SR-IOV for NFV Solutions

Practical Considerations and Thoughts

Technical Brief

Networking Division (ND)

February 2017

Authors:

Patrick Kutch

Cloud Solutions Architect
Solutions Enabling Team, Intel

Brian Johnson

Solutions Architect
Networking Division, Intel

Revision 1.0
335625-001

Revision History

Revision	Date	Comments
1.0	February 23, 2017	Initial release (Intel public).

Contents

NOTE: This page intentionally left blank.

1.0 Introduction

Ethernet Virtualization has evolved considerably since 2008, in relation to Virtual Machine Device Queues (VMDQ) and Single Root I/O Virtualization (SR-IOV) technologies. Previously, only one hypervisor (VMware ESX) supported VMDQ, its use requiring some customization. Support for SR-IOV was essentially an experimental technology available in Linux and only available on a few server platforms, with Intel® providing the only two Ethernet devices that supported it.

Today, VMDQ and similar technology is available in many Ethernet vendors' products and enjoys broad support and default behavior within most Operating Systems and hypervisors. SR-IOV also has broad ecosystem support.

We have now reached a point where the raw processing power provided by the latest generation of Intel® Xeon® processors, in conjunction with ever-evolving platform and Ethernet Virtualization technologies, has given rise to Software Defined Networking (SDN) and Network Functions Virtualization (NFV) technologies.

Utilizing the flexibility provided by SDN, traffic can be routed to different Virtualized Network Functions (VNFs) to perform any number of Network Function workloads, such as load balancing, routing, deep packet inspection, and so on.

SR-IOV would seem to be an excellent technology to use for a NFV deployment; using one or more SR-IOV Virtual Functions (VFs) in a VNF Virtual Machine (VM) or container provides the best performance with the least overhead (by bypassing the hypervisor vSwitch when using SR-IOV).

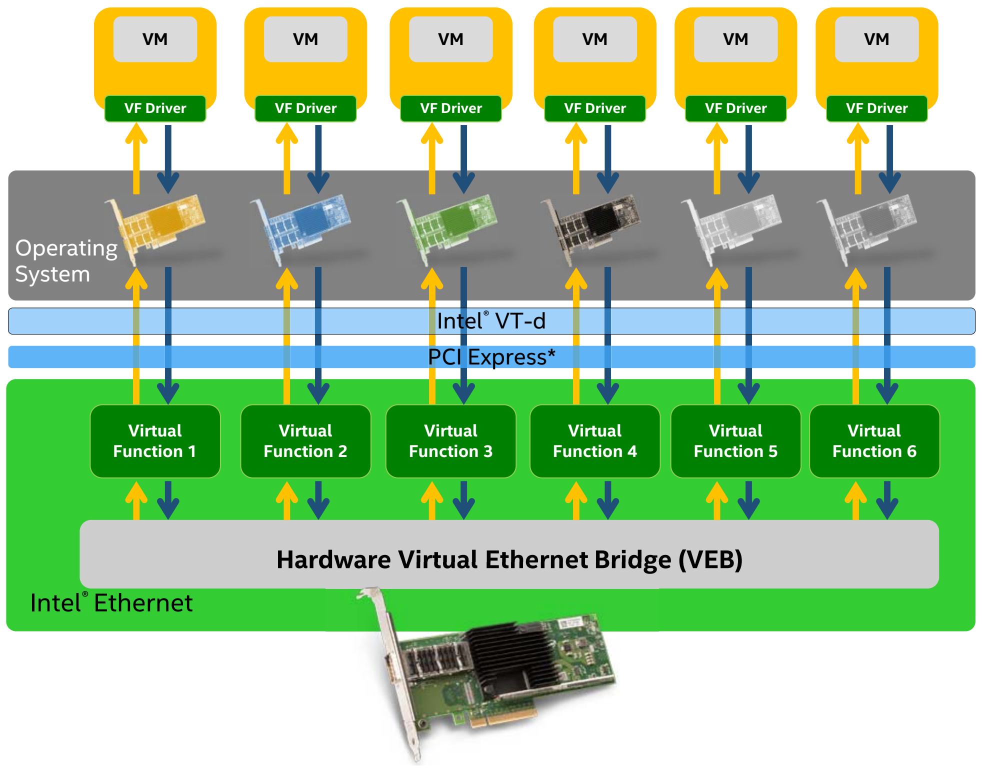

Figure 1. SR-IOV Virtual Functions

There are a number of published articles and papers from various Ethernet vendors touting their SR-IOV solutions as being ideal for NFV; some focus on “Smart NIC” type of capabilities, others have vSwitch offloading, and others speak of raw packet performance. Some of the data is compelling enough to warrant closer examination. This paper outlines the results of that examination and experimentation.

Briefly, SR-IOV in some traffic patterns provides an excellent technology for a NFV solution. However, it can also be a poor choice in other situations. This document explores these topologies, shows what does and does not work well, and explains why.

1.1 Technologies

1.1.1 Data Plane Development Kit (DPDK)

This section provides a very high-level overview of the DPDK. For a deeper understanding, see resources available at http://dpdk.org.

The traditional mechanism by which Ethernet traffic is received and makes it into the Network Stack of an operating system is as follows:

1. Incoming traffic is moved from the Ethernet controller to ring buffers and an interrupt is asserted.
2. The driver services the interrupt, receiving the packet and moving it along the network stack, where it eventually arrives in a socket.
3. The packet data is copied from the socket receive buffer to the application, which requires moving the data from Kernel Space to User Space.

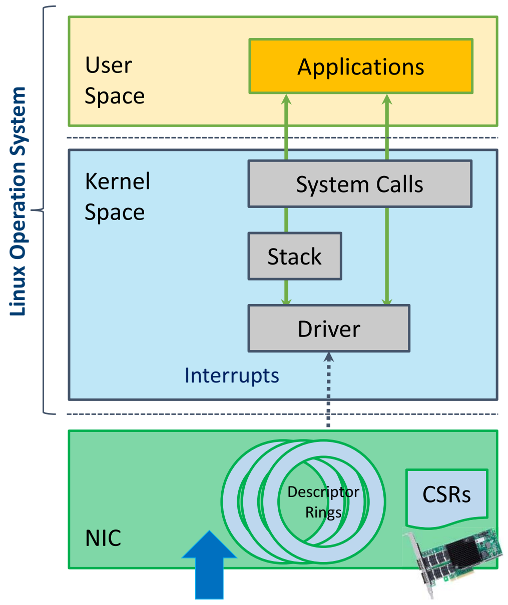

Figure 2. Traditional Packet Flow

The key points are that data is received, an interrupt is asserted, the interrupt is serviced by a CPU core, and the data is moved from Kernel Space to User Space. This model has functioned very well for decades.

Consider for a moment how the maximum line-rate of Ethernet has evolved. Around the year 2000 the 1 GbE devices moved into the market where 10/100 MbE was the norm. Since then, the industry has embraced and moved to 10 GbE and 40 GbE, with 100 GbE and higher on the horizon. All of these modern connected devices (such as smart phones, wearables, and so on) require a tremendous amount of network bandwidth for processing in the data centers.

The smaller the packet size, the more packets per second are processed. For example, on a 1 GbE connection, taking into account all of the overhead involved with Ethernet (Inter Frame Gap, MAC Preamble, and so on) at 64 bytes there can be just under 1.5 million packets per second (Mpps). On a 10 GbE connection there can be nearly 15 Mpps.

From the perspective of this traffic being handled by the Ethernet Controller and the Network Stack, that is roughly 15 million interrupts per second. Interrupts are a fairly expensive compute mechanism, as processing must be stopped on whatever is in process, the state saved, the interrupt processed, then the original process restored and continued. This overhead is one reason that dramatic differences can be seen in performance tests using **netperf** or **iperf** with 64-byte packet size versus 64-KB packet size. Larger packet size means less packets per second, less interrupts, and less of the expensive movement of data from Kernel Space to User space.

On high-speed connections, the overhead of 10's of millions of interrupts per second and moving the data definitely impacts performance.

DPDK takes a different approach to handling Ethernet traffic. Rather than relying on interrupts, DPDK polls for data that has come into the Ethernet Controller and processes it, bypassing the Network Stack. DPDK dedicates one or more CPU cores to polling and handling the data. In fact, the DPDK driver is often referred to as the DPDK Poll Mode Driver (PMD).

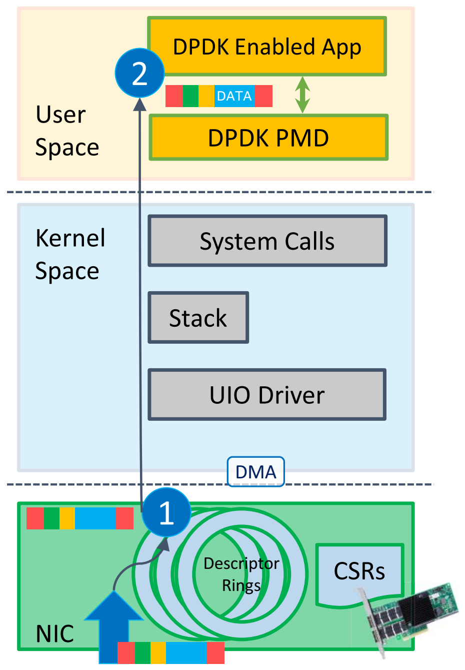

Figure 3. DPDK Overview

DPDK does many things, first and foremost it makes it so the Ethernet Controller is handled and controlled in User Space as opposed to Kernel Space, and the memory buffers the data is DMA'd into are in User Space, thus eliminating the expensive movement of data from Kernel Space to User Space. This is accomplished using the UIO driver, which allows the mapping of a device into User Space.

1.1.1.1 Pros and Cons

DPDK removes some of the bottlenecks involved with packet processing. By moving the memory buffers into User Space and doing polling rather than interrupt-based processing, DPDK is able to achieve enhanced performance with small packet sizes. DPDK is able to achieve line-rate throughput for 64-byte packet sizes - that is nearly 15 Mpps.

While DPDK provides incredible performance, it does come at a cost. Using the DPDK PMD model, the Operating System Network Stack is bypassed and the DPDK application has full control over the packets. This means that a standard web browser or FTP server cannot be run on top of a DPDK PMD. A Network Stack must built on top of those applications needing a full stack (such as a web browser or FTP server).

DPDK is actually a set of libraries used to write the DPDK application. In general, DPDK requires writing a DPDK application to process Ethernet traffic. This means that it is usually a specialized application.

A DPDK application owns the device, and as such the device cannot be shared with other applications. If the DPDK application is run on a 10 GbE port, only that application can use it.

For more detailed information on DPDK, visit http://dpdk.org.

1.1.2 Evolution of the Virtual Switch

The first several generations of Virtual Switches were by modern comparison quite simple, usually only routing traffic to VM's based upon Layer 2 information, such as MAC and VLAN. The vSwitch within a specific hypervisor was in many ways a proprietary solution to solve the problem of virtual environment Ethernet switching.

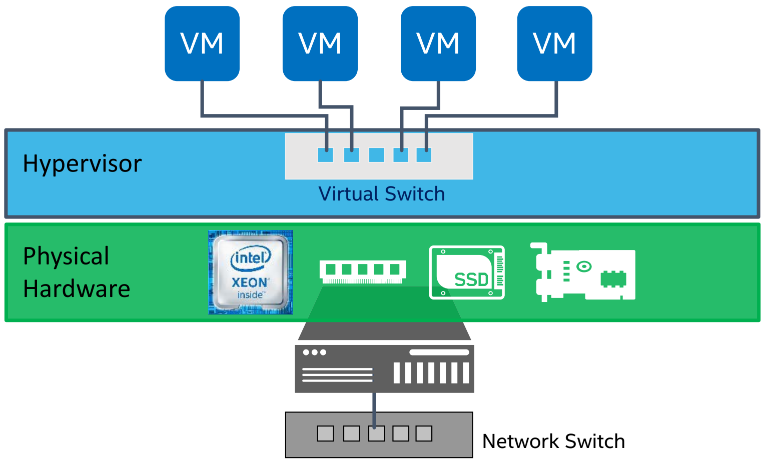

Figure 4. Virtual Switch

The modern Virtual Switch (vSwitch) has evolved into a very powerful component with a very rich set of capabilities, such as VXLAN, GRE, LACP, STP, and IPv6, with new capabilities being added all the time.

Many commercial and open source vSwitch solutions are now available. Open vSwitch (www.openvswitch.org) is arguably the most popular Open Source solution and the one used in this document.

Along with the evolution the vSwitch technology, the Ethernet interfaces used by the VMs has also evolved to include some very efficient para-virtualized solutions such as virtio-net.

1.2 Purpose of This Document

The purpose of this document is to look at some typical use cases for NFV traffic, and to examine the performance using SR-IOV versus Open vSwitch with DPDK enhancements under different conditions (namely North/South versus East/West traffic patterns, discussed in Section 2.1).

While some raw performance numbers are provided for comparison purposes throughout this document, there has been very little done in the way of optimizations. In other words, a system was set up that supports SR-IOV, Open vSwitch, and Open vSwitch with DPDK enhancements.

