OPNFV & NFV Performance Lab

Industrial Technology Research Institute
ICL-X Yu-Wei, Lee (Ray)
Network Function Virtualization (NFV)

Classical Network Appliance Approach

- CDN
- Session Border Controller
- WAN Acceleration
- DPI
- Carrier Grade NAT
- Tester / QoE Monitor
- SGSN / GGSN
- PE Router
- Firewall

Reduce the Capex and Opex

Independent Software Vendors

Virtual Appliance

Orchestrated automatic & remote install

- Standard High Volume Servers
- Standard High Volume Storage
- Standard High Volume Ethernet Switches
Open Platform for NFV (OPNFV)

Open Platform for NFV (OPNFV) facilitates the development and evolution of NFV components across various open source ecosystems. Through system level integration, deployment and testing, OPNFV creates a reference NFV platform to accelerate the transformation of enterprise and service provider networks.
OPNFV Euphrates Release

Virtual Network Functions

- **Open Baton**
  - Compute Virtualization
  - Storage Virtualization
  - Network Virtualization

- **OpenStack + Kubernetes**
  - KVM
  - LXD
  - Ceph
  - OpenDaylight
  - ONOS
  - OpenContrail
  - OVN

- **Data Plane**
  - FD.io
  - DPDK
  - OVS
  - ODP

- **Infrastructure**
  - Pharos Community Labs
  - OPNFV Bare Metal Lab

Integration
- Alignment
- Installers
- Composition

Testing
- Functional
- System
- Performance

New Features
- NFV Features

Continuous Integration / Continuous Deployment
- Documentation
- Security
OPNFV Testing Ecosystem

**Functest:** VIM and NFVI functional testing.

**Yardstick:** Verification of the infrastructure compliance when running VNF applications.

**VSPerf:** Data-plane performance testing

**Storperf:** Storage Performance testing

**QTIP:** Platform Performance Benchmarking
Intel and ITRI build the NFV performance lab cooperatively

Goal: NFV performance characterization

NFVI characterization requires some “VNF”

VNF characterization requires some NFV infrastructure

**VNF Characterization focus**

**NFVI Characterization focus**
Why Performance Lab

Why Performance Lab
- General lack of telco grade conformance / benchmarks
- Unclear network workload dimensions and stress vectors
- Missing system level capacity requirements
- Network workload scalability/agility implications on NFV
- Operators lack comprehensive information for TCO models to plan, procure and deploy NFV

OPNFV Yardstick – NSB Project (Network Service Benchmarking)
- NFVI/VNF characterization and benchmarking
- Different execution environments:
  - Native Linux environment
  - Standalone virtual environment
  - Managed virtualized environment (e.g. OpenStack etc.)
- Contributed by Intel
VNF and NFV Infrastructure Characterization

1. **System Under Test (SUT)**
2. **Traffic Generator**
3. **KPI & Script**

- **Traffic Generator / Receiver**
  - Generate and receive traffic and measure statistics

- **System Under Test (SUT)**
- **Test Cases:** VNFs, Test Vectors, KPIs
- **VNF**
  - OS, Hypervisor, Soft Switch
  - Hardware Platform
- **Control**
  - Test cases measurement
  - Start and stop SUT workload
  - Start and stop traffic generation
  - Collect statistics
  - Generate report
- **Data Collection**
  - Receiver / Measurements Instrumentation

**Diagram Notes:**
- Control and Data Collection
- Test cases measurement
- Start and stop SUT workload
- Start and stop traffic generation
- Collect statistics
- Generate report
Generators

- IXIA
- SPIRENT
- PKTGEN
- MOONGEN
- PROX
- TREX
- Ostinato
- IPerf
- NG4T
- IMS Bench SIPP
- And many others

- Commercial HW tools & VNF – full stack support
- DPDK based tools
- Non DPDK based tools
- Commercial tool for EPC characterization
- Open Source tool for IMS characterization
Throughput

Forwarding Rate at Maximum Offered Load (FROMOL, RFC 2889)
- Generate at line rate, and measure forwarded packets
- Easy to run, fast
- Measure high load behavior (overload?)
- Might represent a completely different number than maximum forwarding rate

Maximum Forwarding Rate (MFR, RFC 2889)
- Start generating at 100% line rate and binary search for higher rate without packet loss
- Quite fast
- Might be very sensitive to spurious packet loss

