.io/csit/rls1810/report/

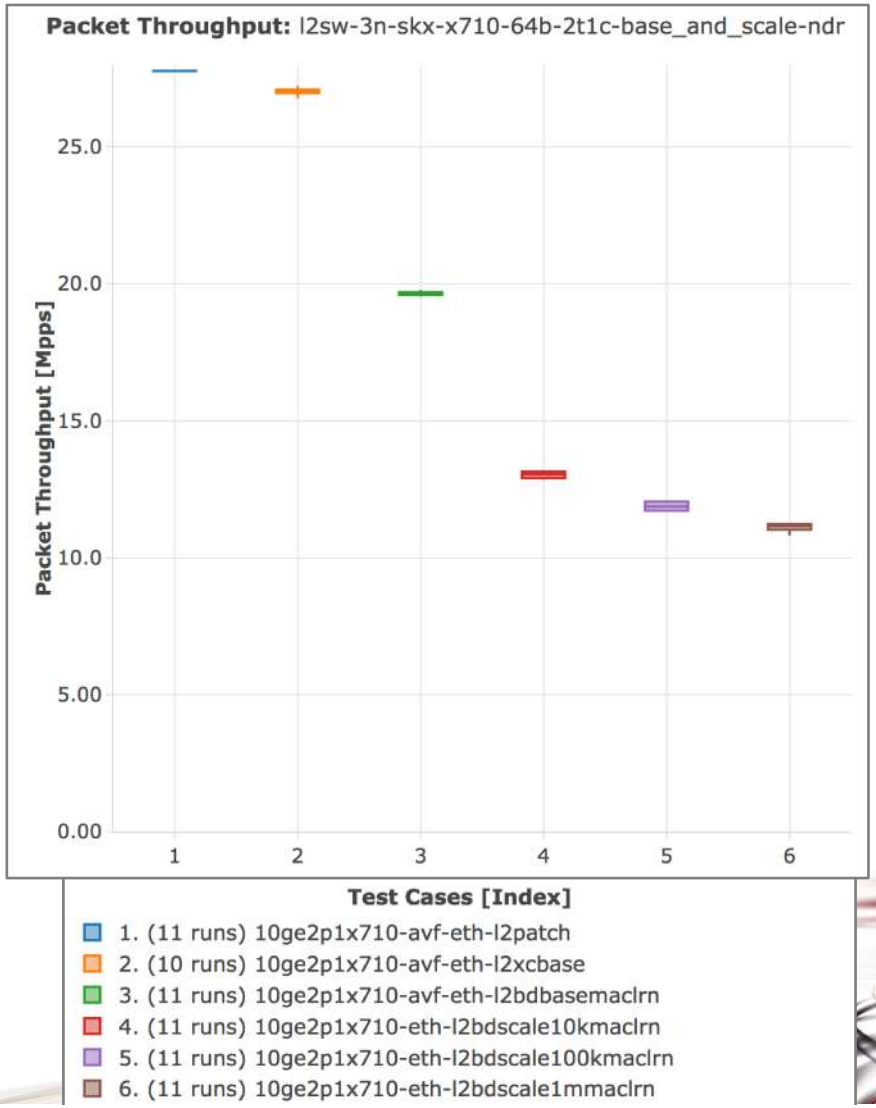
CSIT-1810 report includes Ever More benchmarking data, more graphs (**756** on the last count) that are now more orderly and more beautifully presented than Ever. Report HTML and PDF layouts got revamped more than a bit, striving to improve presentation clarity and browsability. And equally portability and consumability got improved. We are aiming the FD.io CSIT Open-Source Benchmarking reports to be consumable by everyone. And by everyone we mean Everyone. Those who code it, those who test it, those who use it, those who want to use it but are afraid to touch it. Everyone who likes it.

Breath (# of test cases), **Depth** (of measurement) and **Repeatability** (every release, repeatable locally)

Source: https://docs.fd.io/csit/rls1810/report/

FD.io CSIT-1810: Packet Throughput Results





FD.io CSIT-1810: Throughput Speedup Results

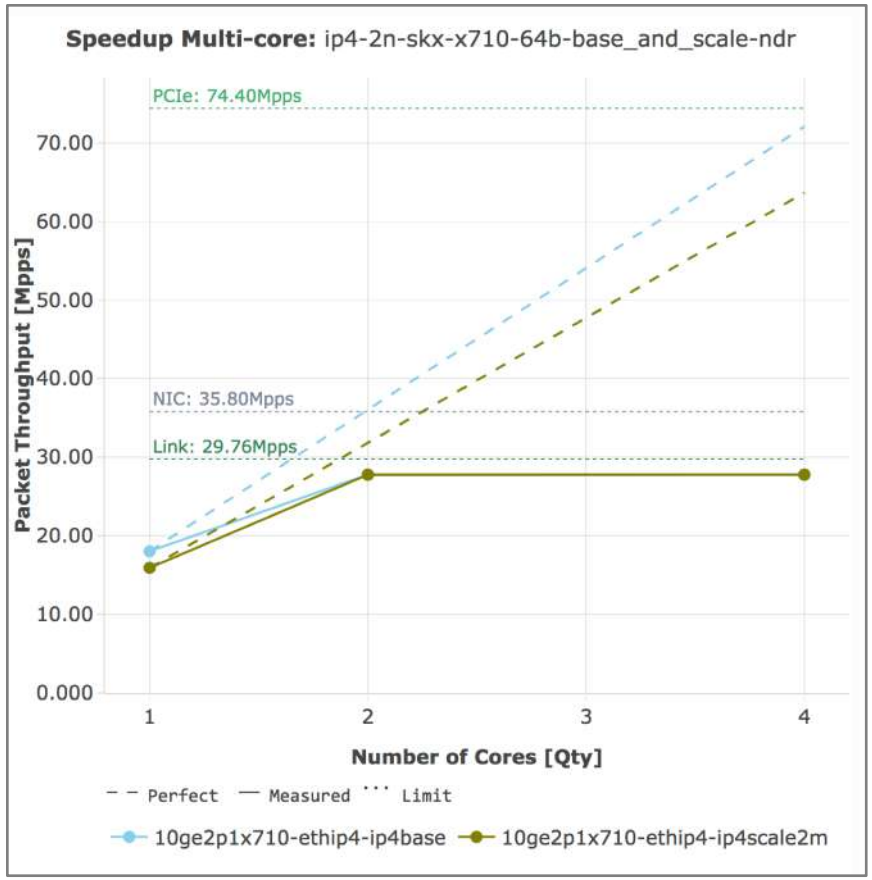
Source: https://docs.fd.io/csit/rls1810/report/