All three of these technologies can be adjusted to improve performance. Since to do so for one technology usually results in the degradation of another, no performance adjustments were made. The goal is not to show the best performance for any of these technologies; it is to show a side-by-side comparison with the exact same setup to promote a better understanding of the best uses and workloads for a specific technology.

2.0 Traffic Flows and Network Interfaces

2.1 Traffic Flow Patterns

There are two ways in which traffic flows in a NFV solution: North/South and East/West. In the North/South flow pattern (Figure 5), traffic is received from outside the physical server, routed to a Virtualized Network Function (VNF), processed, and sent back out to the network. An example of this is a router VNF.

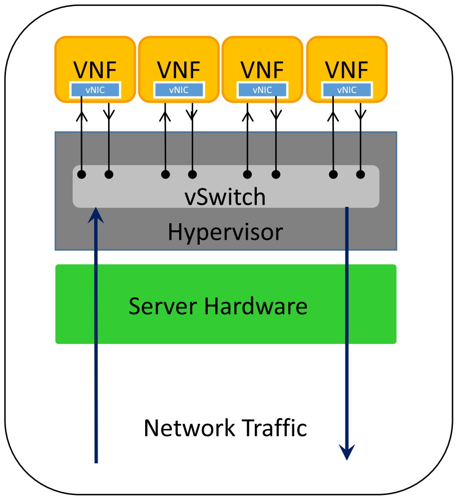

Figure 5. North/South Traffic Pattern

In the East/West flow pattern (Figure 6), traffic is processed by a VNF and sent to another VNF within the same physical server. This can be repeated a number of times wherein the traffic goes from one VNF to another, all within the same physical server. Network Service Chaining is a perfect example of this traffic flow.

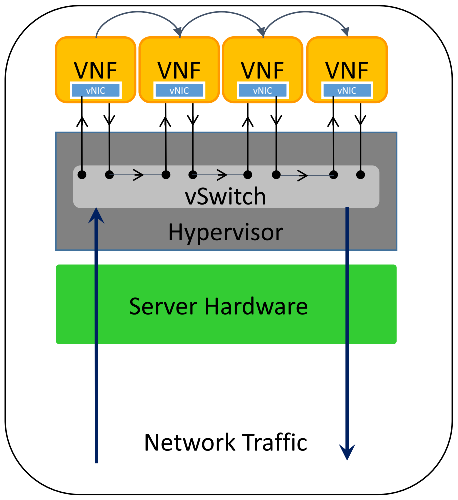

Figure 6. Ease/West Traffic Pattern

A solution could have a combination of these patterns. However, for the purposes of this document North/South traffic is separated from the East/West traffic, as there are some significant differences in their performance depending on what kind of networking technology is deployed to the VNFs.

2.2 Network Interfaces in VNFs

Each of the VMs used for testing are identical (see Section 2.4.2.1 for details), they all have three Network Interfaces used for sending over the different test traffic, and a fourth interface used for managing the VM. The following is a **lspci** listing from one of the VNF VMs:

```
ubuntu@vnf11:~$ lspci | grep Eth
00:03.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8100/8101L/8139 PCI Fast
Ethernet Adapter (rev 20)
00:04.0 Ethernet controller: Red Hat, Inc Virtio network device
00:08.0 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)
00:09.0 Ethernet controller: Red Hat, Inc Virtio network device
```

The first device listed is the management interface that is connected to a standard Linux bridge - it is not connected to OVS. This is also the interface used to send the collected Key Performance Indicator (KPI) data.

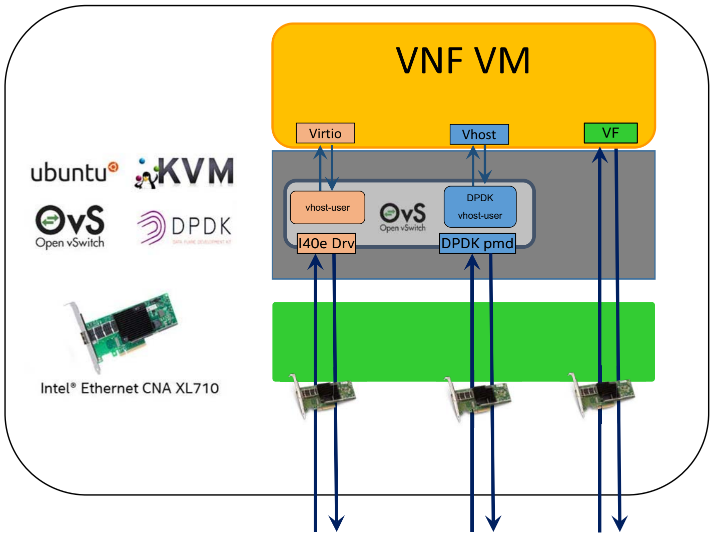

Figure 7. VNF Network Interfaces

All three interfaces are used to do side-by-side comparisons of performance over various conditions.

2.2.1 DPDK-Based VHOST Interface

The second device is a virtio-net device that is backed in OVS by a DPDK enhanced VHOST-user interface. While the device within the VNF VM is actually a virtio-net device, to easily distinguish it from the other virtio-net device it is referred to as a "VHOST" device throughout this document.

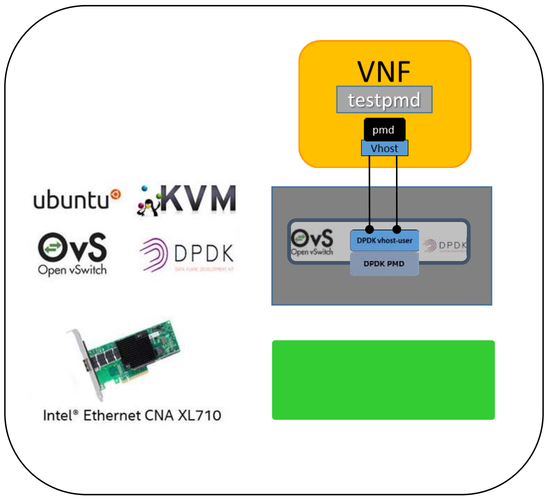

Figure 8. DPDK-Based VHOST Interface

2.2.2 SR-IOV Virtual Function Interface

The next Ethernet device in each VNF VM is a SR-IOV Virtual Function on an Intel® Ethernet CNA XL710. Ethernet traffic to and from the VNF VM bypass the hypervisor and OVS.

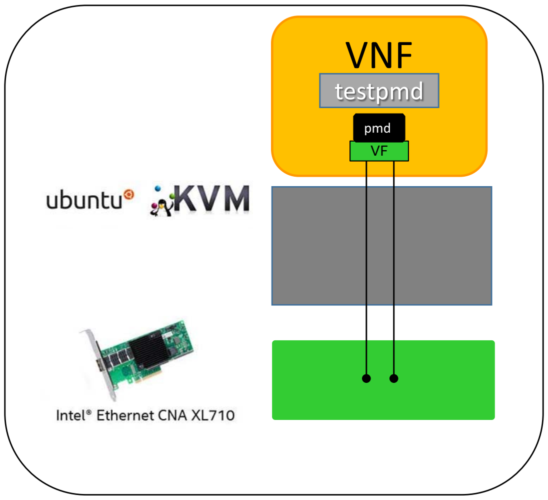

Figure 9. SR-IOV VF Interface

2.2.3 Virtio Interface

The last Ethernet interface in each of the VNF VMs is a para-virtualized virtio-net device that is backed by a VHOST-user device within OVS.

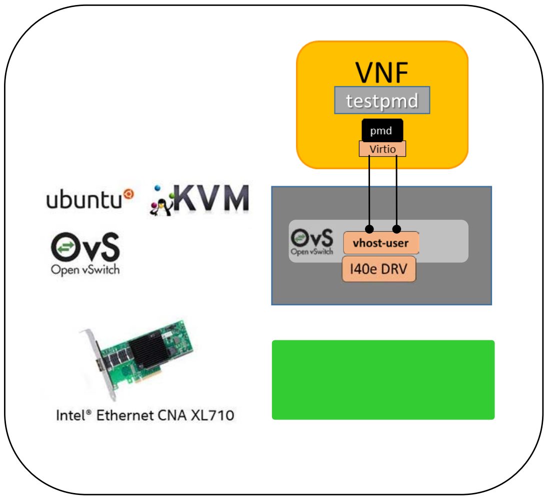

Figure 10. Virtio Interface

2.3 Network "Stacks"

The majority of the testing was performed using a DPDK application within the VMs. Specifically, **testpmd** was used to read packets in and write them back out to simulate a VNF application.

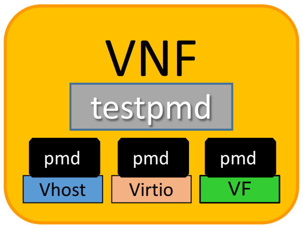

Figure 11. VNF Virtual Machine

For North/South traffic testing, **testpmd** changes the destination MAC Address of the received Ethernet packet to be that of the link partner external to the physical server. With East/West traffic, **testpmd** inserts the MAC Address of the next VM in the East/West "chain" as the destination MAC Address.

2.3.1 testpmd

The DPDK **testpmd** application acts as a NFV application. While it is useful and robust DPDK application, it is not optimized, and there is some packet loss at high packet rates.

For example, when performing SR-IOV North/South testing, the a single VNF VM receives approximately 17.4 Gbps, yet only transmits 9.8 Gbps. Two VNF VMs utilizing the SR-IOV interface each receive 8.5 Gbps and transmit 7.9 Gbps.

Note: This performance could likely be improved by assigning more cores to the **testpmd** PMD for transmitting and receiving traffic.

For this testing only the data rate and packets per second that each VNF VM transmits were recorded, at small packet sizes (in particular 64 bytes) and few VNF VMs. The results shown throughout this document may be less than what could be achieved with an optimized application and configuration.

2.3.2 Traditional Stack

As a contrast, there is also a dedicated VM for testing the performance of a "standard" Ethernet stack using **netperf** over each of the Network Interfaces.

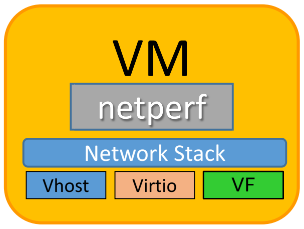

Figure 12. Traditional (Standard) Network Stack

The test setup has nine identical VMs. Eight are set up to use the DPDK **testpmd** application, while the last is identical to the other eight used in testing. The only difference is that instead of using the DPDK **testpmd** application and simulating a VNF application, **netperf** is run using the traditional Network Stack.

This is to provide a clear contrast to the performance of the three interfaces when using a DPDK application versus using a traditional network application.

```
netperf -H $otherSys $timeToRun $size
```

2.4 Testing Topology

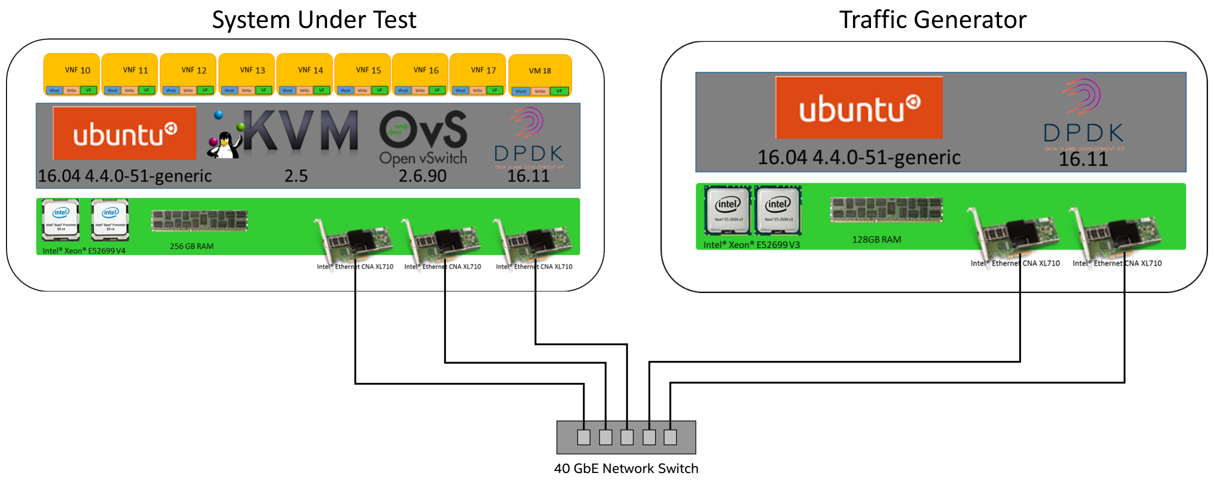

Figure 13. System Setups

2.4.1 Traffic Generator

- Dell* PowerEdge R730, 2 Intel® Xeon® E5-2699 v3 @ 2.30 GHz, 128 GB RAM
- Intel® Ethernet CNA XL710 (40 GbE)
  - Netperf
  - DPDK **pktgen**
- Ubuntu 16.04
- DPDK **pktgen** 1.13
- Netperf 2.7.0
- Minion 16.12.17
  - KPI Gathering
    - Rx/Tx Gbps
    - Rx/Tx Packets per second

2.4.2 System Under Test

- Dell* PowerEdge R730, 2 Intel® Xeon® E5-2699 v4 @ 2.20 GHz, 256 GB RAM
- Intel® Ethernet CNA XL710 (40 GbE)
  - Open vSwitch
  - Open vSwitch with DPDK enhancements
  - SR-IOV
- Ubuntu 16.04, KVM 2.5
- Open vSwitch 2.6.90
  - DPDK 16.11
- Minion 16.12.17
  - KPI Gathering
    - CPU Utilization

2.4.2.1 Virtual Machines

- Ubuntu 16.04, 3 GB RAM
- virtio-net Interface
  - Standard OVS interface (see Section 2.2.3)
- SR-IOV VF
  - Bypasses hypervisor and OVS (see Section 2.2.2)
- DPDK-based VHOST Interface
  - DPDK enhanced OVS interface (see Section 2.2.1)
- DPDK **testpmd** 16.07 or **netperf** 2.7.0
  - DPDK testpmd for VNF11–VNF17; netperf for VM18

- Minion 16.12.17
  - KPI Gathering
    - CPU Utilization
    - Rx/Tx Gbps
    - Rx/Tx Packets per second

2.5 Testing Methodology

Nine identical VMs were created. Eight of them were used for comparing DPDK workloads, and one was used for the traditional Network Stack. For all comparisons, whether they are using DPDK or traditional Network Stack, six different packet sizes were used and a number of KPIs collected, the most important ones being data and packet rates, reported in Gigabits per second (Gbps) and millions of packets per second (Mpps), respectively.