Measure packet loss for any rate, starting from 0.1% to 100% of line rate
- Slow
- Better picture of the performance
- Will highlight spurious packet loss
Intel Xeon Processor E5-2695 v4
NFVI Performance Report

Produced by ITRI Performance Lab
NFVI Characterization using NSB

Control and Data Collection
- Test cases measurement
- Start and stop SUT workload
- Start and stop traffic generation
- Collect statistics
- Generate report

System Under Test (SUT)

Traffic Generator
- Instrumentation

Traffic Generator: PROX

Traffic Generator / Receiver
- Generate and receive traffic and measure statistics

Receiver / Measurements Instrumentation

Control
- Test cases measurement
- Start and stop SUT workload
- Start and stop traffic generation
- Collect statistics
- Generate report

Data Collection

Instrumentation

VNF
- L2-Fwd
- L3-Fwd
- ACL
- MPLS

Packet Size:
- 64 Bytes
- 128 Bytes
- 256 Bytes
- 512 Bytes
- 1024 Bytes
- 1280 Bytes

Test Cases:
- VNFs
- Test Vectors
- KPIs

OS, Hypervisor, Soft Switch

Hardware Platform

Instrumentation
## Platform Configuration

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Server Platform** | Supermicro X10DRH-C/I  
Dual Integrated 1GbE ports via Intel® i350-AM2 Gigabit Ethernet          |
| **Chipset**   | Intel® C612 chipset (formerly Lynx-H Chipset)                               |
| **Processor** | 2x Intel® Xeon® Processor E5-2695 v4 (formerly Broadwell-EP)  
2.10 GHz; 120 W; 45 MB cache per processor  
18 cores (HT disabled)                                    |
| **Memory**    | 64GB Total; Samsung 8GB 2Rx8 PC4-2400MHz, 8GB per channel, 4 Channels       |
| **PCIe**      | 6x PCI-E 3.0 x8 slot  
1x PCI-E 3.0 x16 slot                                                      |
| **Local Storage** | 2 x Seagate Nytro® XFI230 SATA SSD 240G                                    |
| **NICs**      | 2 x Intel® Ethernet Converged Network Adapter X710-DA4  
Total: 8 Ports; 4 ports are used in tests.                               |
| **BIOS**      | AMIBIOS Version: 2.0a Build Date : 06/30/2016                               |
# Software Configurations (1/2)

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Operating System</td>
<td>Ubuntu 16.04</td>
</tr>
<tr>
<td></td>
<td>Kernel version: 4.4.0-101-generic</td>
</tr>
<tr>
<td>VM Operating System</td>
<td>Ubuntu 16.04</td>
</tr>
<tr>
<td></td>
<td>Kernel version: 4.4.0-98-generic</td>
</tr>
<tr>
<td>QEMU-KVM</td>
<td>QEMU emulator version 2.10.1 (v2.10.1-dirty)</td>
</tr>
<tr>
<td>Open vSwitch (native OvS and OvS with DPDK)</td>
<td>Open vSwitch 2.8.1</td>
</tr>
<tr>
<td></td>
<td>DB Schema 7.15.0</td>
</tr>
<tr>
<td>Intel® Ethernet Drivers (Physical Function)</td>
<td>driver: i40e</td>
</tr>
<tr>
<td></td>
<td>version: 2.2.4</td>
</tr>
<tr>
<td></td>
<td>firmware-version: 5.05 0x80002a9b 1.1313.0</td>
</tr>
<tr>
<td></td>
<td>expansion-rom-version:</td>
</tr>
<tr>
<td></td>
<td>bus-info: 0000:04:00.2</td>
</tr>
<tr>
<td></td>
<td>supports-statistics: yes</td>
</tr>
<tr>
<td></td>
<td>supports-test: yes</td>
</tr>
<tr>
<td></td>
<td>supports-eeprom-access: yes</td>
</tr>
<tr>
<td></td>
<td>supports-register-dump: yes</td>
</tr>
</tbody>
</table>
## Software Configurations (2/2)