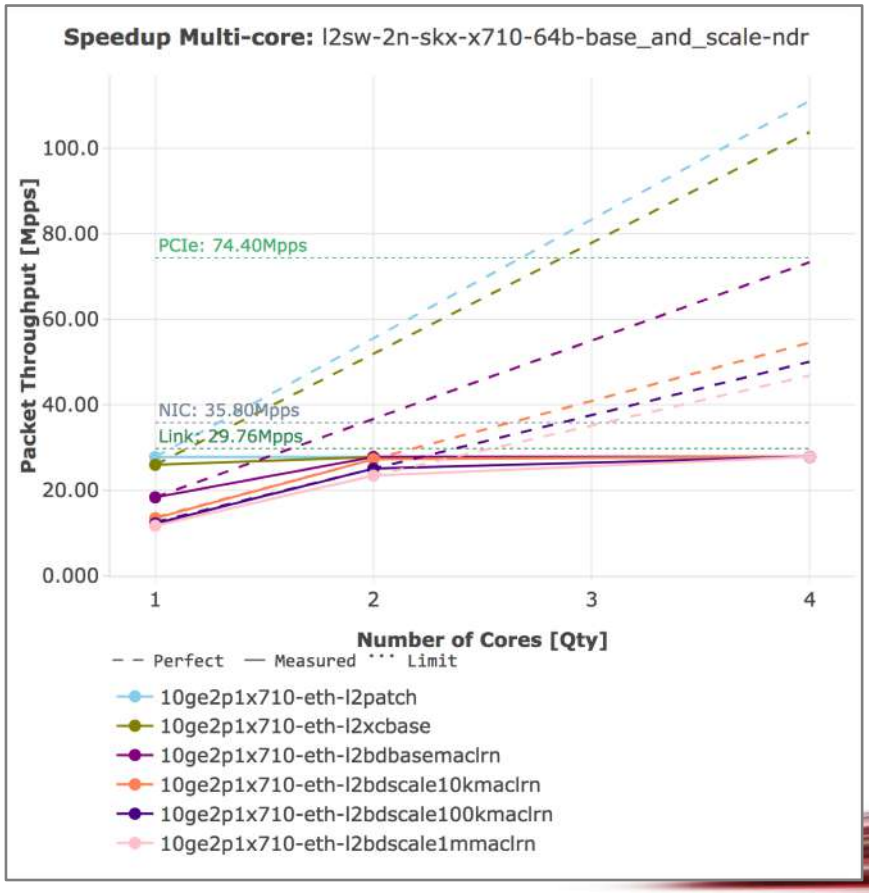
FD.io VPP Multi-Core Speedup Properties:

A. Predictable performance

B. Linear scaling with cores

C. Follows Amdahl's Law





https://docs.fd.io/csit/rls1810/report/_static/vpp/performance-compare-testbeds-3n-hsw-3n-skx-mrr.txt

Test case	3-Node Hsw Receive Rate [Mpps]	3-Node Hsw Stdev [Mpps]	3-Node Skx Receive Rate [Mpps]	3-Node Skx Stdev [Mpps]	Delta [%]
10ge2p1x710-64b-1c-ethip4-ip4scale2m	8.51	0.0	15.47	0.02	81
10ge2p1x710-78b-1c-ethip6-ip6scale2m	4.99	0.0	8.62	0.05	72
10ge2p1x710-64b-1c-ethip4udp-ip4base-iacl1sf-10kflows	5.85	0.0	9.78	0.02	67
10ge2p1x710-64b-2c-eth-l2bdscale1mmaclrn	13.56	0.0	22.7	0.07	67
10ge2p1x710-64b-1c-ethip4udp-ip4base-iacl50sf-10kflows	5.85	0.0	9.8	0.05	67
10ge2p1x710-64b-1c-eth-l2bdscale1mmaclrn	6.92	0.0	11.38	0.04	64
10ge2p1x710-64b-1c-eth-l2bdbasemaclrn-iacl50sf-10kflows	5.07	0.0	8.34	0.02	64
10ge2p1x710-64b-1c-ethip4udp-ip4scale1000-udpsrcscale15-nat44	6.61	0.0	10.78	0.03	63
10ge2p1x710-64b-1c-eth-l2xcbase	15.99	0.0	26.19	0.01	63
10ge2p1x710-64b-2c-ethip4udp-ip4base-iacl50sf-10kflows	11.73	0.0	19.2	0.07	63
10ge2p1x710-64b-2c-ethip4udp-ip4base-iacl1sf-10kflows	11.77	0.0	19.17	0.1	62
10ge2p1x710-64b-1c-eth-l2xcbase-eth-1memif-1dcr	10.85	0.0	17.36	0.02	60
10ge2p1x710-64b-2c-eth-l2bdbasemaclrn-iacl50sf-10kflows	10.11	0.0	16.25	0.09	60
10ge2p1x710-78b-1c-ethip6-ip6scale20k	7.8	0.0	12.48	0.03	60
10ge2p1x710-64b-2c-eth-l2bdbasemaclrn-iacl1sf-10kflows	10.12	0.0	16.25	0.26	60
10ge2p1x710-64b-1c-eth-l2bdbasemaclrn-iacl1sf-10kflows	5.05	0.0	8.13	0.18	60
10ge2p1x710-64b-1c-eth-l2bdscale100kmaclrn	7.52	0.0	12.1	0.03	60
10ge2p1x710-64b-2c-eth-l2bdscale100kmaclrn	15.05	0.0	24.1	0.1	60
10ge2p1x710-78b-1c-ethip6-ip6scale200k	6.58	0.0	10.51	0.06	59
10ge2p1x710-64b-2c-ethip4udp-ip4scale1000-udpsrcscale15-nat44	12.85	0.0	20.46	0.09	59
10ge2p1x710-64b-1c-ethip4-ip4scale20k	11.22	0.0	17.88	0.02	59
10ge2p1x710-78b-2c-ethip6-ip6scale20k	15.67	0.0	24.76	0.04	58
10ge2p1x710-64b-1c-dot1q-l2xcbase	10.54	0.0	16.44	0.02	55
10ge2p1x710-64b-1c-ethip4udp-ip4base-iacl50sl-10kflows	5.76	0.0	8.95	0.09	55
10ge2p1x710-64b-1c-eth-l2bdbasemaclrn	12.4	0.0	19.34	0.02	55

...

FD.io CSIT-1901: NFV Service Density Measurements

Work resulting from collaboration with CNCF CI team

Problem

Clear performance metrics and standards are missing, despite SDN and NFV services being on the rise for few years now, and with many questions unanswered: how many (qty) and how much ($) NFV services can be supported on compute node(s) ?

How much compute resources is consumed (e.g. cores), how do VM and Container technologies compare, VNF and CNF ?

Applicability

Applies to all user verticals either already operating or moving to SDN NFV network designs.

Developers, CD/CI Integrators, Network Engineers, Designers, Decision Makers.

Solution

Benchmark NFV Service Densities using **Virtual Machine** and **Container technologies**, single compute node measurements for now.

Leveraged a collective cross-team experience of FD.io CSIT and CNCF dev and test teams.

WARNING WHAT FOLLOWS IS VERY FRESH OFF THE PRESS
PLEASE PROCEED WITH CAUTION !