The packet sizes used in the tests are:

- 64 bytes
- 128 bytes
- 256 bytes
- 512 bytes
- 1024 bytes
- 1518 bytes

The packet size is inversely proportional to the packets per second. The smaller the packets, the more packets per second. For example, under one test (SR-IOV North/South) a single VNF/VM can process 64-byte packets at approximately 10 Gbps and more than 20 Mpps.

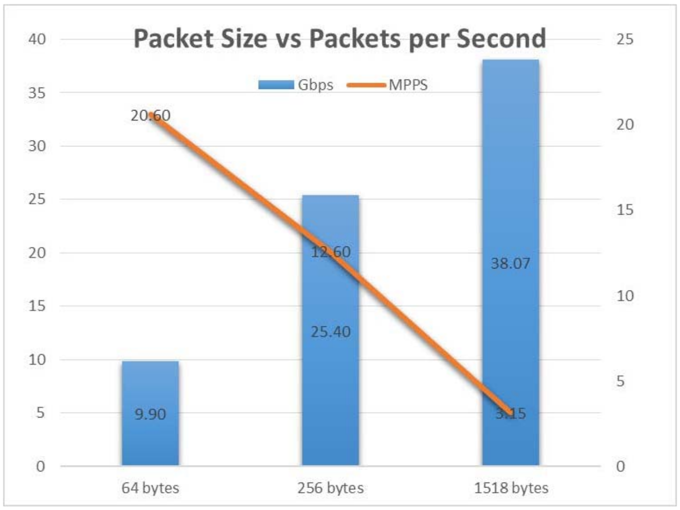

Figure 14. Sample Rates

The same test run with 1518-byte packets results in over 38 Gbps, but only slightly over 3 Mpps. This is normal and expected behavior, and is also why most network performance numbers advertised by network device vendors use large (usually 64 KB) packet sizes, as it always provides the greatest Gbps numbers at the lowest CPU utilization due to fewer packets per second.

2.5.1 Recorded Data Values

This section explains the method used in collecting the data provided in this document. Figure 15 shows an example of a GUI created using the BIFF Instrumentation Framework. BIFF is a framework by which data can be collected from a system under test and displayed in a flexible GUI.

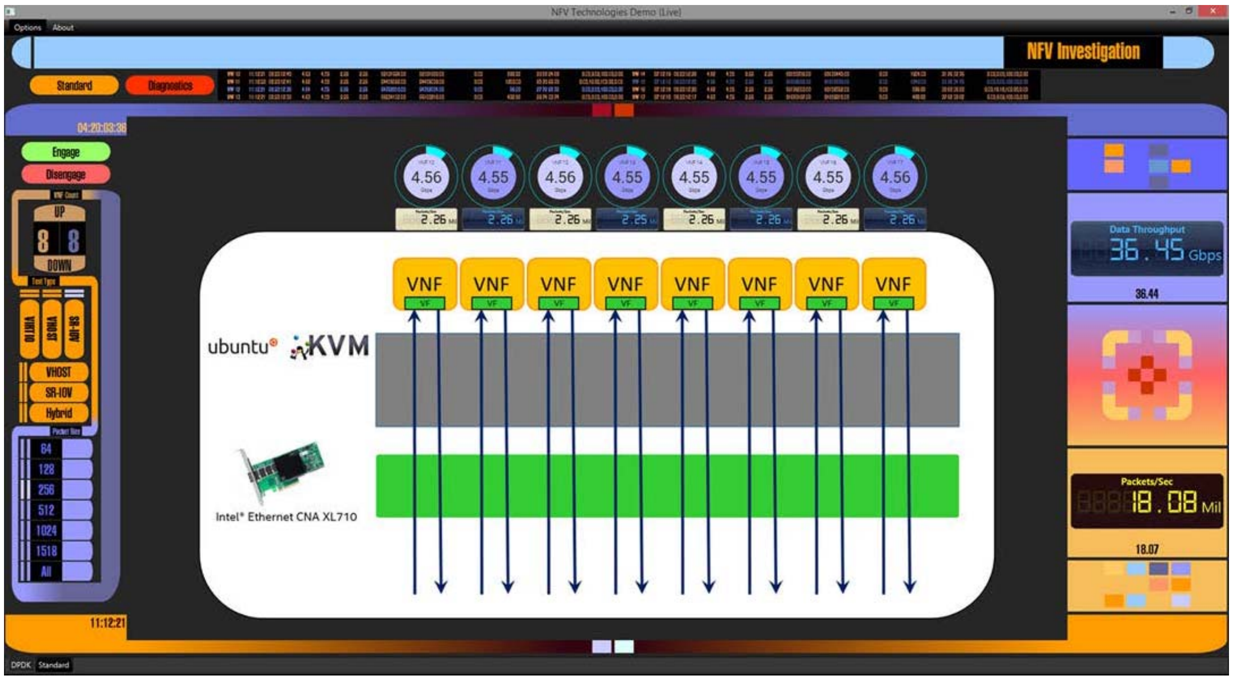

Figure 15. Screen Shot of Test UI

This GUI displays a number of KPIs gathered from the individual VNF VMs, the system that those VMs reside on, and the system running DPDK **pktgen**. On the right side of the GUI are "widgets" that show the total Gbps throughput and the packets per second being processed by all of the VMs running. As the image shows, there are eight VNFs running, each using SR-IOV, with the packet size of 256 bytes. The resulting total throughput is 36.45 Gbps and 18.08 Mpps.

To gather the data recorded for this document, a given test was started. After allowing a short waiting period for normalization, the total Gbps and Mpps were recorded. These numbers do fluctuate slightly during the tests, but in general are reasonably constant.

Note: All numbers used for the KPIs are on the Transmit side of the VNF VM only. The Receive side was also instrumented, and the data could easily be gathered. However, for the purposes of this document (discussing the use of SR-IOV for different NFV traffic patterns) providing the Transmit data was sufficient to illustrate the performance.

3.0 Results

This section presents and explains the results of the testing.

3.1 Traditional Traffic

In an attempt to provide as full of a picture as possible, one VM was set up for only doing **netperf** traffic utilizing the standard Linux Network Stack. Everything in this VM is identical to the other eight VMs, except that **netperf** was used to send data to the remote Transmit System rather than reading the data with **testpmd** and sending it back.

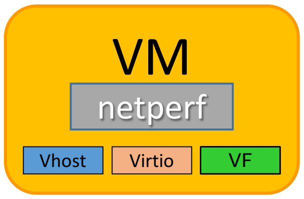

Figure 16. netperf Virtual Machine

For this test four concurrent sessions of **netperf** were run inside of the VM, with the target being the transmit system. Each test was ran serially over each of the three interfaces (Virtio, DPDK-based VHOST, and SR-IOV). In addition, the size was set using the **netperf -m** option.

The exact usage was:

```
netperf -H TransmitBox.intel.com -l 6000 -- -m SIZE
```

where the sizes were 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes, 1518 bytes, and 64 KB. Each of the interfaces were assigned IP Addresses on different subnets that had a corresponding IP Address assigned to the Intel® Ethernet CNA XL710 on the Transmit System.

The results of the testing are shown in Figure 17:

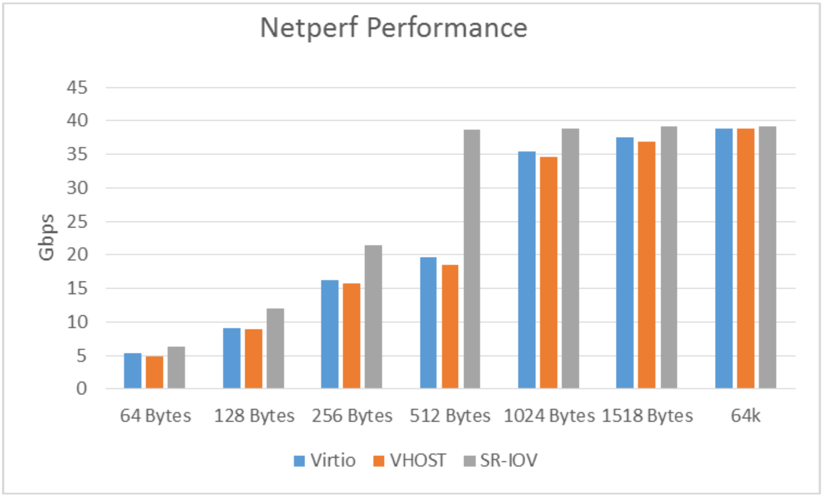

Figure 17. netperf Performance

The para-virtualized Virtio interface and the DPDK-backed VHOST interface performed well - in most cases nearly as well as the SR-IOV interface. This speaks to the performance enhancements made in vSwitch technologies, in this case the performance of OVS. Data from similar tests performed in 2008 revealed that the maximum throughput at 64 KB packets using the emulated path was less than 3 Gbps over a 10 GbE connection.

3.1.1 Performance Dependencies

During the long process (6+ months) of setting up the systems used for testing, and learning about DPDK and OVS, processor upgrades were made along the way (to gain more CPU cores for DPDK workloads), and new versions of both DPDK and OVS were released. The original data collection, using the previous generation processor (Intel® Xeon® CPU E5-2699 v3 @ 2.30GHz) and previous versions of OVS and DPDK was notably different than the data obtained with the newer versions.

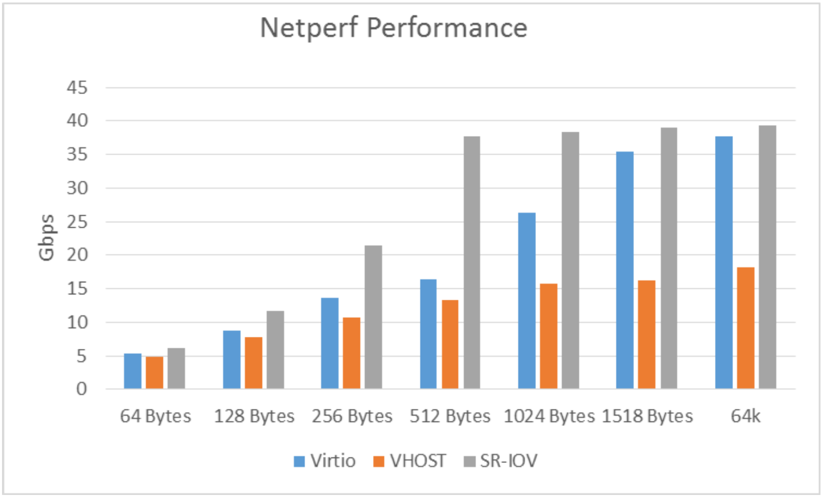

Figure 18. netperf Performance with Previous Generation Components

A comparison of Figure 17 and Figure 18 shows that there was an improvement in both the standard OVS and DPDK enhanced OVS paths. This was likely partially due to improvements in OVS and an updated CPU.

The dramatic difference between the performance on the "old" components verses the newer ones is on the DPDK-backed VHOST interface; it went from being significantly less performant than the other emulated path (Virtio) to having nearly the same performance and in some cases equal performance.

This performance change can be directly attributed to the continual improvements being made in the DPDK. In this case, the big improvement was most likely due to the new DPDK PMD enabling more intelligent offloads provided by the Intel® Ethernet CNA XL710. This is a terrific example of continual improvements being made in DPDK.

3.2 DPDK

For all the DPDK testing, the DPDK application **pktgen** generates traffic at a specific size and sends it to one or more destination VNFs located on the system under test.

3.2.1 North/South Traffic Pattern Tests

This test involves sending traffic from the transmit system using the DPDK **pktgen** application to one or more VNF VMs, and at different packet sizes. This test utilizes up to eight VNF VMs.