<table>
<thead>
<tr>
<th>System Compilation</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| **Intel® Ethernet Drivers (Virtual Function)** | driver: i40evf  
version: 3.1.4  
firmware-version: 5.05 0x80002a9b 1.1313.0  
supports-statistics: yes  
supports-test: yes  
supports-eeprom-access: yes  
supports-register-dump: yes |
| **DPDK** | DPDK version: 17.05.2  
(http://fast.dpdk.org/rel/dpdk-17.05.2.tar.xz)  
Compiled with:  
EXTRA_CFLAGS="–Ofast –march=native" make –j10 |
| **Packetgen** | PROX (HEAD detached at 4a31f76) |
| **OVS** | OVS compiled as follows:  
# .configure --with-dpdk=$DPDK_DIR/x86_64-native-linuxapp-gcc  
CFLAGS="–Ofast –march=native"  
# make "CFLAGS=–Ofast –march=native" -j10 |
# BIOS Tuning Configurations (1/2)

<table>
<thead>
<tr>
<th>Menu (Advanced)</th>
<th>BIOS Setting</th>
<th>Required Settings for Performance</th>
<th>BIOS Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Configuration yrs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Power Management Configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hyper-Threading (ALL)</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td></td>
<td>Power Technology</td>
<td>Disable</td>
<td>Custom</td>
</tr>
<tr>
<td></td>
<td>Energy Performance Tuning</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td></td>
<td>Energy Performance BIAS Setting</td>
<td>Performance</td>
<td>Enable</td>
</tr>
<tr>
<td></td>
<td>Energy Efficient Turbo</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td>-&gt; CPU P State Control</td>
<td>EIST (P-States)</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td>-&gt; CPU P State Control</td>
<td>Turbo Mode</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td>-&gt; CPU P State Control</td>
<td>P-State Coordination</td>
<td>HW_ALL</td>
<td>HW_ALL</td>
</tr>
<tr>
<td>-&gt; CPU C State Control</td>
<td>Package C State Limit</td>
<td>[C0/C1 State]</td>
<td>[C6 (Retention)]</td>
</tr>
<tr>
<td>-&gt; CPU C State Control</td>
<td>CPU C3 Report</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td>-&gt; CPU C State Control</td>
<td>CPU C6 Report</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td>-&gt; CPU C State Control</td>
<td>Enhanced Halt State (C1E)</td>
<td>Disable</td>
<td>Enable</td>
</tr>
</tbody>
</table>

Ref:
Intel® Xeon® Processor E5-2695 v4 Performance report: Open vswitch with dpdk(OVS-DPDK) VS. VT-d PCI passthrough P.10
## BIOS Tuning Configurations (2/2)

*In passthrough test cases, we will turn on VT-d

<table>
<thead>
<tr>
<th>Chipset Configuration</th>
<th>EV DFX Features</th>
<th>Intel VT for Directed I/O (VT-d)</th>
<th>Discrete Graphics</th>
<th>Enforce POR</th>
<th>Memory Frequency</th>
<th>DRAM RAPL Baseline</th>
<th>A7 Mode</th>
<th>EHCI Hand-off</th>
<th>USB3.0 Support</th>
<th>ASPM</th>
<th>Onboard LAN 1 OROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>North Bridge - IIO</td>
<td>Disable</td>
<td>Disable</td>
<td>Enable</td>
<td>Disable</td>
<td>2400</td>
<td>Auto</td>
<td>Enable</td>
<td>Disable</td>
<td>Disable</td>
<td>Disable</td>
<td>Disable</td>
</tr>
<tr>
<td>North Bridge - IOAT</td>
<td>Enable IOAT</td>
<td>Enable</td>
<td>Enable</td>
<td>Auto</td>
<td>Disable</td>
<td>Auto</td>
<td>Enable</td>
<td>Auto</td>
<td>Disable</td>
<td>Disable</td>
<td>Disable</td>
</tr>
<tr>
<td>North Bridge - QPI</td>
<td>Link L0 P</td>
<td>Disable</td>
<td>Enable</td>
<td>Auto</td>
<td>Enable</td>
<td>Auto</td>
<td>Enable</td>
<td>Auto</td>
<td>Disable</td>
<td>Disable</td>
<td>Enable</td>
</tr>
<tr>
<td>North Bridge - Memory</td>
<td>Enforce POR</td>
<td>Disable</td>
<td>Enable</td>
<td>Auto</td>
<td>Enable</td>
<td>Auto</td>
<td>Enable</td>
<td>Disable</td>
<td>Disable</td>
<td>Disable</td>
<td>Disable</td>
</tr>
<tr>
<td>South Bridge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCIe/PCI/PnP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** VT-d refers to Virtualization Technology Directed I/O.
The Mapping Between CPU and NIC