IT IS FOCUSING ON TECHNOLOGY BENCHMARKING AND COMPARISON
NOT A PRODUCT – TREAT IT AS SOFT LAUNCH ;)

NFV Technologies: VNF and CNF

Virtualized Network Function

VNF

- Virtual machine based
- Larger compute footprint
- Slower to start, stop, change
- Hard to move around

Cloudified Network Function

CNF

- Linux container based
- Smaller compute footprint
- Faster to start, stop, change
- Easy to move around

Q1: How to compare NF technologies like VNF and CNF?
A: Standalone not doing any work. Eh?
A: Inter-connected doing packet processing work. Yes!

Q2: How to measure NF compute footprint?
A: Processing packets. Yes!
A: Forwarding packets over inter-connects. Yes!

Q3: How to benchmark NFV services based on any NF technologies and any HW and SW stack?
A: Inter-connect few of them *somehow* into *some* useful topology, and measure some work. Eh?
A: Define an NF inter-connection model and benchmark NF service density using it. Yes!

NFV Technologies: VNF and CNF

Virtualized Network Function

VNF

Virtual machine based
Larger compute footprint
Slower to start, stop, change
Hard to move around

Cloudified Network Function

CNF

Linux container based
Smaller compute footprint
Faster to start, stop, change
Easy to move around

Q1: How to compare NF technologies like VNF and CNF?
~~**A**: Standalone not doing any work. Eh?~~
A: Inter-connected doing packet processing work. Yes!

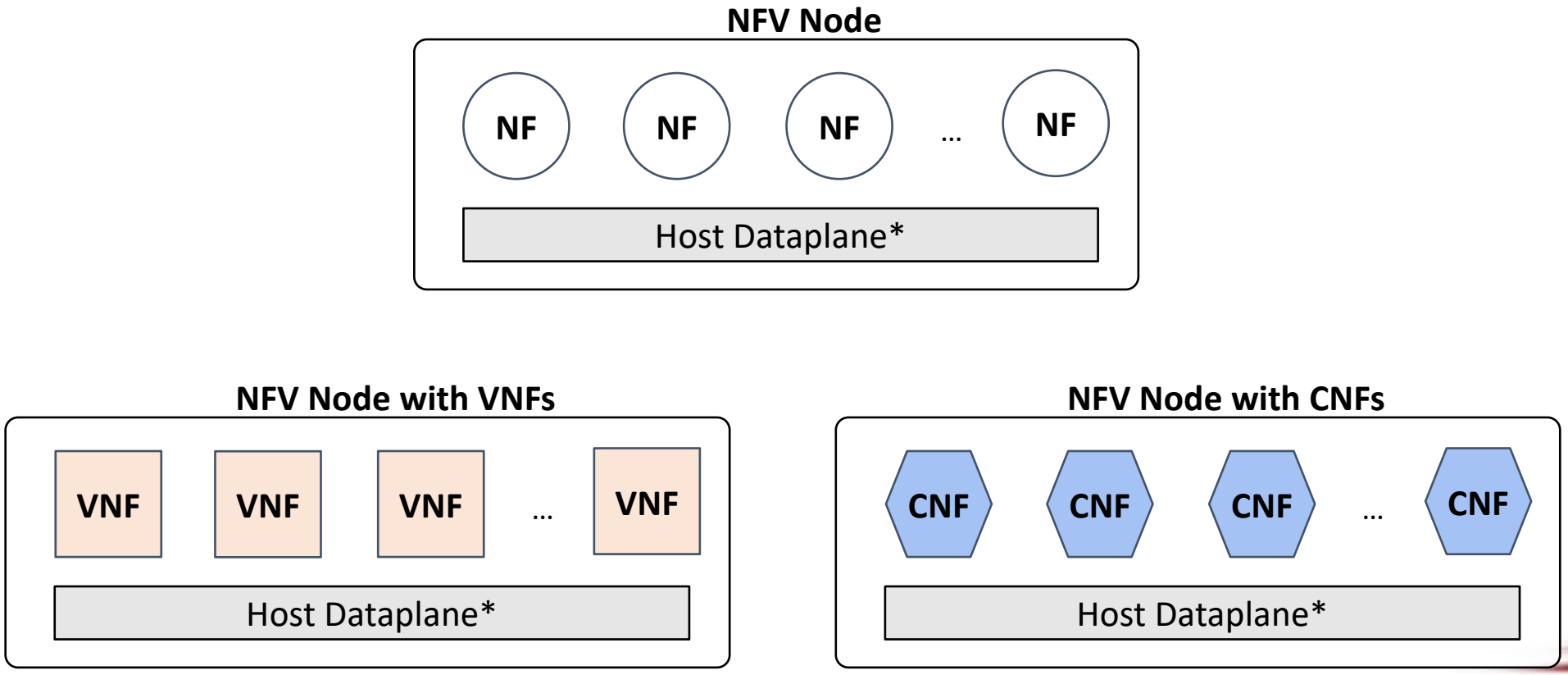
Q2: How to measure NF compute footprint?
A: Processing packets. Yes!
A: Forwarding packets over inter-connects. Yes!

Q3: How to benchmark NFV services based on any NF technologies and any HW and SW stack?
~~**A**: Inter-connect few of them *somehow* into *some* useful topology, and measure some work. Eh?~~
A: Define an NF inter-connection model and benchmark NF service density using it. Yes!

NFV Service Density Matrix is a particular methodology of addressing Q1, Q2 and Q3.
It is a result of collaboration across open source projects: **FD.io CSIT**, **CNCF/CNFs** and **OPNFV/NFVbench**.

NFV Service Design – with VNFs and CNFs

Single Compute Node

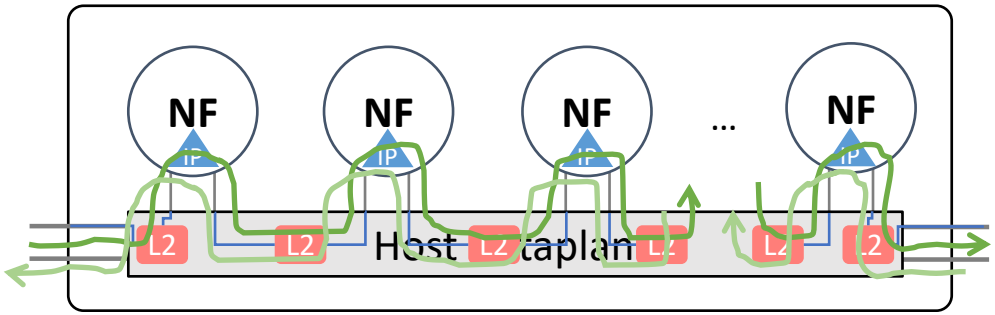


* Linux User-Mode SW Switch, Linux Kernel-Mode Networking, Direct Virtual Function, ..