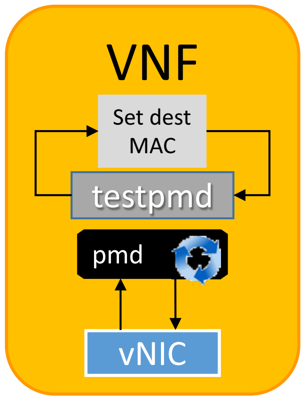

Figure 19. VNF Running testpmd

Each VNF VM is identical, all running the DPDK **testpmd** application, which receives the packets sent from DPDK **pktgen** on the interface under test (DPDK-based VHOST, Virtio, or SR-IOV VF), inserts the MAC Address of the Intel® Ethernet CNA XL710 used in the transmit system, and then writes the packet back out to the interface under test.

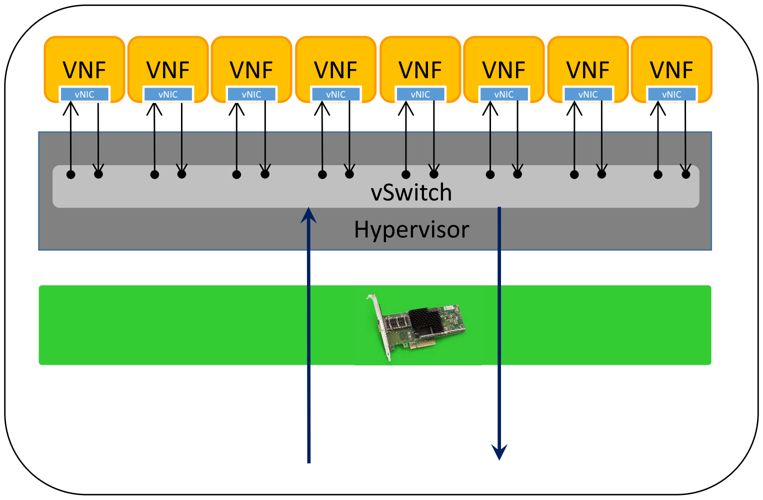

Figure 20. North/South Traffic Pattern

3.2.1.1 Virtio

This test setup uses the Virtio device within the VNF VMs to receive traffic from an external system over a 40 Gbps connection. The DPDK **testpmd** application running in each of the VNF VMs receives the traffic and sends it back to the external system by inserting the MAC Address of the external system into the destination MAC Address field of the Ethernet packets.

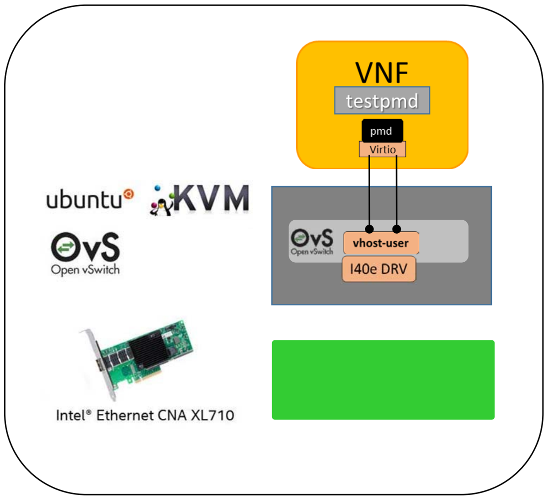

Figure 21. Virtio Test Topology

Recorded Data:

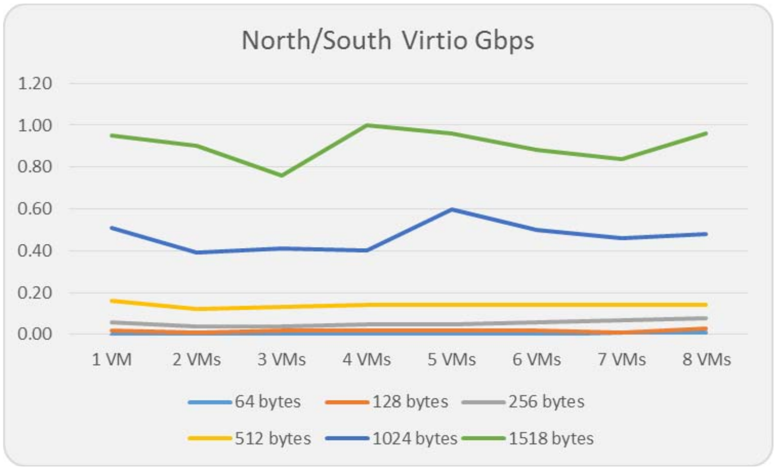

Figure 22. Virtio Performance - North/South Gigabits per Second

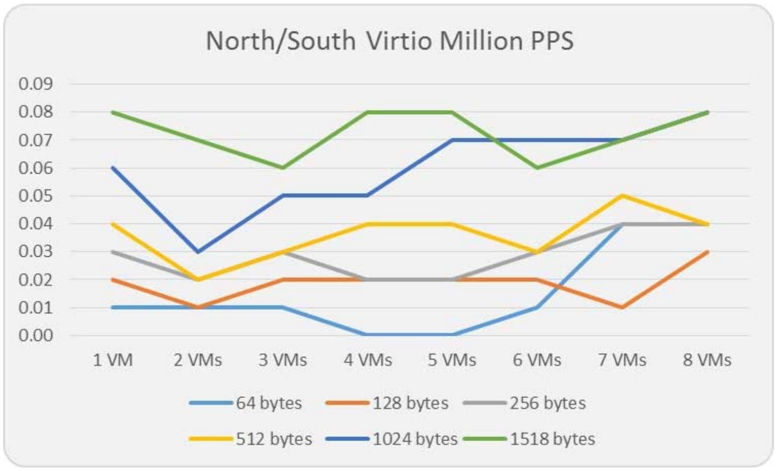

Figure 23. Virtio Performance - North/South Packets per Second

Analysis:

As shown in Figure 22 and Figure 23, the data rate only reached 1 Gbps in a single test, most were significantly less, and some were so small that they were not within the precision of the data collection method. Never once was one million packets per second processing achieved.

The reason for this is that Virtio devices are para-virtualized and are backed by a component within the hypervisor itself. This kind of para-virtualization lends itself nicely to a traditional network stack (as shown by the results in Section 3.1). However, it does not work well with a DPDK application that is polling for data.

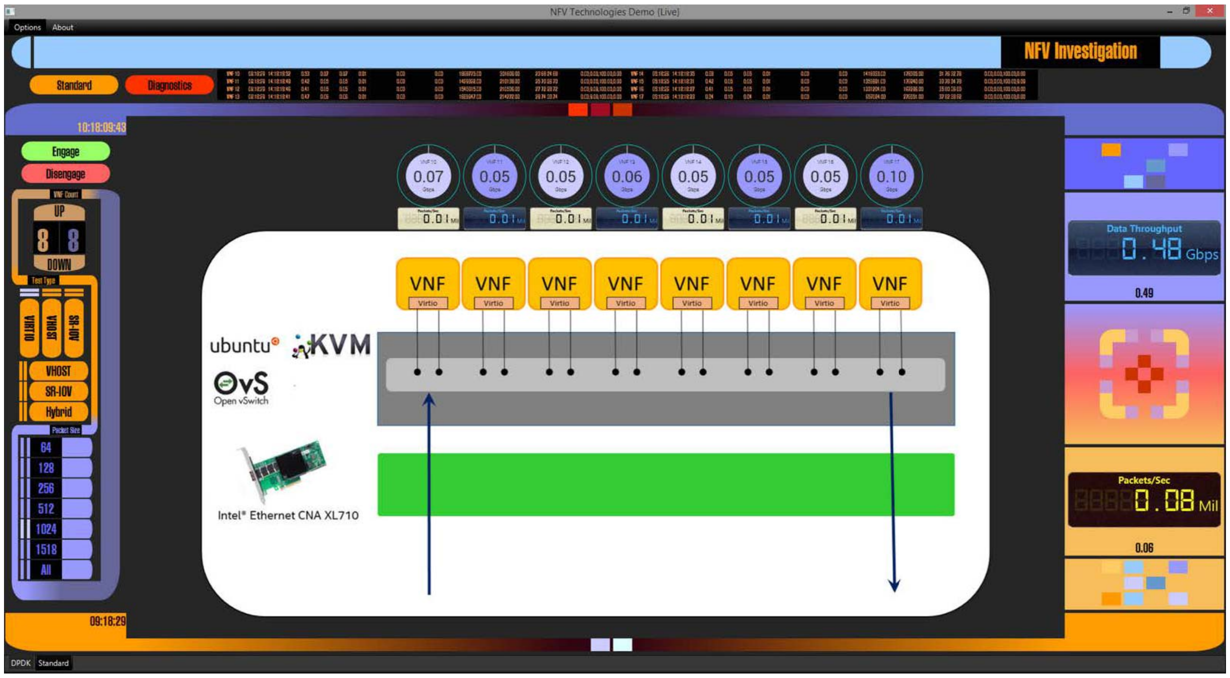

Figure 24. Virtio Performance - Eight VMs at 1024 Bytes

3.2.1.2 DPDK-Backed VHOST

This test setup uses the VHOST device within the VNF VMs to receive traffic from an external system over a 40 Gbps connection via the DPDK enhanced OVS.

The DPDK **testpmd** application receives the traffic and sends it back to the external system by inserting the MAC Address of the external system into the destination MAC Address field of the Ethernet packets.

Note: Recall that it is referred to as a "VHOST" interface to distinguish it from the Virtio interface that is not backed by a DPDK device in the hypervisor. In fact, the device within the VNF VM is a para-virtualized virtio-net interface that is backed in the hypervisor by a DPDK VHOST-user device.

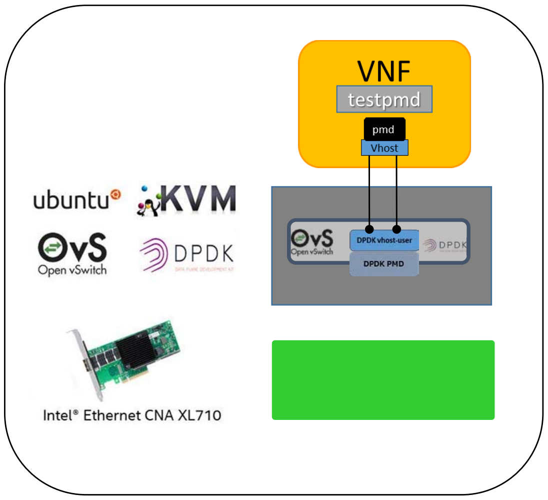

Figure 25. VHOST Test Topology

Recorded Data:

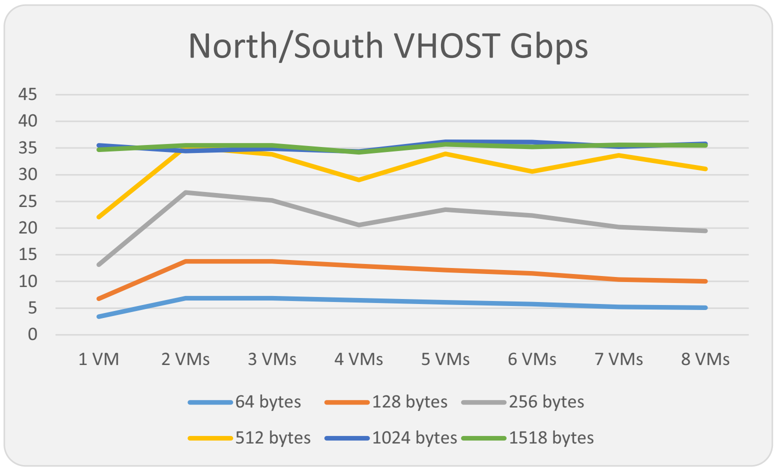

Figure 26. VHOST Performance - North/South Gigabits per Second

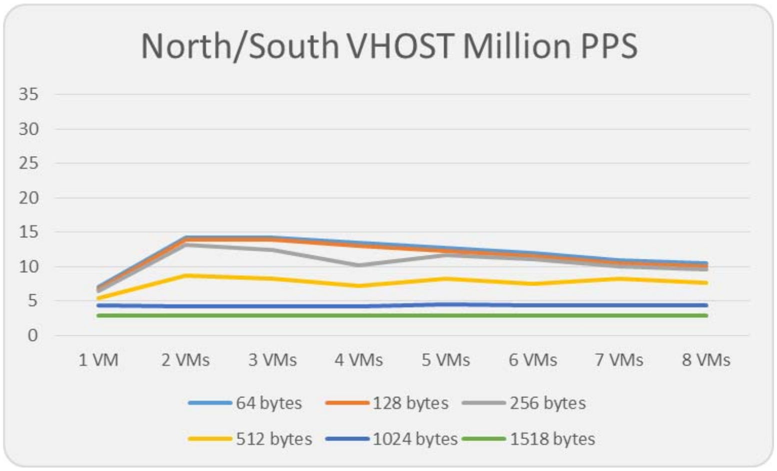

Figure 27. VHOST Performance - North/South Packets per Second

Analysis:

Figure 27 shows that the max number of packets a single VNF VM can process is around 7 Mpps with 64-byte packets. This scales to around 14 Mpps when a second VNF VM is added.

After more than two VNF VMs are in use, the total number of packets per second processed drops slightly. This occurs because of the overhead incurred in processing and classifying all of the incoming data.