- The CPU core and NIC for packet generator should be in the same CPU socket, or the QPI interface will be the performance bottleneck.
  - 10GbE line rate (64 bytes packet) = 10.00e9 bits / (8 bits * (64 + 20) bytes) = 14.88e6 packets.
  - CPU and NIC in same socket, 10G port could generate 14.88 Mpps.
  - CPU and NIC in different socket, 10G port only could generate 10 Mpps.
Baremetal Architecture

GEN

Socket 0
10 Gigabit Ethernet X710

Socket 1
10 Gigabit Ethernet X710

SUT

Socket 1
10 Gigabit Ethernet X710

Socket 1
10 Gigabit Ethernet X710
SRIOV Passthrough Architecture

GEN

Socket 0
10 Gigabit Ethernet X710

Socket 1
10 Gigabit Ethernet X710

SUT

VM

Socket 1
OVS-DPDK Architecture

GEN

Socket 0
10 Gigabit Ethernet X710

Socket 1
10 Gigabit Ethernet X710

SUT

Socket 1

OVS-DPDK

dpdk0 vhostuser0

dpdk1 vhostuser1

dpdk2 vhostuser2

dpdk3 vhostuser3

VM
SRIOV Performance

<table>
<thead>
<tr>
<th>Throughput (Mpps)</th>
<th>L2-Fwd Rx</th>
<th>L2-Fwd Tx</th>
<th>L3-Fwd Rx</th>
<th>L3-Fwd Tx</th>
<th>ACL Rx</th>
<th>ACL Tx</th>
<th>MPLS Rx</th>
<th>MPLS Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM</td>
<td>59.52</td>
<td>59.52</td>
<td>53.29</td>
<td>53.29</td>
<td>37.38</td>
<td>33.82</td>
<td>59.52</td>
<td>58.17</td>
</tr>
<tr>
<td>VM w/ NUMA</td>
<td>59.52</td>
<td>59.52</td>
<td>53.29</td>
<td>53.29</td>
<td>37.41</td>
<td>33.78</td>
<td>59.52</td>
<td>58.17</td>
</tr>
<tr>
<td>VM w/o NUMA</td>
<td>59.52</td>
<td>59.46</td>
<td>53.36</td>
<td>43.4</td>
<td>37.46</td>
<td>30.66</td>
<td>59.52</td>
<td>56.96</td>
</tr>
</tbody>
</table>

VM performance with NUMA-aware configuration is similar to Baremetal performance.

VM without NUMA-aware configuration may decrease 10 – 20% performance.

Baremetal vs SRIOV 4 Port - Throughput (64Bytes)
OVS-DPDK Performance

SRIOV could get better performance than OVS-DPDK in small packet size scenario.

OVS-DPDK vs SRIOV vs Baremetal 4 Port - Throughput (64Bytes)

<table>
<thead>
<tr>
<th></th>
<th>L2-Fwd Tx</th>
<th>L3-Fwd Tx</th>
<th>ACL Tx</th>
<th>MPLS Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVS-DPDK-1-Core</td>
<td>2.51</td>
<td>2.33</td>
<td>2.24</td>
<td>2.71</td>
</tr>
<tr>
<td>OVS-DPDK-2-Cores</td>
<td>6.31</td>
<td>4.71</td>
<td>4.64</td>
<td>6.11</td>
</tr>
<tr>
<td>OVS-DPDK-4-Cores</td>
<td>12.96</td>
<td>9.06</td>
<td>10.1</td>
<td>11.36</td>
</tr>
<tr>
<td>SRIOV</td>
<td>59.52</td>
<td>53.29</td>
<td>33.78</td>
<td>58.17</td>
</tr>
<tr>
<td>BM</td>
<td>59.52</td>
<td>53.29</td>
<td>33.82</td>
<td>58.17</td>
</tr>
</tbody>
</table>
OVS-DPDK Performance

OVS-DPDK vs SRIOV vs Baremetal 4 Port - Throughput (L2-Fwd)

Throughput (Mpps)