NFV Service Abstraction

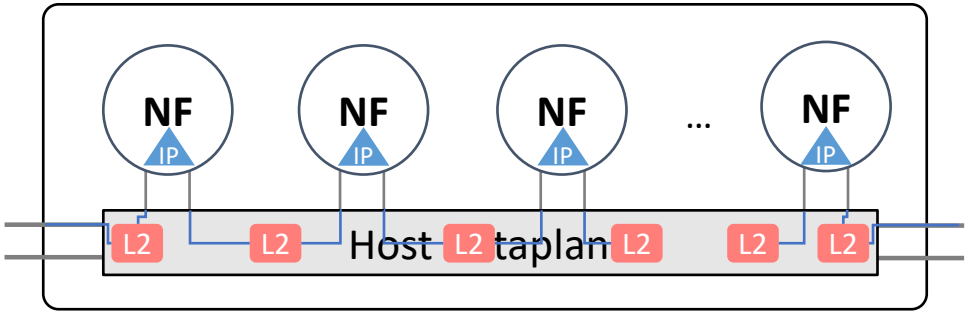
Service Forwarding Graph

How are the network functions and network devices forwarding packets



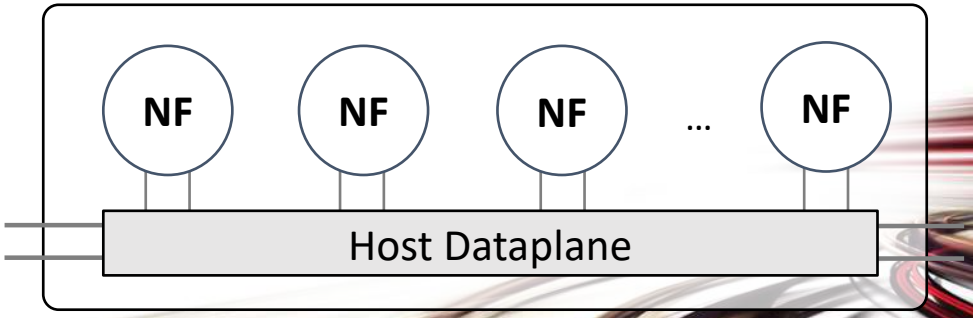
Service Topology

How are the network functions and network devices configured



Network Topology

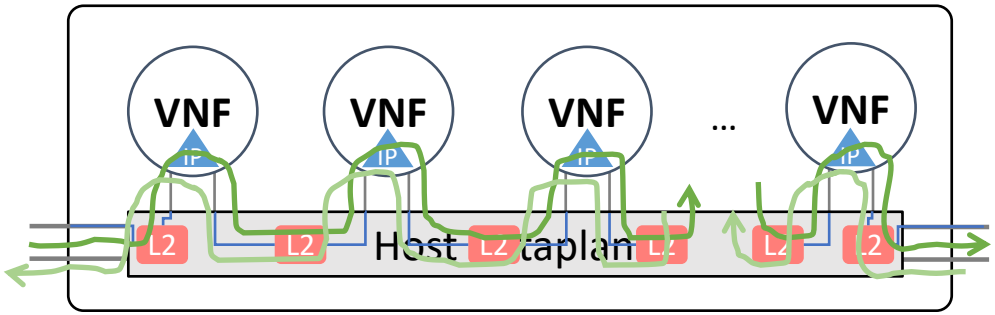
How are the network functions and network devices interconnected



Service Topologies, Defined and Testing So Far..

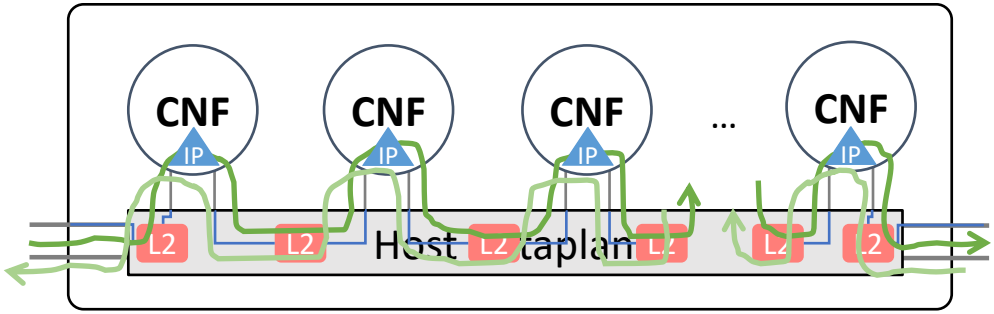
Service Topology — **VNF Service Chain (VSC)**

Forwarding Graph — Snake Forwarding



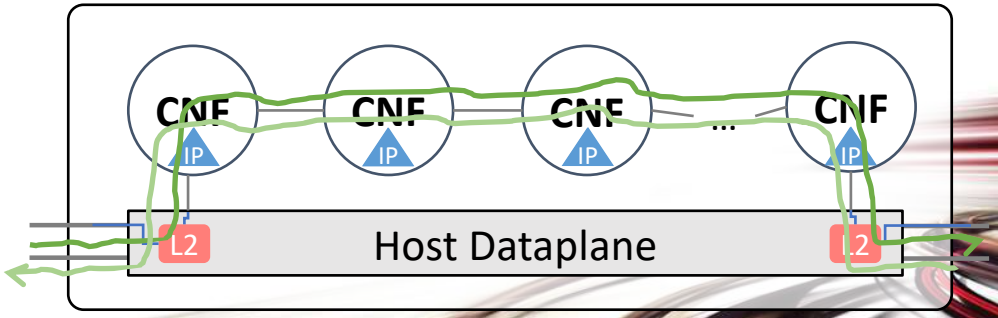
Service Topology — **CNF Service Chain (CSC)**

Forwarding Graph — Snake Forwarding



Service Topology — **CNF Service Pipeline (CSP)**

Forwarding Graph — Pipeline Forwarding



Service Density Matrix: Network Function View

More is Better

Row: **1..10** Network Service Instances.
Column: **1..10** Network Functions per Service Instance.
Value: **1..100** Network Functions.

SVC	001	002	004	006	008	010
001	1	2	4	6	8	10
002	2	4	8	12	16	20
004	4	8	16	24	32	40
006	6	12	24	36	48	60
008	8	16	32	48	64	80
010	10	20	40	60	80	100

Source: https://github.com/cncf/cnfs/blob/master/comparison/doc/cncf-cnfs-results-summary.md

Per NF Physical Core Resource Allocation

Methodology

Allocation of processor physical cores per NF instance is as follows:

1. **Two mapping ratios are defined and used in NF service matrix benchmarking:**
   a. `pcdr4nf` value determines **P**hysical **C**ore to **D**ataplane **R**atio **for NF.**
   b. `pcmr4nf` value determines **P**hysical **C**ore to **M**ain **R**atio **for NF.**

2. **Target values to be benchmarked:**
   a. pcdr4nf = [(1:1),(1:2),(1:4)].
   b. pcmr4nf = [(1:2),(1:4),(1:8)].

