White Paper

Cisco and Intel
High-Performance VNFs on Cisco NFV Infrastructure

Cisco and Intel have created a strategic partnership to develop a unique environment to foster innovation and virtualization adoption. The initiative enables service providers' network virtualization by making available high-performance virtual network functions (VNFs), with run on standard network function virtualization infrastructure (NFVI) environments.

Contents

Executive Summary 3

Challenges and Objectives 3

Specific Obstacles to Overcome 3

Cisco and Intel Solutions to Challenges 4

- Cisco Vector Packet Processing Integrated with the Data Plane Development Kit as a VNF Data Plane 4

About the PoC Environment 5

SDN in the PoC Environment 5

PoC Methodology 6

PoC Results and Findings 6

Conclusion 6

Additional Sources 6

Configuration Details 7

Executive Summary

Network function virtualization (NFV) is fast becoming a disruptive force in the service provider landscape. Successful adoption of NFV in production deployments requires a combination of VNFs and NFVI, which can together deliver high data throughput and low latency. However, software-based packet processing, particularly in a virtualized environment, suffers from inefficiencies that limit the packet processing performance.

This proof of concept (PoC) uses important innovations from Cisco and Intel that have demonstrated the ability to deliver Layer 3 packet processing at 10-Gbps line rate per port with zero packet drops and minimal latency and jitter.

Challenges and Objectives

When running VNFs on a virtual infrastructure, you still expect levels of efficiency associated with physical network hardware: high packet throughput, low latency, minimal jitter, and zero packet drops.

It is important to recognize that virtual machines running network workloads are highly sensitive to their underlying infrastructure. Virtual machines running on infrastructural, as opposed to cloud platforms (where the aim is to act as a substitute for an established physical server while providing equivalent levels of reliability) tend to get controlled environments. Virtual machines are scheduled by constraints rather than precise placement, and they share their server resources with unknown and unpredictable neighbors. This sharing can lead to a situation where a virtual machine is capable of delivering good headline performance but performs poorly when running on cloud NFVI platforms.

The goal of this PoC is to demonstrate that, by combining innovations from both Cisco and Intel, it is possible to achieve impressive throughput rates for high-performance VNFs running on an NFVI using Cisco® and Intel® hardware and Red Hat OpenStack.

Specific Obstacles to Overcome

The general challenges of operating high-performance VNFs on standard NFVI cloud architecture translates to specific issues within elements of the NFVI system:

- NFVI computing—Workloads cannot guarantee uninterrupted CPU time, and every interruption stalls packet processing for a variable quantity of time, leading to latency, jitter, and ultimately packet loss.
- NFVI networking—Traditional cloud computing typically involves soft switches optimized for shared cloud workloads. Many issues arise, including limitations on flow counts and on the maximum number of packets processed per second.
- Virtual machine hypervisor—The hypervisor emulates certain privileged functions of a physical machine in ways that take considerably longer than the bare-metal machine itself would. The time required by the emulator for supervisory activities also frequently competes for CPU cycles with what the virtual machine needs.
- Virtual machine—Control plane elements within a VNF's virtual machine can compete for CPU time with the packet forwarding code.
- VNF orchestration—Automation is desired. Manual deployment and optimization of workloads is not an option in an NFV environment.
- General—The number of CPU cycles per packet is so small that even cache misses can cause packet processing to exceed the CPU cycles available.

Cisco and Intel Solutions to Challenges

In mitigating the NFVI computing and network challenges, it was essential to remain within the feature set offered by the Linux kernel and the tools commonly used in an OpenStack deployment. With appropriate tuning and without change to the infrastructure, it is possible to dedicate hardware resources to high-performance VNFs such that the VNF executes optimally.

Intel Enhanced Platform Awareness (EPA) in OpenStack enables deployment of virtual machines on server platforms with the specific hardware and silicon capabilities that are optimal for the particular needs of virtual machines. This deployment helps ensure that the application is provided with the most suitable combination of hardware resources and features available across the spectrum of available systems in the data center. This is key to service level agreement (SLA) adherence, providing performance and functionality, while maintaining a high infrastructure utilization level and reducing cost per unit of workload.