DPDK enhanced OVS has made tremendous progress in processing/classifying Ethernet traffic and moving it efficiently throughout the system, especially when there is a DPDK application such as **testpmd** consuming that traffic, as in these tests. That being said, each packet sent and received to a VNF VM must still be processed three times: once by the PMD running in the VNF VM via **testpmd**, once by the PMD running within the DPDK enhanced OVS, and once by OVS itself. Due to this, there is an inherent limit in how many packets can be processed in a second.

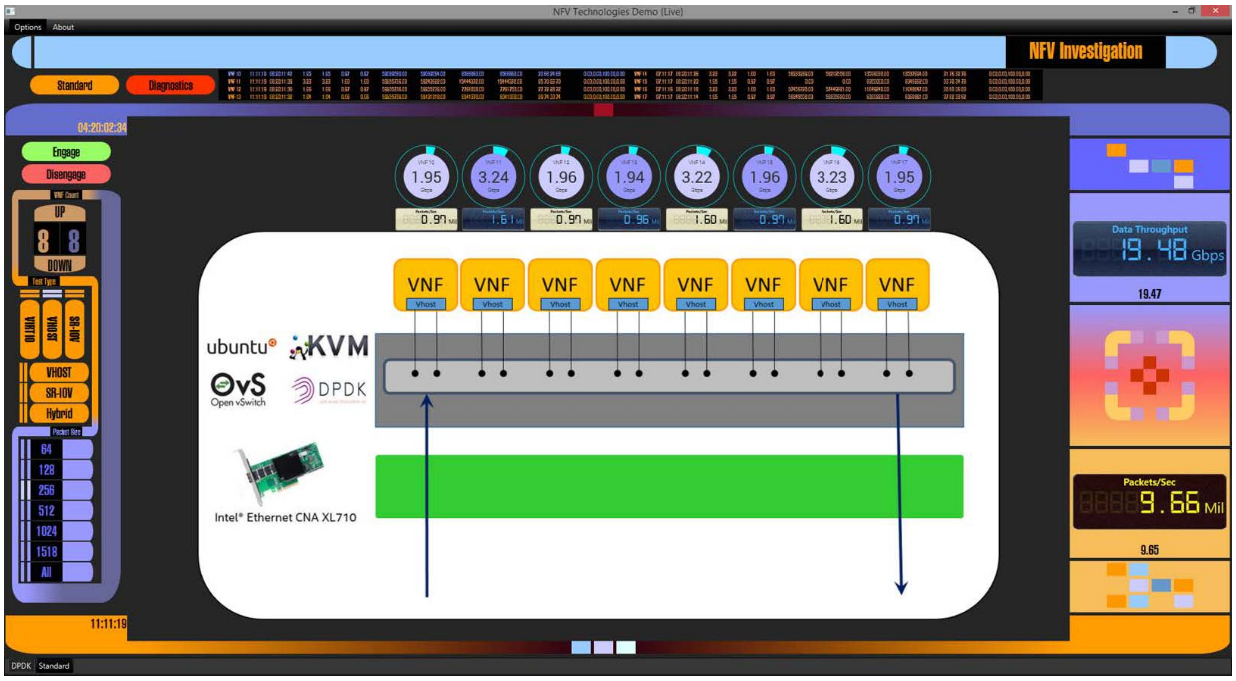

Figure 28. Screen Shot of VHOST Performance

3.2.1.3 SR-IOV Virtual Function

For this test, the Intel® i40e VF assigned to the VNF VM is utilized. Being a SR-IOV device, the movement of packets to and from the VNF bypasses the hypervisor and the vSwitch (OVS). As with the other tests, the DPDK **testpmd** application receives the Ethernet frames and writes them back with the destination MAC Address set to that of the system running **pktgen**.

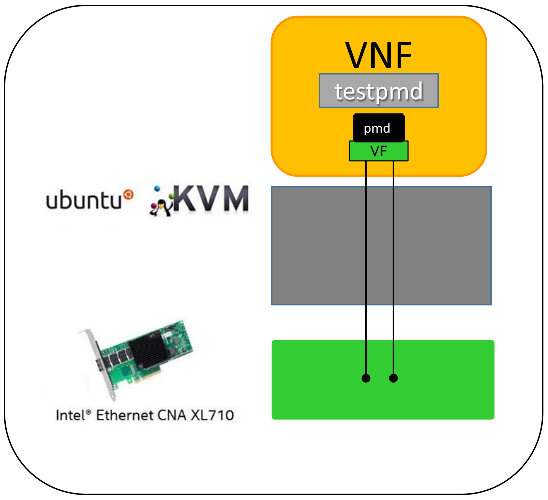

Figure 29. SR-IOV Test Topology

Recorded Data:

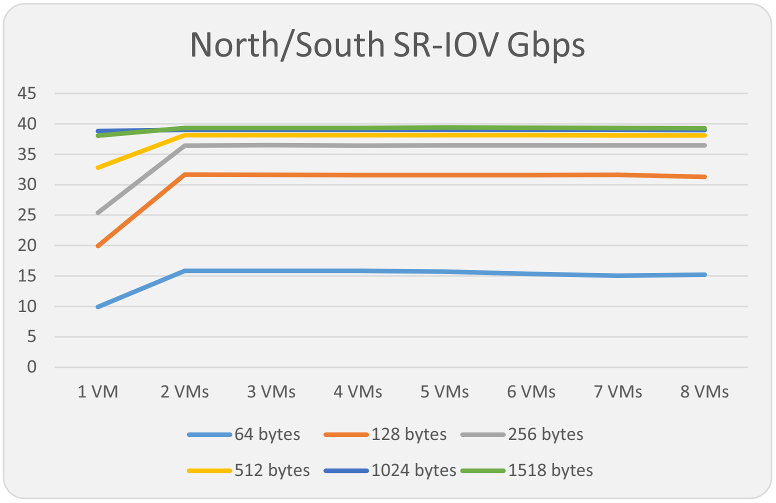

Figure 30. SR-IOV Performance - North/South Gigabits per Second

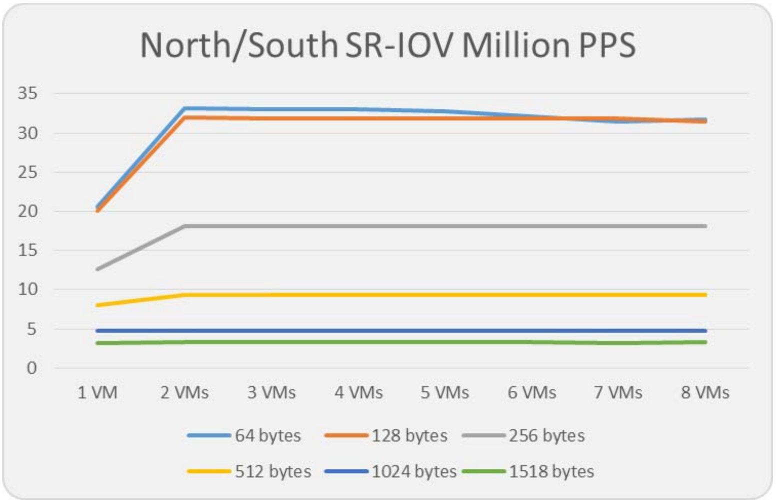

Figure 31. SR-IOV Performance - North/South Packets per Second

Analysis:

Figure 30 and Figure 31 reveal an increase in performance from a single VNF VM to two or more. After two VNF VMs are part of the test, performance is very consistent and scales extremely well.

Note: More than 30 Mpps was achieved in a single VNF VM using 64 bytes and SR-IOV. However, it required adjustments and optimizations. As a reminder, the testing was specifically conducted in a "general setup" with small performance adjustments that would be specific to one type of test.

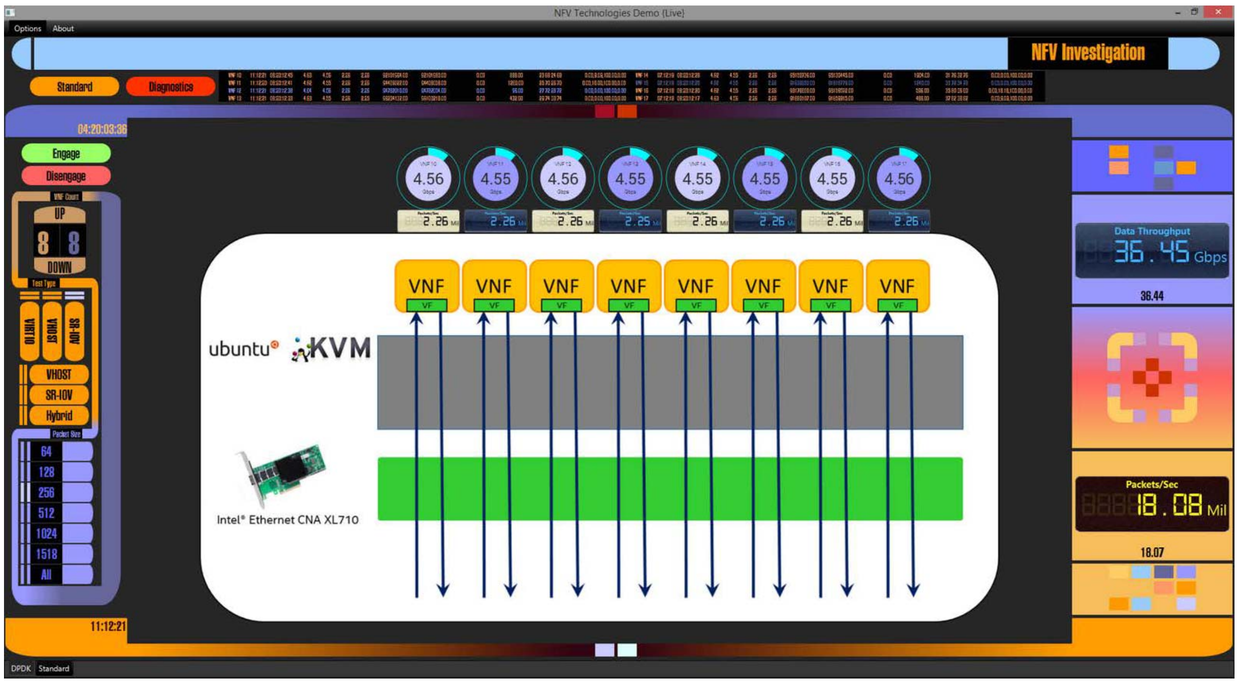

Figure 32. Screen Shot of SR-IOV North/South Test

3.2.2 East/West Traffic Pattern Tests

For this series of tests, the remote system running the DPDK **pktgen** application sends traffic to the first VNF VM in the chain and is received by the DPDK **testpmd** application, which in turn sends the traffic to the next VNF VM in the chain by changing the destination MAC Address in the packet to that of the next VNF VM in the chain.

This chain continues until the last VNF VM in the chain, in which the **testpmd** application inserts the MAC Address of the system running the DPDK **pktgen** application into the destination MAC Address field.

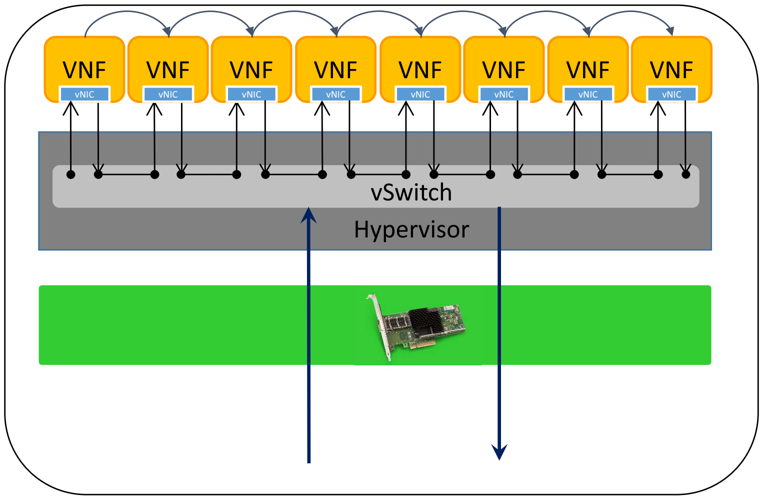

Figure 33. East/West Traffic Pattern

The performance when using the Virtio interface (see Section 3.2.1.1) was so poor that the recorded data and analysis are not included in this section. Instead, more focus is given to the VHOST and SR-IOV VF interfaces.

3.2.2.1 DPDK-Backed VHOST

This test uses two to eight VNF VMs. The first VNF VM receives traffic over the VHOST interface from the remote transmit system. The DPDK **testpmd** application receives the traffic, inserts the MAC Address of the VHOST interface in the VNF in the chain, and then writes the data. This continues along the chain until the final VNF VM, where the destination MAC Address written into the packet by **testpmd** is the MAC Address on the system transmitting data using **pktgen**.