3. **Number of physical cores required for the benchmarked NFs' service matrix is calculated as follows:**

```
*     #pc = pcdr4nf * #dnf + pcmr4nf * #mnf
* where
*     #pc  - total number of physical cores required and used.
*     #dnf - total number of NF dataplane thread sets (1 set per NF instance).
*     #mnf - total number of NF main thread sets (1 set per per NF instance).
```

Source: https://github.com/cncf/cnfs/blob/master/comparison/doc/cncf-cnfs-results-summary.md

Service Density Matrix: Core Usage View

Less is Better!

```
Row:    1..10  Network Service Instances.
Column: 1..10  Network Functions per Service Instance.
Value:  1..NN Processor Physical Cores Used.
pcdr4nf = (1:1), pcmr4nf = (1:2)
```

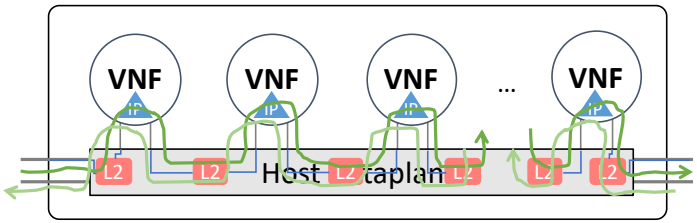
SVC	001	002	004	006	008	010
001	2	3	6	9	12	15
002	3	6	12	18	24	30
004	6	12	24	36	48	60
006	9	18	36	54	72	90
008	12	24	48	72	96	120
010	15	30	60	90	120	150

Source: https://github.com/cncf/cnfs/blob/master/comparison/doc/cncf-cnfs-results-summary.md

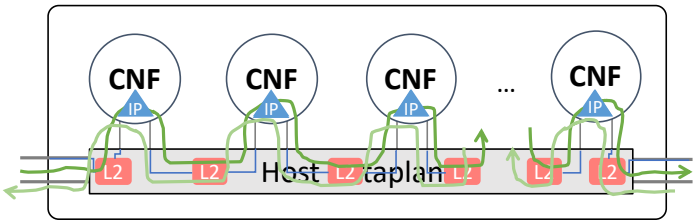
Service Density Matrix: MRR Throughput Results

More is Better!

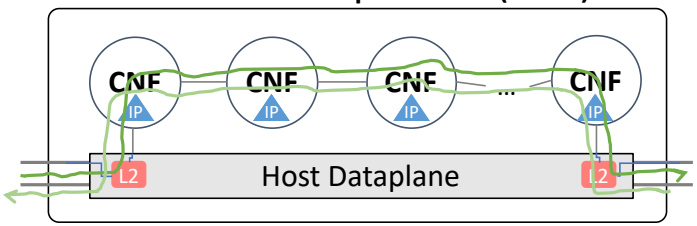
VNF Service Chains (VSC)

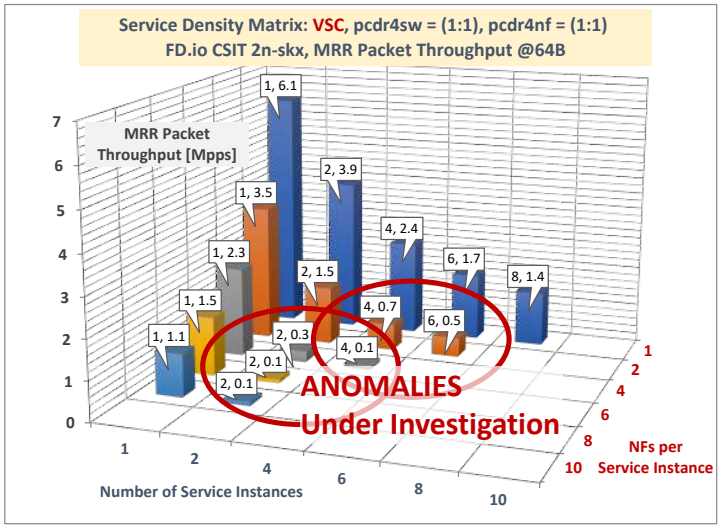


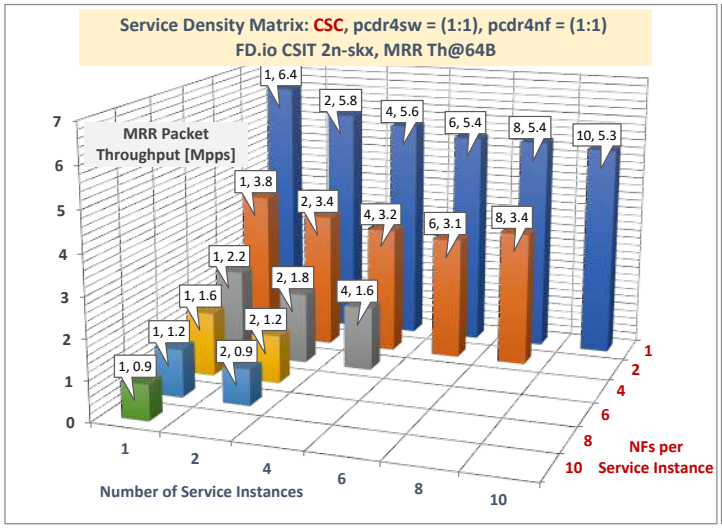
CNF Service Chains (CSC)

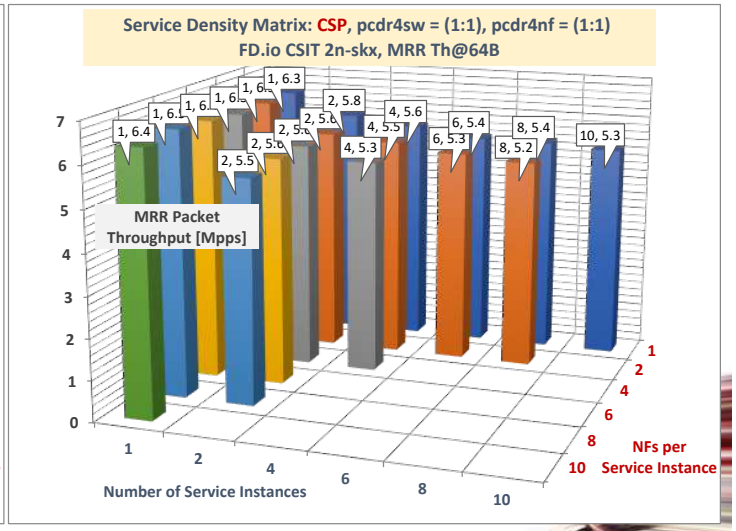


CNF Service Pipelines (CSP)









Source: https://github.com/cncf/cnfs/blob/master/comparison/doc/cncf-cnfs-results-summary.md

Service Density Matrix: What's Next..