- 64 Bytes
- 128 Bytes
- 256 Bytes
- 512 Bytes
- 1024 Bytes
- 1280 Bytes

<table>
<thead>
<tr>
<th></th>
<th>64 Bytes</th>
<th>128 Bytes</th>
<th>256 Bytes</th>
<th>512 Bytes</th>
<th>1024 Bytes</th>
<th>1280 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVS-DPDK-1-Core</td>
<td>2.51</td>
<td>2.46</td>
<td>2.38</td>
<td>2.26</td>
<td>1.74</td>
<td>1.47</td>
</tr>
<tr>
<td>OVS-DPDK-2-Cores</td>
<td>6.31</td>
<td>6.1</td>
<td>5.64</td>
<td>4.98</td>
<td>3.71</td>
<td>3.04</td>
</tr>
<tr>
<td>OVS-DPDK-4-Cores</td>
<td>12.96</td>
<td>12.5</td>
<td>11.82</td>
<td>9.32</td>
<td>4.77</td>
<td>3.83</td>
</tr>
<tr>
<td>SRIOV</td>
<td>59.52</td>
<td>32.89</td>
<td>17.85</td>
<td>9.32</td>
<td>4.77</td>
<td>3.83</td>
</tr>
<tr>
<td>BM</td>
<td>59.52</td>
<td>32.89</td>
<td>17.85</td>
<td>9.32</td>
<td>4.77</td>
<td>3.83</td>
</tr>
</tbody>
</table>

OVS-DPDK performance is similar to SRIOV performance in large packet size scenario.
Intel Atom Processor C3758
SDWAN Performance Report

Produced by ITRI Performance Lab
SDWAN Topology

Enterprise Branch Node (DUT)
- vDPI (VM)
  - DPI app
  - Guest OS
  - virtio
  - vhost (DPDK)

- vIPSec (VM)
  - IPSec app
  - Guest OS
  - virtio

Traffic Termination Endpoint
- vDPI (VM)
  - DPI app
  - Guest OS
  - virtio

- vIPSec (VM)
  - IPSec app
  - Guest OS
  - virtio

Host OS / HW
- phys (DPDK)
- phys port

OvS/DPDK

Ixia Traffic Generator

Uplink
Downlink
Bi-directional link
# CPU Core Assignment – 4 Cores, 1 PMD Thread

## 4-Core Configuration

<table>
<thead>
<tr>
<th></th>
<th>Core 0</th>
<th>Core 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host OS</td>
<td>Core 0</td>
<td>Core 0</td>
</tr>
<tr>
<td>Hypervisor (KVM/Qemu)</td>
<td>Core 0</td>
<td>Core 0</td>
</tr>
<tr>
<td>OVS Mgmt Threads</td>
<td>Core 0</td>
<td>Core 0</td>
</tr>
<tr>
<td>OVS DPDK PMD</td>
<td>Core 1</td>
<td></td>
</tr>
<tr>
<td>DPI VNF (vCPU threads)</td>
<td>Core 0 (VM vCPU 0), Core 2 (VM vCPU 1)</td>
<td>Core 0 (VM vCPU 0), Core 2 (VM vCPU 1)</td>
</tr>
<tr>
<td>IPSec VNF (vCPU threads)</td>
<td>Core 0 (VM vCPU 0), Core 3 (VM vCPU 1)</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram](image-url)
SDWAN Performance with 4 Cores

Use QAT accelerator can improve IPSec performance

QAT performance will be decrease when flow number is increase, OVS is the performance bottleneck.

Performance test results (Intel® C3758 Atom™) - 4 Cores

<table>
<thead>
<tr>
<th>Flows</th>
<th>ITRI Lab SHA1 - 10G</th>
<th>ITRI Lab SHA1 (QAT) - 10G</th>
<th>ITRI Lab GCM - 10G</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1,469.20</td>
<td>2,378.46</td>
<td>2,550.45</td>
</tr>
<tr>
<td>500</td>
<td>1,469.20</td>
<td>2,176.69</td>
<td>2,330.27</td>
</tr>
<tr>
<td>1,000</td>
<td>1,466.45</td>
<td>2,107.14</td>
<td>2,234.31</td>
</tr>
<tr>
<td>2,000</td>
<td>1,454.09</td>
<td>2,049.09</td>
<td>2,107.14</td>
</tr>
</tbody>
</table>
CPU Core Assignment – 6 Cores, 3 PMD Thread

