High Performance Cloud-native Networking
K8s Unleashing FD.io

Giles Heron
Principal Engineer, Cisco
giheron@cisco.com

Maciek Konstantynowicz
FD.io CSIT Project Lead
Distinguished Engineer, Cisco
mkonstan@cisco.com

Jerome Tollet
Distinguished Engineer, Cisco
jtollet@cisco.com
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/rls1807/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 – ..

- Portability and Efficiency
- Scalability and Self-Healing
- Software Defined Networking
- Cloud Native Designs
- Open Source Platforms
THE SOFTWARE DEFINED OPERATOR

DO YOU REMEMBER .. ?
5 Pillars of Next Generation Software Data Planes

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

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

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

**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 optimizations

FD.io VPP meets these challenges

How can one use it in large scale Cloud-native networks?
The Way Applications Are Developed and Deployed Has Changed.
The Way Networks are Deployed and Used… has Changed…

**Corporate LAN/WAN**
- "80:20 rule"
- LAN → WAN
- Intranets & Internet
- LAN → WAN
- SD-WAN & “BeyondCorp”
- WiFi → Internet

**Internet**
- Internet exchanges & public peering
- Tiered Transit & Private Peering
- Tier1 – A → Tier1 – B → Tier1 – C
- Tier2 – D → Tier2 – E → Tier2 – F
- Telco/Cable Access & OTT/CDN Content
- Cloud Provider 2 Global Backbone
- Telco 1 → Telco 2 → MSO 1

**Data-Center**
- Core/Distr/Access, VLAN based
- Spine/Leaves & L3 Core/L2 Access
- L3 Fabric/SW Overlay & Virt Networking
Aside: A Trip Down Memory Lane
(Transporting Data vs. Processing Data)

• Year 2012
  • Internet service provider comment at IETF: **processing bits is cheaper** than transporting bits, computing and networking - networking is becoming 1st order citizen on compute platforms.

• Year 2013
  • RIPE67 Terastream been fixing the cost of transporting bits - 96 of 100GE coherent lambdas per fibre span - **transporting is getting cheaper**, so challenging the compute part again
  • more bandwidth delivered to Data Centres
  • **Most/all network services in Data Dentres**

https://ripe67.ripe.net/presentations/131-ripe2-2.pdf
Remember 1965 "Moore’s Law" – ..
Remember 1965 "Moore’s Law" – ..

Remember **1965** "Moore’s Law" – Is It Still Applicable?

Remember 1965 "Moore’s Law" – Is It Still Applicable?

Remember **1965** "Moore’s Law" – Yes, It Surely Is ..

"Ramble On .."

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

\[
\text{CyclesPerPacket} = \frac{\text{#Instructions}}{\text{Packet}} \times \frac{\text{#Cycles}}{\text{instruction}}
\]

\[
\text{Throughput [pps]} = \frac{1}{\text{Packet Processing Time [sec]}} = \frac{\text{CPU freq [Hz]}}{\text{Cycles per Packet}}
\]

\[
\text{Throughput [bps]} = \text{Throughput [pps]} \times \text{Packet Size [pps]}
\]
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 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)

---

Moore’s Law in Action

\[
\text{Cycles Per Packet} = \frac{\text{Clock Cycles}}{\text{Instructions Packet}} \times \frac{\text{Clock Cycles}}{\text{Instructions Packet}}
\]

\[
\text{Throughput (bps)} = \text{Throughput (pps)} \times \text{Packet Size (bps)}
\]
FD.io VPP – Vector Packet Processing
Compute-Optimised SW Networking Platform

Packet Processing Software Platform
- High performance
- Linux user space
- Runs on compute CPUs:
  - And “knows” how to run them well!

Shipping at volume in server & embedded products

Bare-Metal / VM / Container

Dataplane Management Agent

Packet Processing

Network IO
FD.io VPP – The “Magic” of Vectors
Compute Optimized SW Network Platform

1. Packet processing is decomposed into a directed graph of nodes ...
2. ... packets move through graph nodes in vector ...
3. ... graph nodes are optimized to fit inside the instruction cache ...

* Each graph node implements a "micro-NF", a "micro-NetworkFunction" processing packets.

Microprocessor

3. Instruction Cache
4. Data Cache

... packets are pre-fetched into the data cache.

Makes use of modern Intel® Xeon® Processor micro-architectures.
Instruction cache & data cache always hot ➔ Minimized memory latency and usage.
Cloud-native Network Micro-Services
For Native Cloud Network Services

- **kubernetes**
  - Production-Grade Container Orchestration

- **Contiv**
  - Performance-Centric Container Networking

- **LIGATO**
  - Cloud-native Network Function Orchestration

- **Cloud Native Network Micro-Services**
  - For Native Cloud Network Services

- **Containerized Fast Data Input/Output**

---

Enabling Production-Grade Native Cloud Network Services at Scale

- **Service Policy**
- **Service Topology**
- **Lifecycle**

---

**Production-Grade Container Orchestration**

- Kubernetes
- API Proxies

**Network Function and Network Topology Orchestration**

- Contiv Netmaster

**Containerized Network Data Plane**

- Networking Plugin
- Contiv Netmaster
- Kubelet

---

**Agent**
- FD.io VPP
- Container Switch
- Container Networking
- CNF

---

**Enabling Production-Grade Native Cloud Network Services at Scale**
Contiv-VPP Architecture

K8s Master

K8s State Reflector

K8s policy & state distribution

Data Centre Fabric
Service Function Chaining with Ligato
Ligato – Cloud-native NFs (CNFs)