Work in progress:

- Service Density Matrix in FD.io CSIT-1901
  - VSC, CSC, CSP [1..10] x [1..10] – continuous tests MRR, MLRsearch, PLRsearch
  - Xeon, Arm, Atom Platforms
- Continue collaboration with NSM team
- Continue collaboration with CNCF CI team

In work queue:

- Once "ready for service" add **N**etwork **S**ervice **M**esh orchestration
- Add promising new virtualization and containerization technologies
  - "Micro" VMs: AWS Firecracker, https://firecracker-microvm.github.io/
  - More secure and faster containers: Singularity Containers, https://github.com/sylabs/singularity

networkservicemesh.io

AFTER A SOFT LAUNCH COMES THE HARD LAUNCH ;)

Benchmarking standardization: IETF, ETSI

Done or work in progress:

- Two IETF drafts in bmwg
  - MLRsearch, https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch-00
  - PLRsearch, https://tools.ietf.org/html/draft-vpolak-bmwg-plrsearch-00
  - Aiming to update RFC2544, RFC1242
  - markdown source files in FD.io CSIT repo, https://git.fd.io/csit/tree/docs/ietf
- ETSI NFV
  - NFV vswitch benchmarking
  - FD.io CSIT contributions to the last version of technical report ver. TST009_v0015

In work queue:

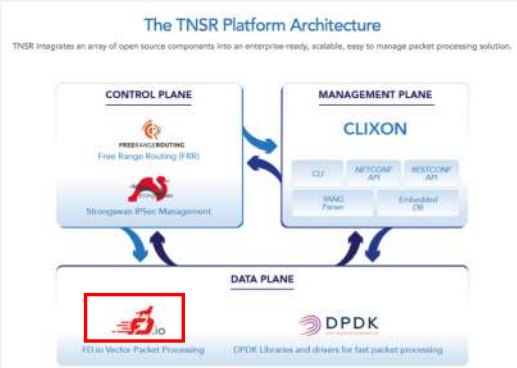
- One IETF draft for bmwg
  - NFV service density benchmarking, https://tools.ietf.org/html/draft-mkonstan(-bmwg)-nf-density-bench-00*
  - Aiming to update RFC2544, RFC1242

* Link not active, .md document will show up first in git.fd.io/csit/tree/docs/ietf.

Packet Vectors are Good for You !

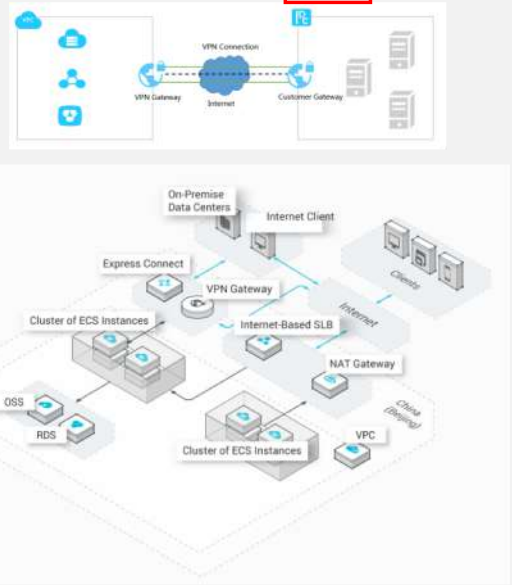
Netgate shipping product(s) [1]

Netgate: TNSR, hw appliances



Alibaba [2]

Alibaba: Network Service Optimization with the VPP Platform



...

[1] https://www.netgate.com/products/tnsr/
[2] https://www.alibabacloud.com/blog/network-service-optimization-with-the-vpp-platform_593985

Source: Slide courtesy of Jerome Tollet, FD.io, Cisco

5 Pillars of Modern Software Data Planes

Blazingly Fast

- Process the *massive explosion* of East-West traffic
- Process *increasing* North-South traffic

Software First

- Cloud means *running everywhere*
- Cloud means hardware and physical *infra agnostic*

Truly Extensible

- Foster *pace of innovation* in cloud-native networking
- *No compromise* on performance (zero-tolerance)

Predictable Performance

- Dataplane performance must be deterministic
- Predictable for a number of VMs, Containers, virtual topology and (E-W, N-S) traffic matrix

Measureable

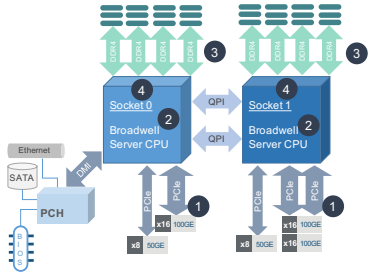
- Counters everywhere to *count everything* for detailed cross-layer operation and efficiency monitoring
- Enables feedback loop to drive engineering optimizations

FD.io VPP meets these challenges

Baremetal Data Plane Performance Limit

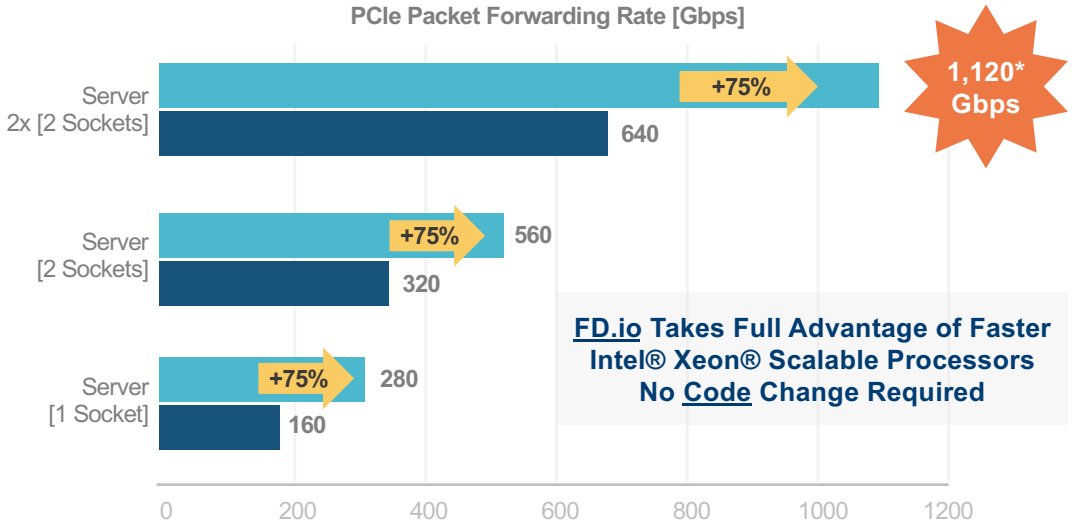
FD.io benefits from increased Processor I/O

YESTERDAY



1. Network I/O: **160 Gbps**
2. Core ALU: **4-wide** parallel µops
3. Memory: **4-channels** 2400 MHz
4. Max power: **145W** (TDP)

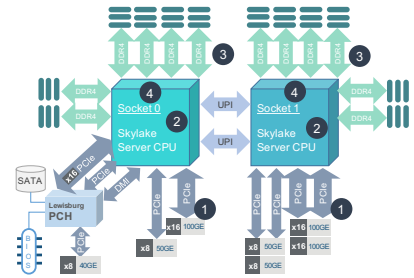
Intel® Xeon® E5-2699v4
22 Cores, 2.2 GHz, 55MB Cache



FD.io Takes Full Advantage of Faster Intel® Xeon® Scalable Processors
No Code Change Required

* On compute platforms with all PCIe lanes from the Processors routed to PCIe slots.