From an NFV perspective, OpenStack does not need to be innately aware of the NFV applications or their functions. However, OpenStack does need the ability to provide a suitably advanced selection of tuning capabilities that enable service providers to deploy their NFV with the necessary performance and efficiency characteristics. Following are some of these tuning (EPA) capabilities:

- Non-uniform memory access (NUMA) Extensions—In a NUMA environment, such as a modern Intel Architecture (IA) server, some areas of memory and I/O devices are closer to—and faster to access from—some CPU sockets. Intel NUMA functionality in OpenStack ensures that guests can be deployed to take advantage of this locality. Proper NUMA settings ensure that code is adjacent to the CPU core executing it and to the network cards that it is using with significant performance improvements.
- Huge Pages—Virtual memory is mapped to physical memory in Intel servers by means of page tables in the host's memory. Using huge 2 MB or 1 GB pages instead of the default 4 KB significantly reduces the frequency of page table lookups and the lost CPU time that goes along with them.
- CPU pinning—Pinning ensures that a virtual CPU (vCPU) in the virtual machine always runs on the same physical CPU (pCPU). This pinning ensures that the cache is not polluted by the needs of other processes, and that the virtual machine always has the information it needs close by.
- PCI Passthrough—By handing complete control of a PCI device to a virtual machine, the device can be used by the data plane with significantly more efficiency and less overhead.

However, EPA enhancements alone are not enough. High-performance virtual machines can also harness the power of the data plane development kit (DPDK) for efficiency of packet processing. Using DPDK, the Cisco Vector Packet Processing (VPP) data plane serves as the packet data path for the virtual machine, building on DPDK's fundamentals to allow high-performance packet processing and forwarding while keeping within the tight cycles' per-packet budget.

Cisco Vector Packet Processing Integrated with the Data Plane Development Kit as a VNF Data Plane

DPDK is a set of libraries and drivers that take advantage of Intel instruction-set-architecture features to accelerate packet processing on Intel architectures. The Intel Open Network Platform (ONP) reference architecture demonstrates accelerated packet processing with DPDK.

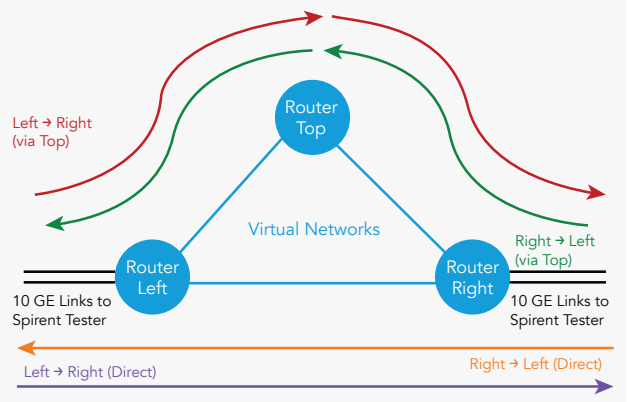
Cisco VPP uses DPDK to provide optimal packet-processing performance. VPP is an innovation from Cisco that is a full-featured network subsystem with highly optimized software packet processing and forwarding engine, enabling a very lightweight and high-performance software data plane. Compared to a traditional scalar forwarding engines, Cisco VPP allows for parallel processing of multiple packets to maximize performance, vastly reducing the cycles required to process each packet.

Cisco is proud to note that VPP technology is now available in an open source project under the Linux Foundation, FD.io. Intel and Cisco have collaborated on optimizing VPP with DPDK for FD.io.