- Kubernetes does not provide a way to stitch micro-services together today
- Ligato enables you to wire the data plane together into a service topology
- Network functions can now become part of the service topology
- Dedicated Telemetry Engine in VPP to enable closed-loop control
- Offload functions to NIC but via vSwitch in host memory
“Without data, you're just another person with an opinion.” — W. Edwards Deming
Open Source Benchmarking – Guiding Principles

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

“One can’t violate the laws of physics, but one can ’stretch’ them..”
**Benchmarking Data and Public References:**

- **Multi-Platform/-Vendor**
  - Intel & ARM (WiP)
- **Packet Throughput & Latency**
  - Non-Drop & Partial Drop Rates
- **Data Plane Workloads**
  - FD.io VPP
  - DPDK L3fwd, Testpmd
- **Scaling**
  - Single-, Multi-Core
  - MACs, IPs, Flows, ACLs etc.
- **Performance Test Suites (#s)**
  - L2: 58
  - L3 (IPv4 / IPv6): 63
  - VM vhostuser: 26
  - Containers memif: 10
  - Crypto: 13
  - SRv6: 3
  - **Total:** 173

**FD.io CSIT-CPL**

**Per release test and performance reports**

**1T/1C x520 Packet Throughput Tests**

- 10ge2p1x520-ethip4-ip4scale2m
- 10ge2p1x520-ethip4-ip4base
- 10ge2p1x520-eth-l2bdasem.acln
- 10ge2p1x520-eth-l2xbase
- 10ge2p1x520-ethip4-ip4base-l3fwd
- 10ge2p1x520-eth-l2xbase-testpmd

**Newer data available !!**

**Benchmarking Data and Public References:**

- MK

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

**https://docs.fd.io/csit/rls1807/report/index.html**

**https://docs.fd.io/csit/rls1801/report/index.html**

* Selection of testcases from the FD.io CSIT 18.01, 17.10 and 17.07 reports
FD.io CSIT-18.07: Packet Throughput Results

IPv4 Routing (ip4)

- 19 Mpps
- 18 Mpps
- 17 Mpps
- 16 Mpps

200k of /32 hFIB prefixes
20k of /32 hFIB prefixes
2M of /32 hFIB prefixes

L2 Switching with MAC Learning (l2bd)

- 18 Mpps
- 16 Mpps
- 14 Mpps
- 12 Mpps

100k of /48 L2FIB MACs
10k of /48 L2FIB MACs
1M of /48 L2FIB MACs

Source: https://docs.fd.io/csit/rls1807/report/
FD.io CSIT-18.07: Throughput Speedup Results

VPP Multi-Core Speedup
Properties:
- Predictable performance
- Linear scaling with cores
- Follows Amdahl’s Law

* Capped by 14.88 Mpps
10GE 64B link rate limit
VPP: Multi-Core Speedup Properties

VPP Multi-Core Speedup Properties:

• Predictable performance
• Linear scaling with cores
• Follows Amdahl’s Law

Figure 14. Packet throughput speedup with Multithreading and Multi-core.
Packet Vectors are Good for You!

**Netgate** shipping product(s) [1]

**Alibaba** [2]

Baremetal Data Plane Performance Limit
FD.io benefits from increased Processor I/O

YESTERDAY
- Intel® Xeon® E5-2699v4
  - 22 Cores, 2.2 GHz, 55MB Cache
  - Network I/O: 160 Gbps
  - Core ALU: 4-wide parallel µops
  - Memory: 4-channels 2400 MHz
  - Max power: 145W (TDP)

- Socket 0
  - Broadwell Server CPU
- Socket 1
  - Broadwell Server CPU

TODAY
- Intel® Xeon® Platinum 8168
  - 24 Cores, 2.7 GHz, 33MB Cache
  - Network I/O: 280 Gbps
  - Core ALU: 5-wide parallel µops
  - Memory: 6-channels 2666 MHz
  - Max power: 205W (TDP)

Intel® Xeon® v3, v4 Processors
Intel® Xeon® Platinum 8180 Processors

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

0                 200               400               600               800              1000            1200
0                  160               320               560               640               640

PCIe Packet Forwarding Rate [Gbps]

1,120* Gbps

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

https://goo.gl/UtbaHy

Breaking the Barrier of Software Defined Network Services
1 Terabit Services on a Single Intel® Xeon® Server!
Internet Mega Trends – ..

- Portability and Efficiency
- Scalability and Self-Healing
- Software Defined Networking
- Cloud Native Designs
- Open Source Platforms

Cloud
NFV
SDN
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.
High Performance Cloud-Native Networking
K8s Unleashing FD.io

THANK YOU!
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

Opportunities to Contribute

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!