TODAY



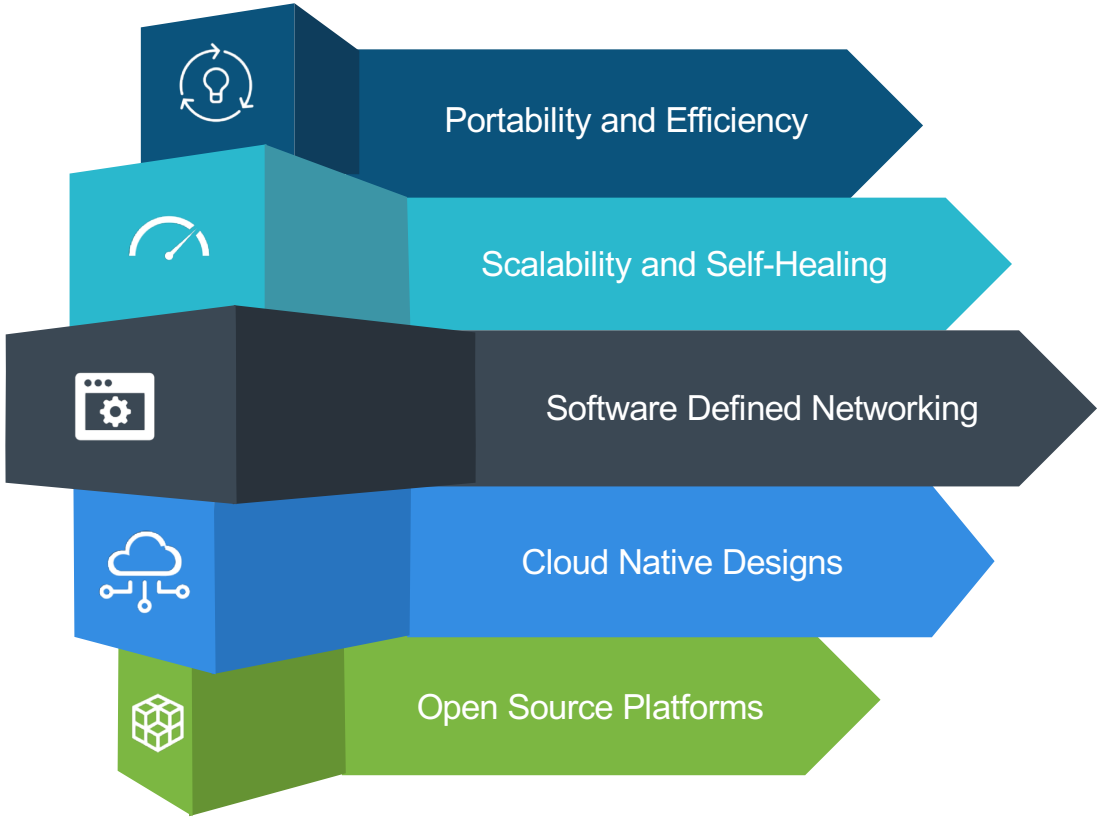
1. Network I/O: **280 Gbps**
2. Core ALU: **5-wide** parallel µops
3. Memory: **6-channels** 2666 MHz
4. Max power: **205W** (TDP)

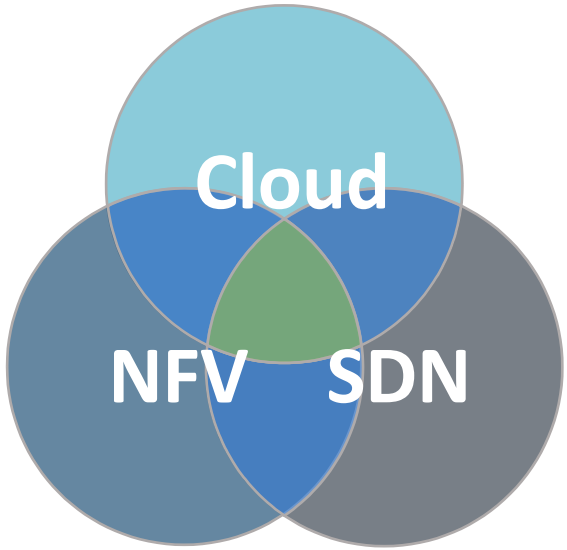
Intel® Xeon® Platinum 8168
24 Cores, 2.7 GHz, 33MB Cache

https://goo.gl/UtbaHy

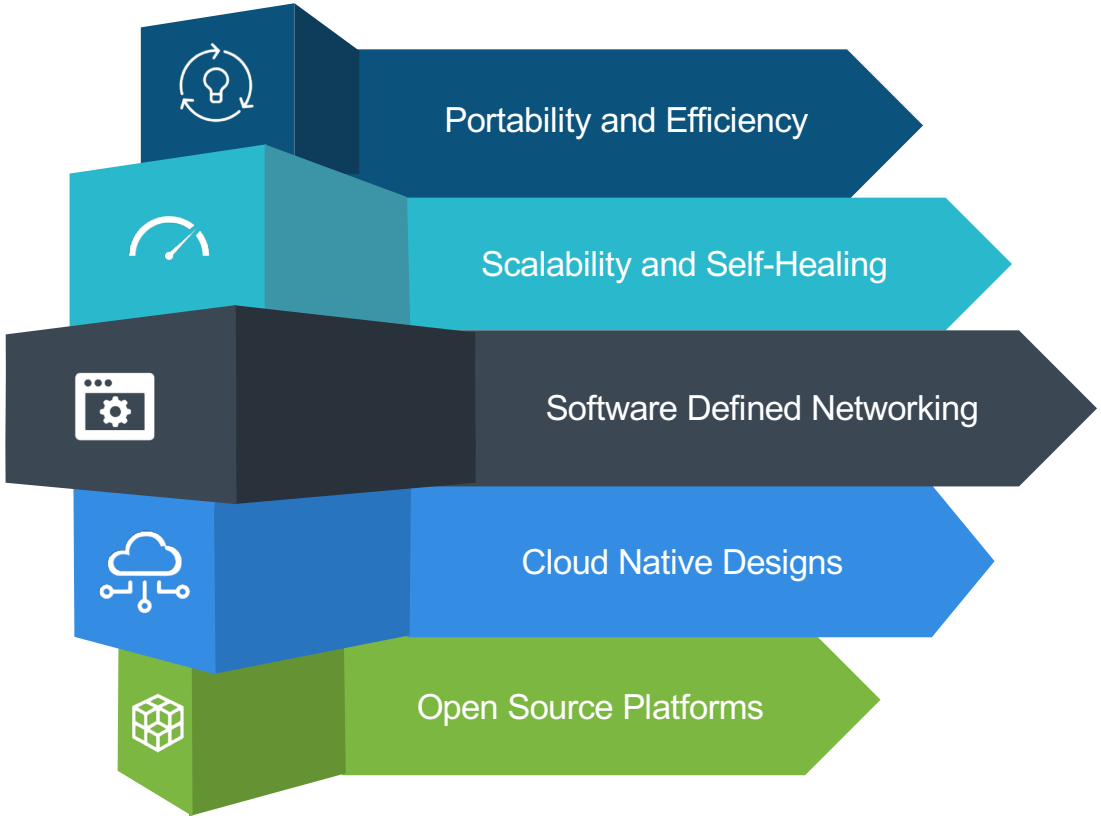
Breaking the Barrier of Software Defined Network Services
1 Terabit Services on a Single Intel® Xeon® Server !