### 6-Core Configuration

<table>
<thead>
<tr>
<th>Component</th>
<th>Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host OS</td>
<td>Core 0</td>
</tr>
<tr>
<td>Hypervisor (KVM/Qemu)</td>
<td>Core 0</td>
</tr>
<tr>
<td>OVS Mgmt Threads</td>
<td>Core 0</td>
</tr>
<tr>
<td>OVS DPDK PMD</td>
<td>Core 1, Core 2, Core 3</td>
</tr>
<tr>
<td>DPI VNF (vCPU threads)</td>
<td>Core 0 (VM vCPU 0), Core 4 (VM vCPU 1)</td>
</tr>
<tr>
<td>IPsec VNF (vCPU threads)</td>
<td>Core 0 (VM vCPU 0), Core 5 (VM vCPU 1)</td>
</tr>
</tbody>
</table>

![Diagram of CPU Core Assignment](image)
SDWAN Performance of QAT Scenario

QAT performance will be increase when we allocate more CPU core to OVS.

4 Cores v.s 6 Cores (QAT Scenario)
SDWAN Performance on 6 Core Environment

The performance of IPSec SHA without QAT is almost the same with 4 core scenario, bottleneck is IPSec itself.
CPU Core Assignment – 7 Cores, 3 PMD Thread

### 7-Core Configuration

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host OS</strong></td>
<td>Core 0</td>
</tr>
<tr>
<td><strong>Hypervisor (KVM/Qemu)</strong></td>
<td>Core 0</td>
</tr>
<tr>
<td><strong>OVS Mgmt Threads</strong></td>
<td>Core 0</td>
</tr>
<tr>
<td><strong>OVS DPDK PMD</strong></td>
<td>Core 1, Core 2, Core 3</td>
</tr>
<tr>
<td><strong>DPI VNF (vCPU threads)</strong></td>
<td>Core 0 (VM vCPU 0), Core 4 (VM vCPU 1)</td>
</tr>
<tr>
<td><strong>IPsec VNF (vCPU threads)</strong></td>
<td>Core 0 (VM vCPU 0), Core 5 (VM vCPU 1), Core 6 (VM vCPU 2)</td>
</tr>
</tbody>
</table>
SDWAN Performance of AES-NI Scenario

The performance of IPSec AES-NI will be increase when we allocate more CPU core to IPSec VNF.
Intel Xeon Processor E5-2695 v4
Performance Report: 2-8 VMs Service Chain
with SR-IOV, OVS-DPDK, VPP and SPP

Produced by ITRI Performance Lab
NTT’s Presentation on DPDK Summit

Implementation and Testing of Soft Patch Panel

Yasufumi Ogawa (NTT)
Tetsuro Nakamura (NTT)
DPDK Summit - San Jose – 2017
VM Chaining Scenario

- Large-scale cloud for telecom services
- Service Function Chaining for virtual network appliances
- Flexibility, Maintainability and High-Performance
8 VMs Service Chain Test Setup Diagram (SRIOV)

Diagram (SRIOV)

- VT-d PCI Passthrough
- Ubuntu-16.04
- Intel® Ethernet X710-DA4
- PROX Software Packet Generator

VM1 (DPDK testpmd)
VM2 (DPDK testpmd)
VM3 (DPDK testpmd)
VM4 (DPDK testpmd)
VM5 (DPDK testpmd)
VM6 (DPDK testpmd)
VM7 (DPDK testpmd)
VM8 (DPDK testpmd)
8 VMs Service Chain Test Setup Diagram (OVS-DPDK)

1. VM1 (DPDK testpmd)
2. VM2 (DPDK testpmd)
3. VM3 (DPDK testpmd)
4. VM4 (DPDK testpmd)
5. VM5 (DPDK testpmd)
6. VM6 (DPDK testpmd)
7. VM7 (DPDK testpmd)
8. VM8 (DPDK testpmd)

VT-d PCI Passthrough

Qemu-KVM

Ubuntu-16.04

VT-d PCI Passthrough

PROX
Software Packet Generator

Intel® Ethernet Converged Network Adapters X710-DA4
8 VMs Service Chain Test Setup Diagram (SPP)

