FastDataStacks - Fast and Flexible NFV Solution Stacks Leveraging FD.io

Frank Brockners
Building Cloud/NFV Solution Stacks

• OPNFV performs System Integration as an open community effort:
  • Create/Evolve Components (in lock-step with Upstream Communities)
  • Compose / Deploy / Test
  • Iterate (in a distributed, multi-vendor CI/CD system)
• Let’s add “fast and flexible networking” as another focus…
Foundational Assets For NFV Infrastructure: A stack is only as good as its foundation

- **Forwarder**
  - Feature rich, high performance, highly scalable virtual switch-router
  - Leverages hardware accelerators
  - Runs in user space
  - Modular and easy extensible

- **Forwarder Diversity: Hardware and Software**
  - Virtual Domains link and interact with physical domains

- **Domains and Policy**
  - Connectivity should reflect business logic instead of physical L2/L3 constructs

---

<table>
<thead>
<tr>
<th>Service Model</th>
<th>WorkFlow Topology</th>
<th>App Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service/WF Life Cycle Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual Machine/Container Life Cycle Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Controller Forwarder – Switch/Router</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evolving The OPNFV Solution Set: OPNFV FastDataStacks Project

- OPNFV develops, integrates, and continuously tests NFV solution stacks: Historically OPNFV solution stacks only used OVS as virtual forwarder

- **Objective**: Create a new stacks which significantly evolve networking for NFV

- Current scenarios
  - OpenStack – OpenDaylight (Layer2) – VPP
  - OpenStack – OpenDaylight (Layer3) – VPP
  - OpenStack – VPP
  - ...

- Diverse set of contributors:
  - https://wiki.opnfv.org/display/fds

### Components in OPNFV

<table>
<thead>
<tr>
<th>Category</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Tools</td>
<td>Apex, Compass, Fuel, Juju</td>
</tr>
<tr>
<td>VM Control</td>
<td>OpenStack</td>
</tr>
<tr>
<td>Network Control</td>
<td>OpenDaylight, ONOS, OpenContrail</td>
</tr>
<tr>
<td>Hypervisor</td>
<td>KVM, KVM4NFV</td>
</tr>
<tr>
<td>Forwarder</td>
<td>OVS, OVS-DPDK, + VPP</td>
</tr>
</tbody>
</table>
Introducing Vector Packet Processor - VPP

- VPP is a rapid packet processing development platform for highly performing network applications
  - 18+ MPPS, single core
  - Multimillion entry FIBs
  - 480Gbps bi-dir on 24 cores
- Runs on commodity CPUs and leverages DPDK
- Creates a vector of packet indices and processes them using a directed graph of nodes – resulting in a highly performant solution.
- Runs as a Linux user-space application
- Ships as part of both embedded & server products, in volume; Active development since 2002

- See also: FD.IO (The Fast Data Project)
VPP Universal Fast Dataplane: Performance at Scale [1/2]
Per CPU core throughput with linear multi-thread(-core) scaling

IPv4 Routing

IPv6 Routing

Topology: Phy-VS-Phy

Hardware:
Cisco UCS C240 M4
- Intel® C610 series chipset
- 2 x Intel® Xeon® Processor E5-2698 v3 (16 cores, 2.3GHz, 40MB Cache)
- 2133 MHz, 256 GB Total
- 6 x 2p40GE Intel XL710=12x40GE

Software:
Linux: Ubuntu 16.04.1 LTS
Kernel: ver. 4.4.0-45-generic
FD.io VPP: VPP v17.01-5-g234726 (DPDK 16.11)

Resources
- 1 physical CPU core per 40GE port
- Other CPU cores available for other services and other work
- 20 physical CPU cores available in 12x40GE setup
- Lots of Headroom for much more throughput and features