About the PoC Environment

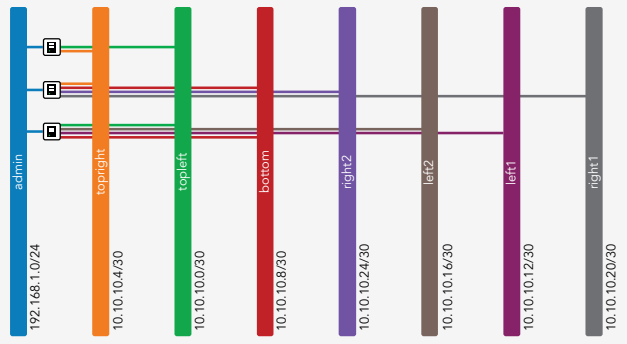
Figure 1 shows a logical topology of three virtual machines, representative of a VNF data plane, which was used as the VNF model. The stated objective for this PoC was to accomplish 40-Gbps packet-processing levels for the overall VNF using the four 10G flows shown. Cisco UCS® servers, Intel® Xeon® CPUs, Intel Fortville Ethernet Adapter, and Red Hat OpenStack software were used for the PoC.

Figure 1. Logical Topology



This logical topology is then instantiated in OpenStack by means of standard API calls. Figure 2 shows the resulting virtual topology as represented from the OpenStack Horizon GUI.

Figure 2. Logical Topology in OpenStack GUI



OpenStack schedules the virtual machines as usual, and they are deployed across the available computing resources. In this case, the virtual machines are spread amongst the two Cisco UCS servers—Router-Left and Router-Top are placed on Cisco UCS#1, Router-Right is placed on Cisco UCS#2.

Figure 3. Physical Topology

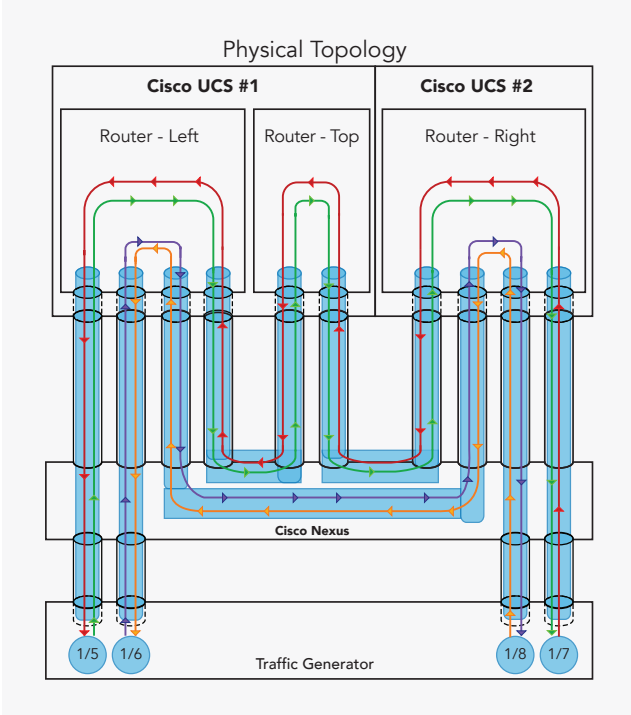


Figure 3 depicts the physical topology of the PoC environment and the resulting placement of the virtual machines in the sample VNF.

SDN in the PoC Environment

The Cisco Virtual Topology System (VTS) is a standards-based, open, overlay management, and provisioning system for data center networks. It automates fabric provisioning for both physical and virtual workloads. VTS enables the creation of a highly scalable, open-standards–based, multitenant solution for service providers and large enterprises. It automates complex network overlay provisioning and management tasks through integration with cloud orchestration systems, such as OpenStack. The solution can be managed from the embedded GUI or entirely by a set of northbound REST APIs that can be consumed by orchestration and cloud management systems.

To use complete PCI passthrough, the packet flow must be controlled from each NIC port to ensure that it reaches other NICs on the same virtual network and is isolated from other networks. The VTS was utilized in this PoC to provision the Cisco Nexus® 9372 switch to make the appropriate connections.

It also provisions virtual ports on hosts, where appropriate (in the PoC this is used for virtual machine control interfaces) and ensures interoperation of the two types of ports.

Use of VTS is smooth in the OpenStack environment, and no special awareness of the SDN controller in use is necessary: the VNF orchestration uses the standard Neutron APIs.