VT-d PCI Passthrough

Soft Patch Panel (SPP)

Ubuntu-16.04

Intel® Ethernet Converged Network Adapters X710-DA4

PROX
Software Packet Generator
8 VMs Service Chain Test Setup Diagram (VPP)

- 8 VMs using DPDK testpmd
- Qemu-KVM
- Ubuntu-16.04
- Intel® Ethernet Converged Network Adapters X710-DA4
- PROX Software Packet Generator

Diagram:
- VM1 (DPDK testpmd) connected to Qemu-KVM
- VM2 (DPDK testpmd) connected to Qemu-KVM
- VM3 (DPDK testpmd) connected to Qemu-KVM
- VM4 (DPDK testpmd) connected to Qemu-KVM
- VM5 (DPDK testpmd) connected to Qemu-KVM
- VM6 (DPDK testpmd) connected to Qemu-KVM
- VM7 (DPDK testpmd) connected to Qemu-KVM
- VM8 (DPDK testpmd) connected to Qemu-KVM

VT-d PCI Passthrough connections between VMs and Qemu-KVM.
Performance Result : Summary

2 to 8 VMs Service Chain Performance
64 Byte, 10K Flow
(Bi-Direction, total 20G)

Binary-Search Pktgen
2 to 8 VMs Service Chain Performance:
64 Bytes, 10K Flow, Bi-Direction, Packet per second

SRIOV’s performance is not good in VM chaining scenario, the bottleneck is NIC’s limitation.

Phy-VM-Phy Core Scalability Performance running One Flows on Intel® Xeon® Processor E5-2695 v4
Multiple VM Service Chain (64Byte, 10K Flow, Bi-direction)

<table>
<thead>
<tr>
<th>VMs</th>
<th>SRIOV</th>
<th>SPP-vhost-4PMD</th>
<th>VPP-vhost-4PMD</th>
<th>OVS-DPDK-4PMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 VMs</td>
<td>18370.39</td>
<td>18405.19</td>
<td>10959.9</td>
<td>8367.34</td>
</tr>
<tr>
<td>3 VMs</td>
<td>9176</td>
<td>9829.09</td>
<td>10083.83</td>
<td>8263.44</td>
</tr>
<tr>
<td>4 VMs</td>
<td>6106.92</td>
<td>6117.86</td>
<td>5393</td>
<td>3566.33</td>
</tr>
<tr>
<td>5 VMs</td>
<td>4582.13</td>
<td>4523.16</td>
<td>5143.81</td>
<td>3537.34</td>
</tr>
<tr>
<td>6 VMs</td>
<td>3610.25</td>
<td>3624.33</td>
<td>3581.41</td>
<td>2290.57</td>
</tr>
<tr>
<td>7 VMs</td>
<td>3004.99</td>
<td>2786.65</td>
<td>3363.37</td>
<td>2228.31</td>
</tr>
<tr>
<td>8 VMs</td>
<td>2580.51</td>
<td>2343.46</td>
<td>2522.53</td>
<td>1669.36</td>
</tr>
</tbody>
</table>
Performance Result: OVS-DPDK

2 to 8 VMs Service Chain Performance
64 Byte, OVS-DPDK, 4 PMD
with Different number of Flow
(Single Direction 10G / Bi-Direction 20G)

Full-Speed Pktgen
2 to 8 VMs Service Chain Performance: 64 Bytes, OVS-DPDK, 4 PMD, Single Direction

OVS performance will be effected by flow number, it may cause 20% performance decrease.

Phy-VM-Phy Core Scalability Performance running One Flows on Intel® Xeon® Processor E5-2695 v4
Multiple VM Service Chain (64 Bytes, OVS-DPDK, 4 PMD, Bi-direction)
NFV Performance Lab

- **VM/Container Performance Tuning**
  - CPU pinning
  - NUMA configuration
  - BIOS configuration

- **Data Plane Acceleration**
  - SRIOV passthrough (include SmartNIC)
  - Enable DPDK (OVS-DPDK, VPP, SPP...etc)
  - QAT, Intel AES-NI

- **Scenario / Use Cases**
  - NFVI / VNF performance characterization
  - SDWAN scenario (uCPE, DPI, IPSEC)
  - VM chaining with different data plane
  - More use cases will be added into lab.
Thanks for your attention