<table>
<thead>
<tr>
<th>IPv4 Throughput [Mpps]</th>
<th>2x 40GE</th>
<th>4x 40GE</th>
<th>6x 40GE</th>
<th>8x 40GE</th>
<th>12x 40GE</th>
<th>16x 40GE</th>
</tr>
</thead>
<tbody>
<tr>
<td>64B</td>
<td>24.0</td>
<td>45.4</td>
<td>69.7</td>
<td>88.8</td>
<td>109.4</td>
<td>130.9</td>
</tr>
<tr>
<td>128B</td>
<td>24.0</td>
<td>45.4</td>
<td>69.7</td>
<td>88.8</td>
<td>109.4</td>
<td>130.9</td>
</tr>
<tr>
<td>IMX</td>
<td>15.0</td>
<td>30.0</td>
<td>45.0</td>
<td>60.0</td>
<td>75.0</td>
<td>90.0</td>
</tr>
<tr>
<td>1518B</td>
<td>3.8</td>
<td>31.6</td>
<td>16.4</td>
<td>19.2</td>
<td>22.8</td>
<td></td>
</tr>
<tr>
<td>I/O NIC max-pps</td>
<td>35.8</td>
<td>71.6</td>
<td>107.4</td>
<td>145.2</td>
<td>179.0</td>
<td>214.8</td>
</tr>
<tr>
<td>NIC max-bw</td>
<td>46.8</td>
<td>93.5</td>
<td>140.3</td>
<td>187.0</td>
<td>233.8</td>
<td>280.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IPv6 Throughput [Mpps]</th>
<th>2x 40GE</th>
<th>4x 40GE</th>
<th>6x 40GE</th>
<th>8x 40GE</th>
<th>12x 40GE</th>
<th>16x 40GE</th>
</tr>
</thead>
<tbody>
<tr>
<td>64B</td>
<td>19.3</td>
<td>35.4</td>
<td>51.5</td>
<td>67.7</td>
<td>83.8</td>
<td>100.0</td>
</tr>
<tr>
<td>128B</td>
<td>19.3</td>
<td>35.4</td>
<td>51.5</td>
<td>67.7</td>
<td>83.8</td>
<td>100.0</td>
</tr>
<tr>
<td>IMX</td>
<td>15.0</td>
<td>30.0</td>
<td>45.0</td>
<td>60.0</td>
<td>75.0</td>
<td>90.0</td>
</tr>
<tr>
<td>1518B</td>
<td>3.8</td>
<td>31.6</td>
<td>16.4</td>
<td>19.2</td>
<td>22.8</td>
<td></td>
</tr>
<tr>
<td>I/O NIC max-pps</td>
<td>35.8</td>
<td>71.6</td>
<td>107.4</td>
<td>145.2</td>
<td>179.0</td>
<td>214.8</td>
</tr>
<tr>
<td>NIC max-bw</td>
<td>46.8</td>
<td>93.5</td>
<td>140.3</td>
<td>187.0</td>
<td>233.8</td>
<td>280.5</td>
</tr>
</tbody>
</table>
L2 Switching

L2 Switching with VXLAN Tunneling

Packet Throughput [Mpps]
NDR - Zero Frame Loss

Frame Size [Bytes]

Service Scale = 100 thousand MAC L2 entries

Service Scale = 16 thousand MAC L2 entries

MAC Thru [Mpps] 2x 40GE 4 core 4x 40GE 4 core 8x 40GE 8 core 16x 40GE 16 core 16x 40GE 16 core

SM 23.0 19.8 15.9 7.95 3.1 1.38

IM 38.0 16.4 11.9 6.05 1.09 1.38

IMX 20.8 18.4 15.9 7.95 3.1 1.38

12GB 38.0 16.4 11.9 6.05 1.09 1.38

I/O NIC max-gps 95.8 71.6 107.4 143.2 179 214.8

NIC max-lex 86.8 53.5 140.3 187.0 233.8 280.5

L2 Switching

L2 Switching with VXLAN Tunneling

Packet Throughput [Mpps]
NDR - Zero Frame Loss

Frame Size [Bytes]