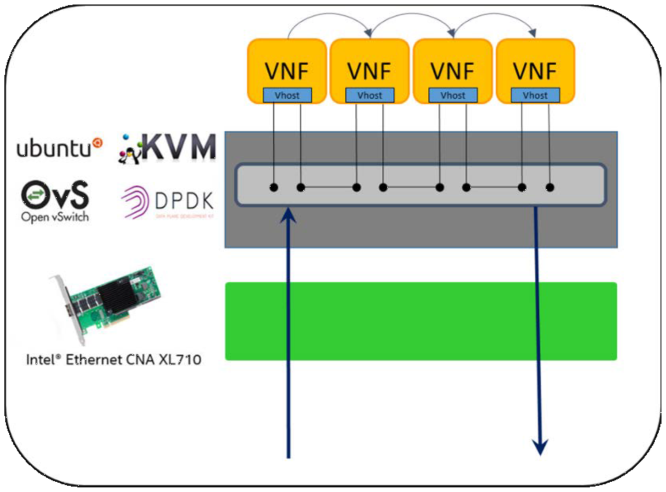

Figure 34. VHOST East/West Test Topology

Recorded Data:

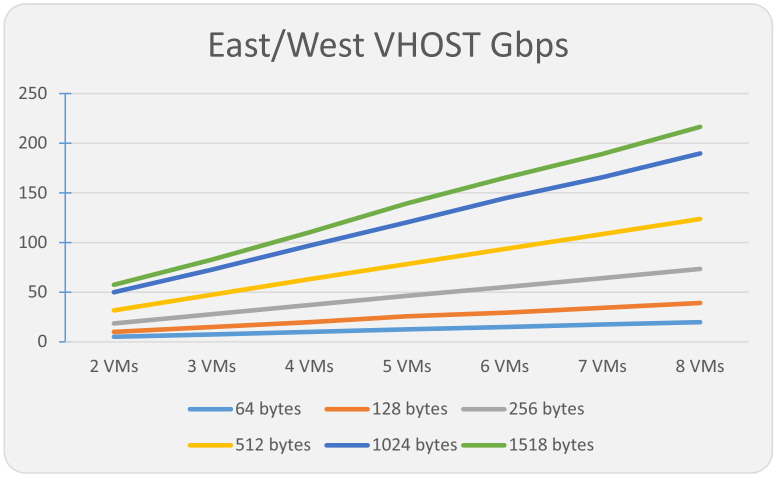

Figure 35. VHOST Performance - East/West Gigabits per Second

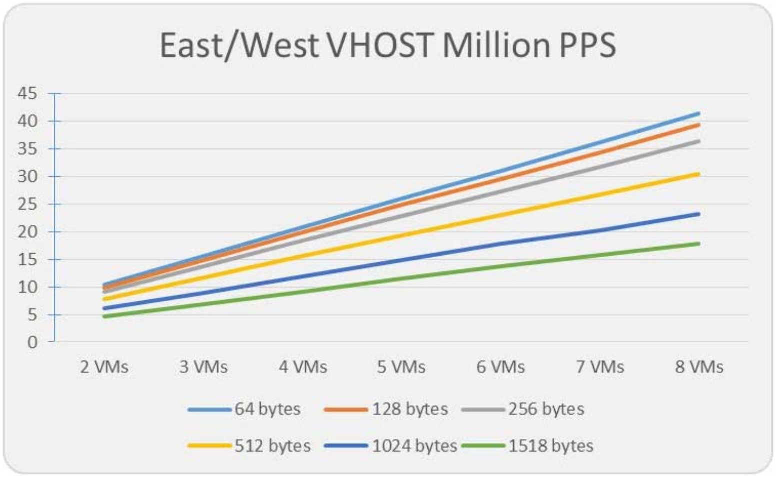

Figure 36. VHOST Performance - East/West Packets per Second

Analysis:

The charts in Figure 35 and Figure 36 show that the VHOST interface scales very nicely. As more VNF VMs are added to the processing chain, more packets are processed.

3.2.2.2 SR-IOV Virtual Function

The testing of the SR-IOV interface in an East/West (chaining) mode was accomplished in the same way as the VHOST East/West testing, with the exception of using the SR-IOV VF in the VNF VM rather than the VHOST interface. Traffic is received by the first VNF VM and sent along the chain until it reaches the last VNF VM, where it is then sent back to the Transmit system.

Since this is a SR-IOV interface, the traffic is never processed by the hypervisor. Instead, the Intel Ethernet Controller receives the packet and routes it to the VM based on the programmed MAC Address assigned to each VF.

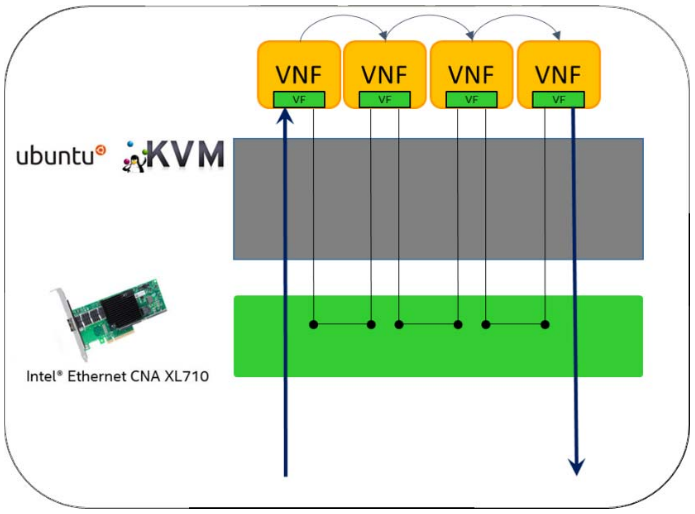

Figure 37. SR-IOV East/West Test Topology

Recorded Data:

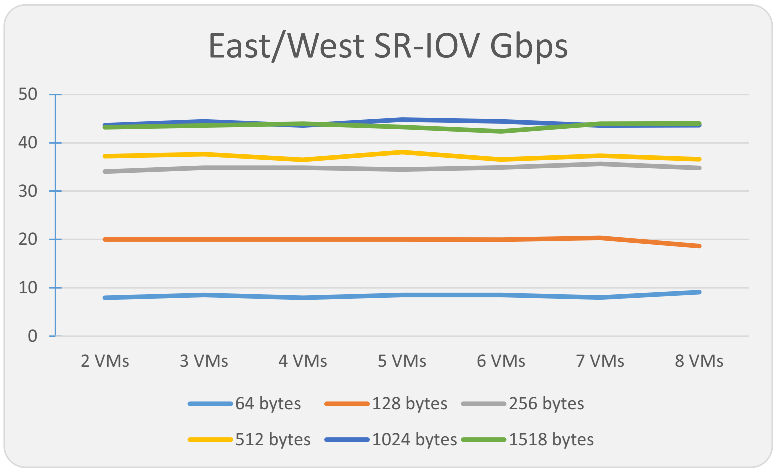

Figure 38. SR-IOV Performance - East/West Gigabits per Second

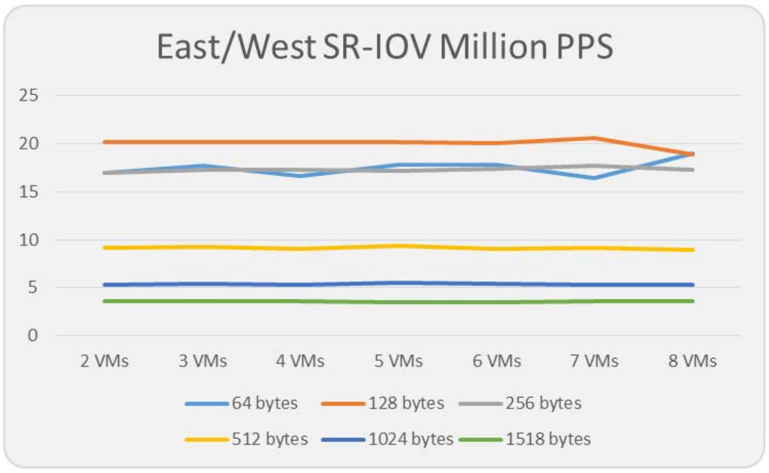

Figure 39. SR-IOV Performance - East/West Packets per Second

Analysis:

Figure 38 and Figure 39 show that East/West SR-IOV performance does not seem to scale. Adding more VNF VMs resulted in no additional packets being processed per second. This is in sharp contrast to what happens in a North/South topology with SR-IOV.

The observed data shown here is the reason for this document, and the explanation is detailed in Section 4.0.

4.0 Deeper Analysis of the Results

This section provides a closer look at the data and results presented in Section 3.0. As a reminder, the charts shown previously and following all reflect the total packets, or Gbps transmitted by all the VMs, not by each individual VM.

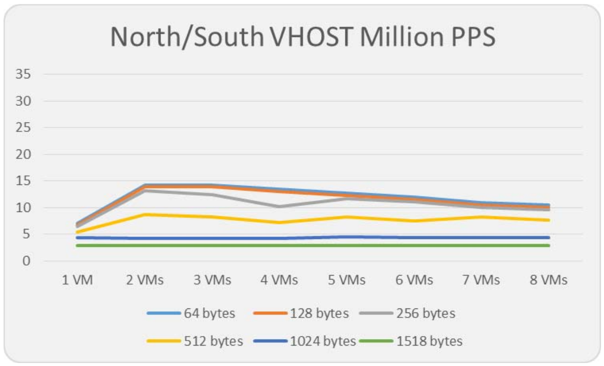

Figure 40. VHOST North/South PPS

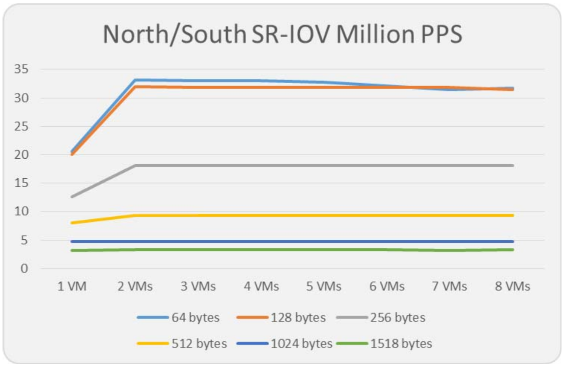

Figure 41. SR-IOV North/South PPS

Examining the North/South traffic pattern results side-by-side reveals that using a SR-IOV interface scales very nicely at a performance that is vastly superior to that of a DPDK-backed VHOST interface. As the number of VMs increases, the overall system throughput and packets processed remains constant.

Examining the same interfaces, using the same VNF VMs and changing the traffic pattern to an East/ West flow, there is a significant difference, as shown in Figure 42 and Figure 43.

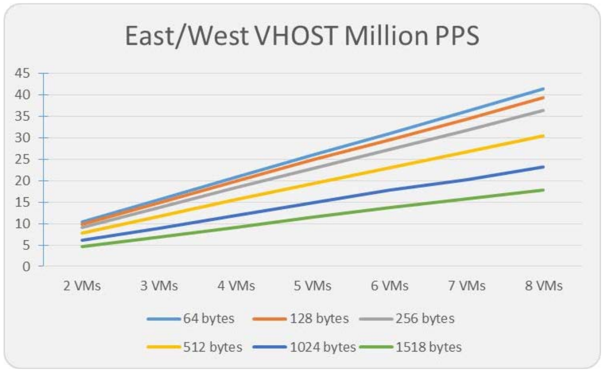

Figure 42. VHOST East/West PPS

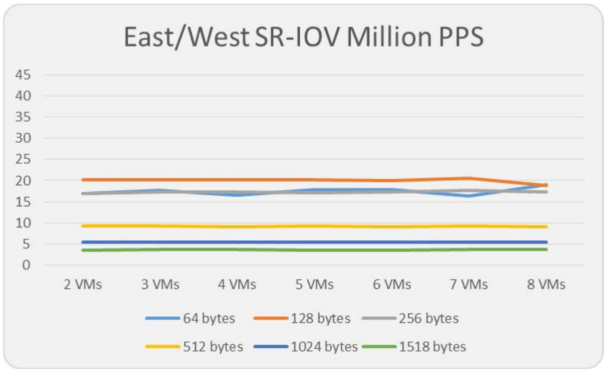

Figure 43. SR-IOV East/West PPS

The reasons for these differences is explained in the following sections.

4.1 North/South Traffic Pattern

The North/South traffic for VHOST and SR-IOV is shown in Figure 44 and Figure 45, respectively.

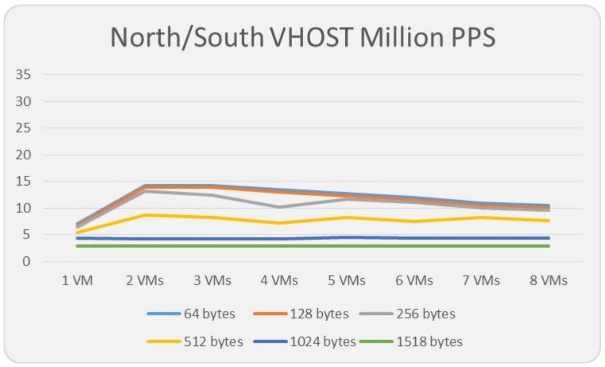

Figure 44. VHOST North/South PPS

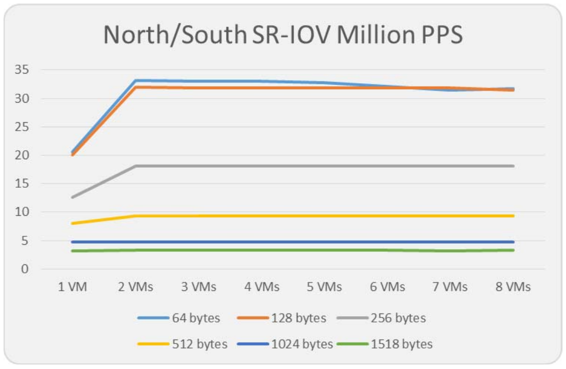

Figure 45. SR-IOV North/South PPS

Both of these tests show a performance gain when going from one VNF VM to two for smaller packet sizes. This is because these setups are not optimized, but are a more "general" configuration. If optimizations are made for one type of test, the other suffers in performance.