PoC Methodology

The topology is constructed using standard OpenStack API calls, and thus the virtual machine and networking orchestration takes place automatically. The Spirent is used to generate four separate unidirectional flows through the resulting topology, as illustrated in Figures 1 and 3.

PoC Results and Findings

Outcome of POC: 4 x 10 Gbps (40 Gbps) per virtual machine, both at IMIX and very small packet sizes, zero drop performance in a 12 hour test with latency below 0.1ms within the system and low jitter.

Using a constant traffic stream with the Spirent TCPv4 IMIX profile[1], we observed:

Bandwidth	10 Gbit/sec per stream
Packets per Second	2.75 Mpps per stream
Latency	80 µsec average, 3 virtual machine path 56 µsec average, 2 virtual machine path[3]
Jitter[2]	0.0049 µsec absolute average 75 µsec maximum[3]
Drops	0 packets lost[4]

In addition to the 40 Gbps per virtual machine achievement, the PoC also demonstrates a single Cisco UCS server processing line rate at 6 x10G–60 Gbps bidirectional or 120 Gbps total. Further experimentation revealed that the zero drop rate can be maintained at packet sizes down to 170 bytes, a rate of 6.5 Mpps per stream.

Conclusion

The collaboration between Cisco and Intel has shown that the promise of a high-performance VNF on NFVI is achievable. The PoC demonstrates a real-world, multiple virtual-machine router topology with 40-Gbps throughput per virtual machine, fully orchestrated, using both physical network devices and OpenStack.

Cisco and Intel have collaborated for more than a decade not just on constructing products but also working together on open standards and open source. Examples include DPDK implementation across a variety of Cisco platforms and collaboration in OpenStack. When it comes to high-performance NFV, the Cisco NFVI architecture coupled with Cisco VPP technologies, infused with Intel EPA and DPDK enhancements, displays just what is possible in the evolution of NFV. This true partnership derives from how Cisco products consume Intel innovation and the joint focus and contributions to open source components that push the ultimate value in NFV forward: optimal service creation.

Additional Sources

For more information, visit the following:

- Cisco NFV Infrastructure: http://cisco.com/go/nfvi
- Network Functions Virtualization: http://cisco.com/go/nfv
- Cisco Virtual Topology System: http://cisco.com/go/vts
- Technology Brief: "Smarter CPU Pinning in OpenStack Nova": https://networkbuilders.intel.com/docs/CPU_Pinning_With_OpenStack_nova.pdf
- White Paper: "Data Center Automation for Next-Generation Cloud Architectures": http://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/virtual-topology-system/white-paper-c11-734904.html

[1] Spirent TCPv4 IMIX profile packet mix as follows: 58.67 percent 72 byte, 2 percent 74 byte, 23.66 percent 576 byte, 15.67 percent 1500 byte
[2] Jitter measurement per RFC 4689
[3] 5 minute test
[4] 12 hour test

Configuration Details

Feature/Platform	Version/Details	Comments
Cisco UCS 240 M4	2 Intel Xeon E5-2630L v3 8 core, 1.8 Ghz CPUs per socket, 256 GB memory	
Memory	16 Gb DDR4 DIMMS, 1866 MHz	
Intel Ethernet Converged Network Adapter NIC	X710 DA4	Intel Ethernet Converged Network Adapter NIC
Cisco Nexus Switch	9372PX	Cisco Nexus switch
Spirent	TestCenter 4.50.5906.0000	TCPv4 IMIX profile. Driving an SPT-N4U chassis containing an 8-port FX2 card (4 ports used)
Cisco VTS		VTS
OpenStack	RHEL OSP 6	Juno
OS	RHEL OS 7.1	
QEMU	2.1.2	qemu-kvm-rhev-2.1.2-23.el7_1.9
BIOS	C240M4.2.0.3c.0.091920142008	stock; none changed for performance tuning, power management enabled
Kernel	3.10.0-327.4.5.el7.x86_64 #1	