Service Scale = 100 thousand MAC L2 entries

Service Scale = 16 thousand MAC L2 entries

MAC Thru [Mpps] 2x 40GE 4 core 4x 40GE 4 core 8x 40GE 8 core 16x 40GE 16 core 16x 40GE 16 core

SM 11.6 21.1 38.6 52.2 65.7 79.2

IM 11.6 21.1 38.6 52.2 65.7 79.2

IMX 10.5 21.0 31.5 42.0 52.5 63.0

12GB 8.8 7.6 11.4 15.2 19.0 22.8

I/O NIC max-gps 95.8 71.6 107.4 143.2 179 214.8

NIC max-lex 46.8 93.5 140.3 187.0 233.8 280.5

VPP Universal Fast Dataplane: Performance at Scale [2/2]
Per CPU core throughput with linear multi-thread(-core) scaling

Hardware:
Cisco UCS C240 M4
- Intel® C610 series chipset
- 2 x Intel® Xeon® Processor E5-2698 v3 (16 cores, 2.3GHz, 40MB Cache)
- 2133 MHz, 256 GB Total
- 6 x 2p40GE Intel XL710=12x40GE

Software:
- Linux: Ubuntu 16.04.1 LTS
- Kernel: ver. 4.4.0-45-generic
- FD.io VPP: VPP v17.01-5-ge234726
  (DPDK 16.11)

Resources:
- 1 physical CPU core per 40GE port
- Other CPU cores available for other services and other work
- 20 physical CPU cores available in 12x40GE switch
- Lots of Headroom for much more throughput and features
NFV is about forwarding delay: VPP means low and predictable delay

- Low long-term max packet delay with FD.io VPP
  - 0.007ms < 0.023ms < 3.5ms
- Other vSwitches
  - >120ms long term max delay

Tests environment
- stock Ubuntu 14.04.03 LTS Kernel: 3.13.0-63-generic (no Linux tuning)
- Cisco UCS C460 M4 (4 x Intel® Xeon® Processor E7-8890 v3 (18 cores, 2.5GHz, 45MB Cache), 9 x 2p40GE Intel XL710)

The Soak Test Proof:
18 x 7.7trillion packets forwarded.
Min Packet Delay 7..10 usec, Avg Packet Delay <23 usec.
Max Packet Delay <3.5 msec incl. the outliers!!
FastDataStacks - Solutions Stacks:

Fast and extensible networking natively integrated into OpenStack
Lean Integration of VPP with OpenStack: Networking-VPP Design Principles

- Main design goals are: simplicity, robustness, scalability

- Efficient management communications
  - All communication is asynchronous
  - All communication is REST based

- Robustness
  - Built for failure – if a cloud runs long enough, everything will happen eventually
  - All modules are unit and system tested

- Code is small and easy to understand
Networking-vpp: current feature set

- Network types
  - VLAN: supported since version 16.09
  - VXLAN-GPE: supported since version 17.04

- Port types
  - VM connectivity done using fast vhostuser interfaces
  - TAP interfaces for services such as DHCP

- Security
  - Security groups based on VPP stateful ACLs
  - Port Security can be disabled for true fastpath
  - Role Based Access Control and secure TLS connections for etcd

- Layer 3 Networking
  - North-South Floating IP
  - North-South SNAT
  - East-West Internal Gateway

- Robustness
  - Component state resync in case of failure: recovers from restart of Neutron, the agent and VPP
FastDataStacks: OS – FD.io
Example: 3 node setup: 1 x Controller, 2 x Compute
FastDataStacks – Solutions Stacks for enhanced Network Control:

OpenStack – OpenDaylight – FD.io/VPP
Towards a Stack with enhanced Network Control