With two or more VNF VMs, the SR-IOV performance is very consistent and scales with the number of VNF VMs. This is because the Hardware Virtual Ethernet Bridge (VEB) within the Intel® Ethernet CNA XL710 has a round-robin scheduler by default, resulting in equal bandwidth being available for each Virtual Function. For more information, refer to the following:

http://www.intel.com/content/www/us/en/pci-express/pci-sig-sr-iov-primer-sr-iov-technology-paper.html

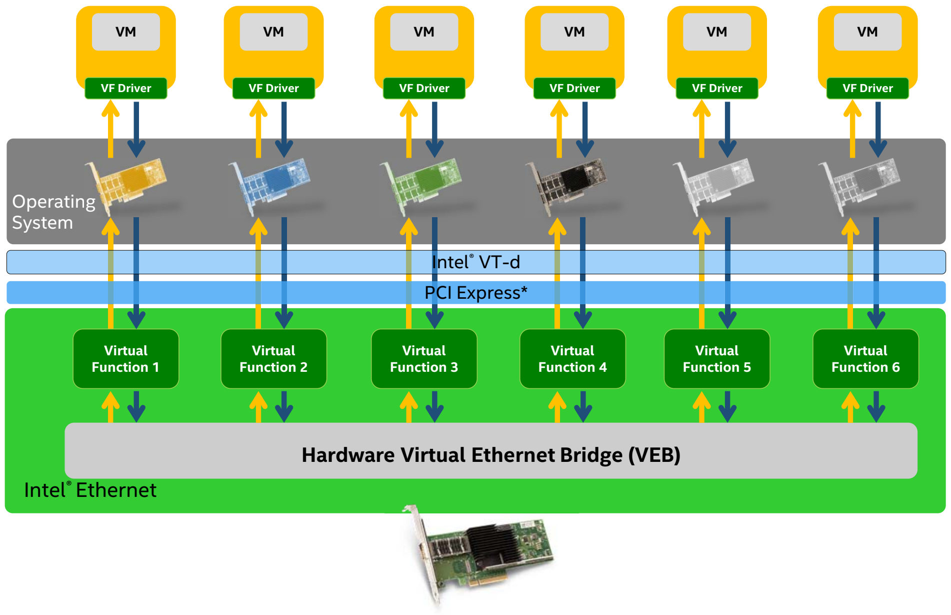

Figure 46. SR-IOV North/South Traffic Flow

The VHOST interface, on the other hand, does not scale very well at anything other than the largest packets sizes used in the testing. There are a few reasons why the VHOST interface does not scale very well for these tests in comparison to SR-IOV. First is that while SR-IOV traffic bypasses the hypervisor and vSwitch, VHOST traffic does not. It is first read by a DPDK PMD, then must be handled by OVS and directed to the appropriate VNF VM. This essentially means that every packet being sent to a VNF VM must be touched by the CPU three times: once while being read from the physical Ethernet device by the DPDK PMD, once within OVS, and another within the VNF VM itself when being processed by the DPDK **testpmd** application.

The same thing occurs when the data is transmitted from the VNF VM. The VM CPU must transmit the packet, which makes its way to OVS, which must examine and process the packet again before transmitting it out to the Intel® Ethernet CNA XL710, which is done by the DPDK PMD.

In comparison, a packet making a round-trip into a VHOST interface in a VNM VM gets "touched" by the CPU twice as often as when a SR-IOV VF is utilized.

Next comes the DPDK itself. Using the DPDK over the standard Linux Network Stack provides a tremendous performance boost. That being said, the DPDK is still a software-based entity, and as such has its limitations. Testing showed that DPDK topped out at about 7.5 Mpps when using two VNF VMs, and slowly degraded after that when more VNF VMs were added.

This performance does scale with improvements in each generation of Intel® Xeon® Processors, as well as improvements continually being added to the DPDK software itself.

4.2 East-West Traffic Pattern

The East/West traffic for VHOST and SR-IOV is shown in Figure 47 and Figure 48, respectively.

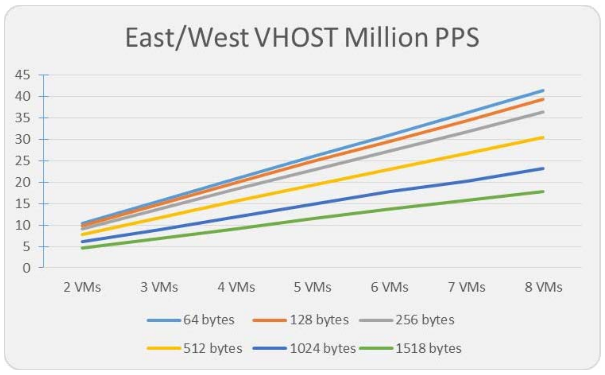

Figure 47. VHOST East/West PPS

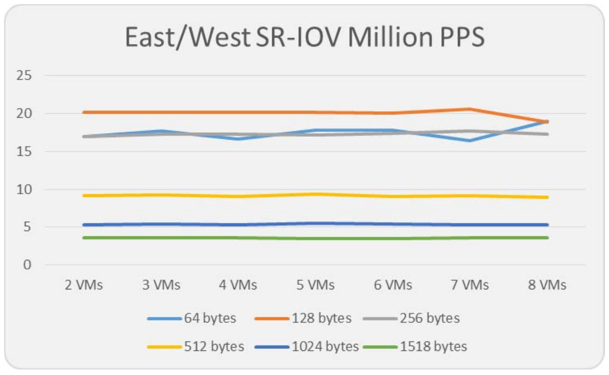

Figure 48. SR-IOV East/West PPS

Figure 47 and Figure 48 show the packets per second performance for the tests using the DPDK-backed VHOST interface and the SR-IOV VF interface. Figure 47 and Figure 48 reveal that East/West traffic using the VHOST interface scales very nicely as more VNF VMs are added, reaching more than 40 million packets per second being processed. When the SR-IOV interface is used, it does not scale, and it actually appears that the packets per second processed stay nearly constant as more VNF VMs are added.

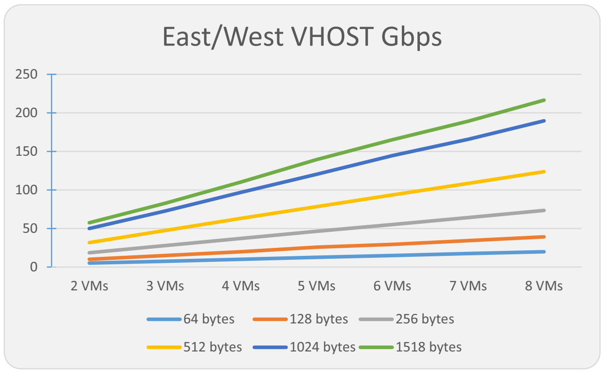

Figure 49. VHOST East/West Gbps

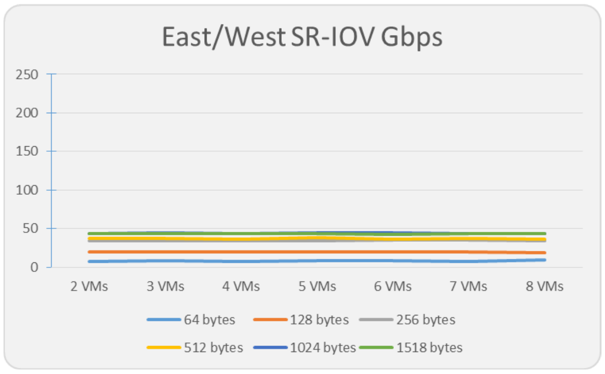

Figure 50. SR-IOV East/West Gbps

Examining the Gbps throughput as depicted in Figure 49 and Figure 50 shows a dramatic difference between using the DPDK-backed VHOST and the SR-IOV VF interfaces. This may seem counter-intuitive at first, because when using the VHOST interface, all the traffic must flow through the hypervisor and OVS, while when using SR-IOV, all the traffic flows over the Intel® Ethernet CNA XL710.

It is precisely because the traffic flows over the Intel® Ethernet CNA XL710 that the performance of East/West traffic does not scale.

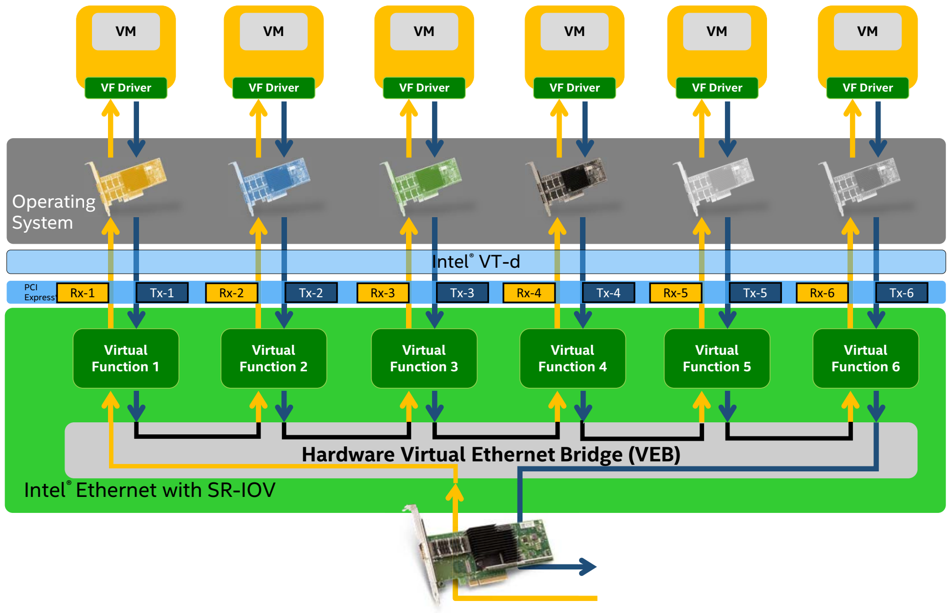

Figure 51. SR-IOV East/West Traffic Flow

Figure 51 shows how the traffic flows across six VM's in an East/West pattern. Each time a packet is moved from one VM to the next, it is DMA'd to the Intel® Ethernet CNA XL710, which examines the packet, sees that it is destined for another SR-IOV VF, and then DMA's the packet into the appropriate VM (the next in the chain).

Each time a packet moves to and from the Intel® Ethernet CNA XL710, it moves across the PCIe interface. So a single packet that flows through the six VM's shown in Figure 51 crosses the PCIe interface 12 times; six in the Tx direction and six in the Rx direction.

The Intel® Ethernet CNA XL710 is a PCI Express 3.0 device. The maximum theoretical data rate that a PCIe 3.0 x8 device can support is 50 Gbps, bidirectional (Tx and Rx).

Note: This does not include the encoding overhead, which slightly reduces the effective maximum throughput.

Many peripheral devices do not internally support that maximum data rate. The Intel® Ethernet CNA XL710, for example, can internally handle around 45 Gbps throughput, which is more than sufficient for a 40 GbE device. This can be seen in Figure 52; the larger packet sizes actually achieve a throughput of greater than 40 Gbps.