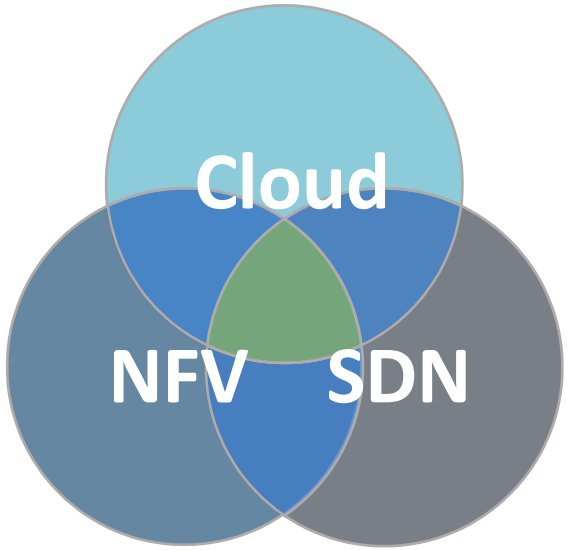
Internet Mega Trends – ..



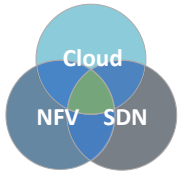


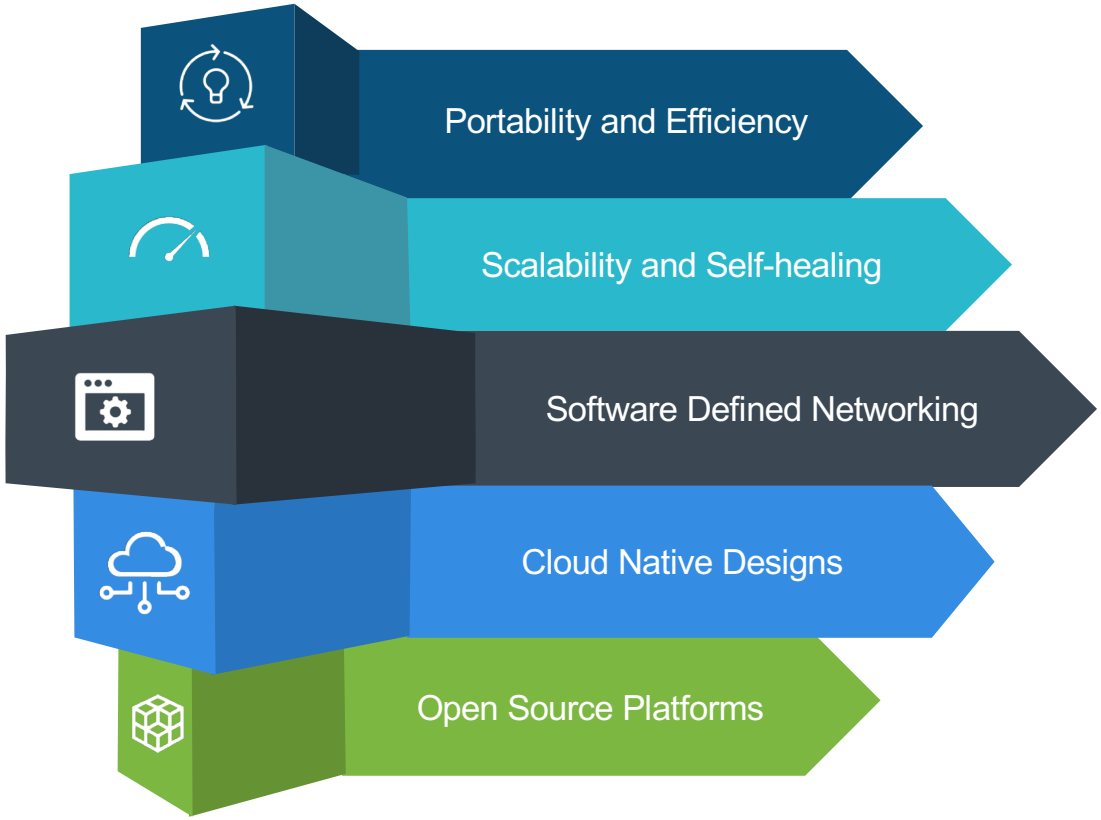
Internet Mega Trends – *Being* Addressed ..





Internet Mega Trends – *Being* Addressed ..





PORTABILITY AND EFFICIENCY

Public, private, hybrid, any-cloud. Over 10 times faster Container networking vs. alternatives.

SCALABILITY and SELF-HEALING

Follows Kubernetes scale and self-healing principles.

SOFTWARE DEFINED NETWORKING

FD.io VPP, the Fastest SW Data Plane on the Planet. Over 200 programmable "micro-NFs" and plugins.

CLOUD NETWORK SERVICES

Containerized NFs managed as true cloud-native apps, provide and consume dat plane microservices.

LINUX FOUNDATION

Based on the best-of-breed collaborative projects in Linux Foundation.

References

FD.io VPP, CSIT and related projects

- VPP: https://wiki.fd.io/view/VPP
- CSIT-CPL: https://wiki.fd.io/view/CSIT
- pma_tools - https://wiki.fd.io/view/Pma_tools

Benchmarking Methodology

- Kubecon Dec-2017, Benchmarking and Analysis.., https://wiki.fd.io/view/File:Benchmarking-sw-data-planes-Dec5_2017.pdf
- "Benchmarking and Analysis of Software Network Data Planes" by M. Konstantynowicz, P. Lu, S.M. Shah, https://fd.io/resources/performance_analysis_sw_data_planes.pdf

NFV Service Density

- Repro guide in FD.io CSIT and packet.net: https://github.com/cncf/cnfs/tree/master/comparison/baseline_nf_performance-packet
- Current results summary: https://raw.githubusercontent.com/cncf/cnfs/master/comparison/doc/cncf-cnfs-results-summary.md

CNCF CNF Project Links

- Collaboration with FD.io CSIT and OPNFV-NFVbench
  - FD.io CSIT
  - OPNFV-NFVbench
- Repo: https://github.com/cncf/cnfs
  - Comparison test case summary and overview
- Comparison and code examples:
  - CNF Edge Throughput and Baseline Single Chain
    - CNF and VNF
    - VPP vSwitch
  - Baseline NF Performance Test
  - In progress: Baseline K8s Chained NF Test
  - Github Projects and Issues

FD.io CSIT Project Links

We invite you to Participate in FD.io

- Get the Code, Build the Code, Run the Code
- Try the vpp user demo
- Install vpp from binary packages (yum/apt)
- Read/Watch the Tutorials
- Join the Mailing Lists
- Join the IRC Channels
- Explore the wiki
- Join FD.io as a member

Thank you!

High Performance Cloud-native Networking

K8s Unleashing FD.io

THANK YOU !!

And Enjoy Seattle !