- FD.io/VPP
  - Highly scalable, high performance, extensible virtual forwarder
- OpenDaylight Network Controller
  - Extensible controller platform
  - Decouple business logic from network constructs: Group Based Policy as mediator between business logic and network constructs
  - Support for a diverse set of network devices
  - Clustering for HA
Solution Stack Ingredients and their Evolution

- OpenDaylight
  - Group Based Policy (GBP) Neutron Mapper
  - GBP Renderer Manager enhancements
  - VPP Renderer
  - Virtual Bridge Domain Mgr / Topology Manager

- FD.io
  - HoneyComb – Enhancements
  - VPP – Enhancements
  - CSIT – VPP component tests

- OPNFV
  - Overall System Composition – Integration into CI/CD
  - Installer: Integration of VPP into APEX
  - System Test: FuncTest and Yardstick system test application to FDS

See also:
FDS Architecture: https://wiki.opnfv.org/display/fds/OpenStack-ODL-VPP+integration+design+and+architecture
Example: Creating a Neutron vhostuser port on VPP

POST PORT
(id=<uuid>, host_id=<vpp>, vif_type=vhostuser)

Update Port

Map Port to GBP Endpoint

Update/Create Policy involving GBP Endpoint

Resolve Policy

Apply policy, update nodes

Bridge domain and tunnel config

Configure bridge domain on nodes over NetConf

Netconf/YANG

Topology Manager (vBD)

Netconf/YANG

configure interfaces over Netconf

VPP Renderer

GBP Renderer Manager

GBP Neutron Mapper

Neutron NorthBound

Neutron
FastDataStacks: OS – ODL(L2) – FD.io
Example: 3 node setup: 1 x Controller, 2 x Compute
FastDataStacks: OS – ODL(L3) – FD.io
Example: 3 node setup: 1 x Controller, 2 x Compute
FastDataStacks: Status

Colorado 1.0 (September 2016)
- **Base O/S-ODL(L2)-VPP stack** (Infra: Neutron / GBP Mapper / GBP Renderer / VBD / Honeycomb / VPP)
  - Automatic Install
  - Basic system-level testing
  - L2 networking using ODL (no east-west security groups), L3 networking uses qrouter/OVS
  - Overlays: VXLAN, VLAN

Colorado 3.0 (December 2016)
- **Enhanced O/S-ODL(L2)-VPP stack** (Infra complete: Neutron / GBP Mapper / GBP Renderer / VBD / Honeycomb / VPP)
  - Enhanced system-level testing
  - L2 networking using ODL (incl. east-west security groups), L3 networking uses qrouter/OVS
- **O/S-VPP** (Infra: Neutron ML2-VPP / Networking-vpp-agent / VPP)
  - Automatic Install, Overlays: VLAN

Danube 1.0 (March 2017)
- Enhanced O/S-ODL(L3)-VPP stack (Infra complete: Neutron / GBP Mapper / GBP Renderer / VBD / Honeycomb / VPP)
  - **L2 and L3 networking using ODL** (incl. east-west security groups)

Danube 2.0 (May 2017)
- Enhanced O/S-ODL(L3/L2)-VPP stack: **HA for OpenStack and ODL** (clustering)
FastDataStacks – Next Steps

• Simple and efficient forwarding model
  • Clean separation of “forwarding” and “policy”:
    • Pure Layer 3 with distributed routing: Every VPP node serves as a router, “no bridging anywhere”
    • Contracts/Isolation managed via Group Based Policy
  • Flexile Topology Services: LISP integration, complementing VBD

• Analytics integration into the solution stacks
  • Integration of OPNFV projects:
    • Bamboo (PNDA.io for OPNFV)
    • Virtual Infrastructure Networking Assurance (VINA)
    • NFVbench (Full Stack NFVI one-shot benchmarking)

• Container Stack using FD.io/VPP
  • Integrating Docker, K8s, Contiv, FD.io/VPP container networking, Spinnaker
An NFV Solution Stack is only as good as its foundation
THANK YOU