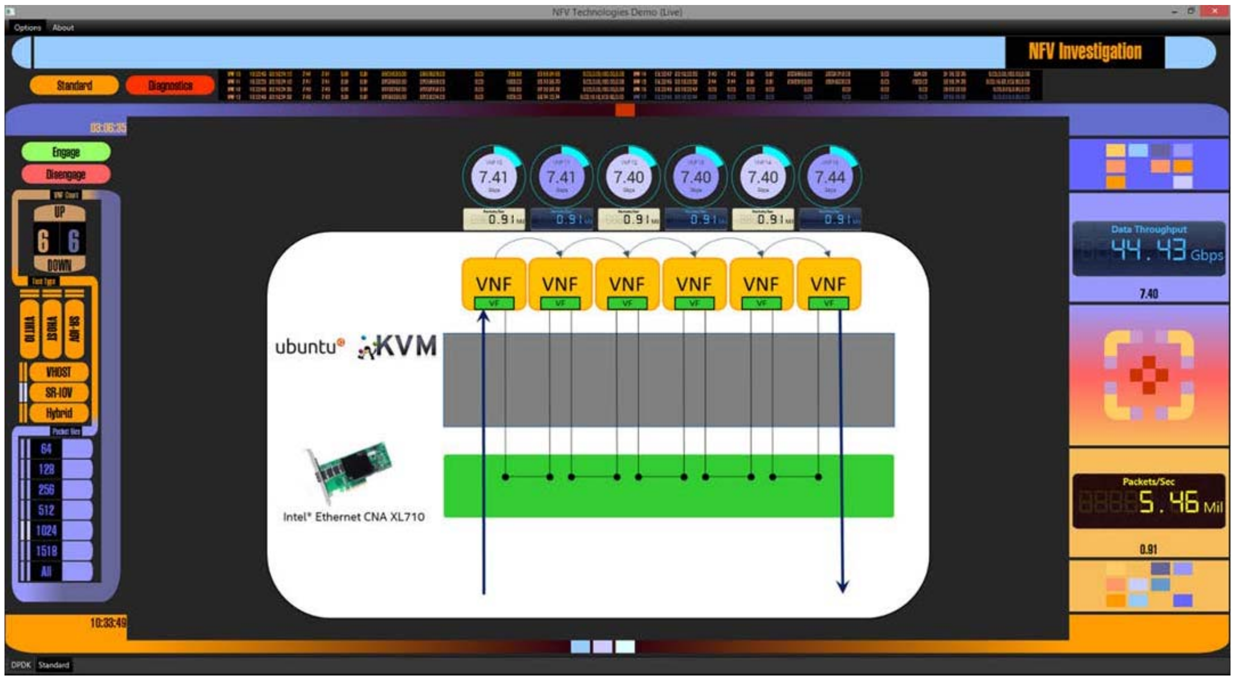

Figure 52. SR-IOV East/West Gbps

Reexamining the packet that flows from VM-to-VM in a SR-IOV chain, assume that the data is coming in at a rate of 5 Gbps from a remote source into the first VM. By the time the data is propagated through all of the VMs, that 5 Gbps is multiplied by six (in this situation as far as the PCIe interface is concerned, 30 Gbps of Tx as well as 30 Gbps of Rx bandwidth, out of a maximum of 50 Gbps has been utilized). To increase to rate to 10 Gbps for each VM, the device and the PCIe interface would have to support 60 Gbps.

Thus, there is an inherent limitation in sending traffic over the PCIe interface, even if a device was capable of sending, receiving, and processing 60 Gbps. The PCIe x8 interface simply cannot handle that much data.

There is also another limitation. The smaller the packet size, the more packets per second can be sent. Ethernet Controllers have a maximum number of packets per second that they can process. Intel® Ethernet CNA XL710 can process up to around 39 Mpps; this can be achieved in optimized configurations, not the general configuration used for this document.

While the performance in sending traffic in an East/West pattern is limited by the PCIe interface and the Ethernet Controller, the DPDK-backed VHOST interface has no such limitations. It is only limited by CPU and memory speeds.

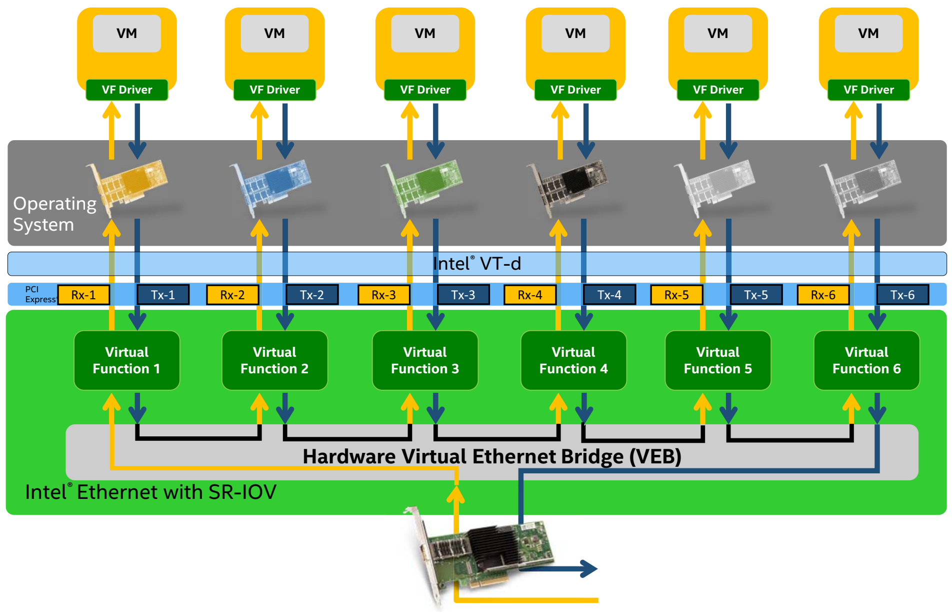

Figure 53. East/West Chaining with SR-IOV

When traffic moves from one VM to another using the VHOST interface, the para-virtualized VHOST interface is able to (at a very high level) do a **memcpy** or page sharing to transfer the data. This is only limited by the platform performance (CPU and memory) and the underlying software (Hypervisor, OVS, DPDK, and so on).

Note: When doing North/South traffic, the VHOST interface that is going through OVS was limited to around 10 Mpps. However, that limitation does not seem to be there when doing East/West traffic over VHOST, where at the same packet size (64 bytes) over 40 Mpps was seen. The reason for this is that in the test a single DPDK PMD and CPU core was used to read data from the physical Ethernet interface (the Intel® Ethernet CNA XL710). If more cores were to allocate to this task, the performance would certainly have increased.

5.0 Summary

Examining the information in Figure 54 might lead to the conclusion that using SR-IOV for NFV solutions is an obvious superior technology choice.

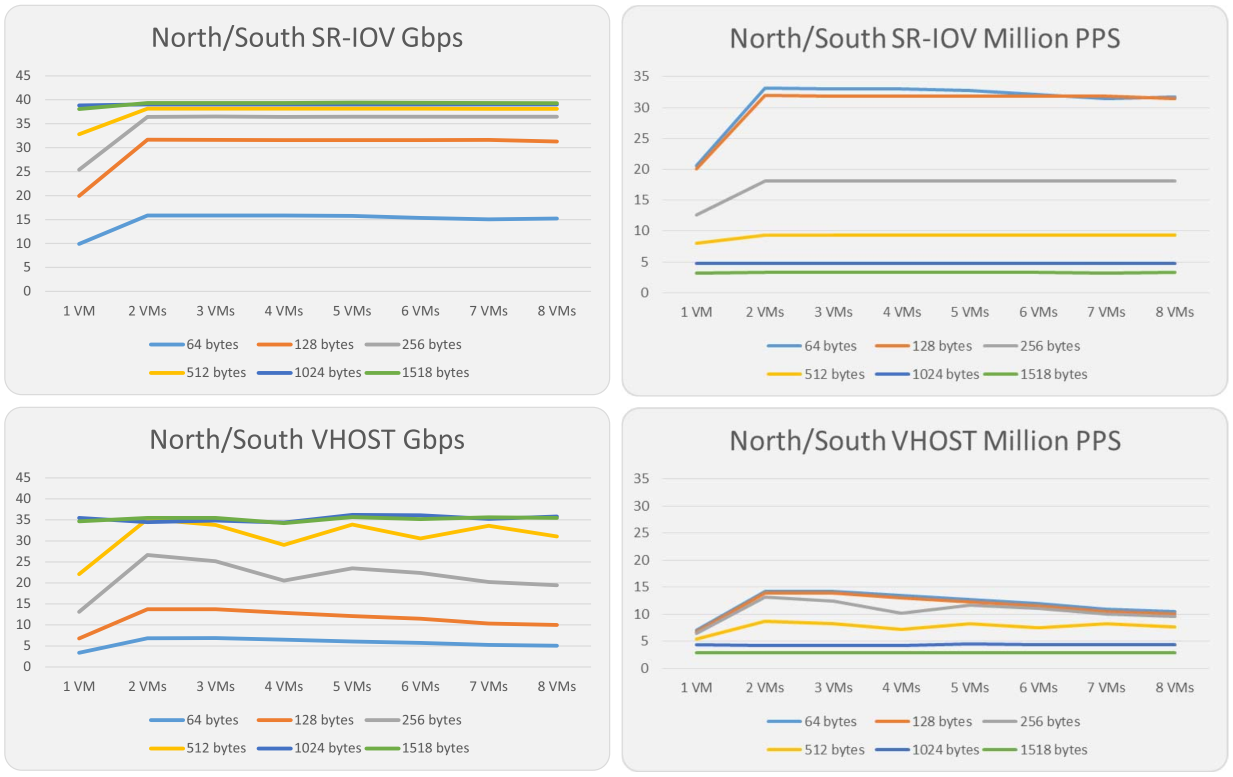

Figure 54. Performance Comparisons - North/South

In fact, it would seem that some Ethernet vendors are promoting that very idea. However, as hopefully this document shows, there is more to consider.

SR-IOV is a great choice for a NFV solution with North/South traffic patterns. It scales very nicely and uses fewer CPU resources, as it bypasses the hypervisor and vSwitch.

However, the same cannot be said when traffic moves in an East/West pattern, from VM-to-VM. When this occurs over SR-IOV, every packet must traverse the PCIe interface for every hop, reducing the amount of available PCIe bandwidth at each step. Instead, using DPDK-backed para-virtualized devices supported by a DPDK enhanced vSwitch provides a much more efficient and scalable solution.

The choice of solution depends on the type of workload. If North/South patterns are needed, SR-IOV is potentially the better choice. If East/West traffic (like service chaining) is needed, using DPDK and para-virtualization is likely desirable.

If the solution does not call for using DPDK, Virtio would seem to be a superior choice over the DPDK-backed VHOST technology.

Our recommendation is to take a careful look at the various options available and the flow your traffic needs to take to achieve the desired results.

5.1 Data Collected

For reference purposes, this section contains the actual data values collected and used within this document.

Traditional Traffic - Netperf

	Virtio	VHOST	SR-IOV
Packet Size	Gbps	Gbps	Gbps
64 Bytes	5.35	4.80	6.31
128 Bytes	9.16	8.94	11.99
256 Bytes	16.25	15.68	21.45
512 Bytes	19.71	18.45	38.72
1024 Bytes	35.36	34.66	38.79
1518 Bytes	37.54	36.85	39.15
64k	38.88	38.88	39.22

Traditional Traffic - Netperf
[previous CPU/OVS/DPDK]

	Virtio	VHOST	SR-IOV
Packet Size	Gbps	Gbps	Gbps
64 Bytes	5.35	4.80	6.13
128 Bytes	8.80	7.75	11.69
256 Bytes	13.63	10.71	21.45
512 Bytes	16.40	13.33	37.72
1024 Bytes	26.39	15.72	38.35
1518 Bytes	35.42	16.22	38.98
64k	37.77	18.23	39.25

Virtio North/South

#VM	1 VM		2 VMs		3 VMs		4 VMs		5 VMs		6 VMs		7 VMs		8 VMs	
Packet Size	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS
64 bytes	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.04	0.01	0.04
128 bytes	0.02	0.02	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.03	0.03
256 bytes	0.06	0.03	0.04	0.02	0.04	0.03	0.05	0.02	0.05	0.02	0.06	0.03	0.07	0.04	0.08	0.04
512 bytes	0.16	0.04	0.12	0.02	0.13	0.03	0.14	0.04	0.14	0.04	0.14	0.03	0.14	0.05	0.14	0.04
1024 bytes	0.51	0.06	0.39	0.03	0.41	0.05	0.40	0.05	0.60	0.07	0.50	0.07	0.46	0.07	0.48	0.08
1518 bytes	0.95	0.08	0.90	0.07	0.76	0.06	1.00	0.08	0.96	0.08	0.88	0.06	0.84	0.07	0.96	0.08

SR-IOV North/South

	1 VM		2 VMs		3 VMs		4 VMs		5 VMs		6 VMs		7 VMs		8 VMs	
Packet Size	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS
64 bytes	9.90	20.60	15.84	33.10	15.84	32.99	15.84	33.00	15.74	32.80	15.35	32.04	15.05	31.39	15.22	31.71
128 bytes	19.94	20.09	31.65	31.92	31.63	31.86	31.60	31.80	31.57	31.79	31.60	31.83	31.64	31.90	31.28	31.48
256 bytes	25.40	12.60	36.44	18.09	36.50	18.09	36.44	18.09	36.45	18.07	36.48	18.07	36.47	18.09	36.45	18.08
512 bytes	32.80	8.08	38.13	9.38	38.13	9.39	38.14	9.39	38.16	9.38	38.16	9.37	38.11	9.37	38.12	9.36
1024 bytes	38.83	4.77	39.05	4.79	39.03	4.77	39.04	4.80	39.05	4.80	39.04	4.79	39.03	4.77	39.01	4.80
1518 bytes	38.07	3.15	39.32	3.24	39.34	3.24	39.31	3.25	39.39	3.25	39.38	3.24	39.33	3.23	39.30	3.27

VHOST North/South

	1 VM		2 VMs		3 VMs		4 VMs		5 VMs		6 VMs		7 VMs		8 VMs	
Packet Size	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS
64 bytes	3.39	7.07	6.84	14.25	6.87	14.30	6.48	13.46	6.07	12.67	5.74	12.00	5.23	10.92	5.06	10.50
128 bytes	6.77	6.84	13.76	13.87	13.75	13.86	12.88	13.00	12.12	12.26	11.50	11.59	10.34	10.44	10.02	10.08
256 bytes	13.12	6.51	26.66	13.23	25.17	12.46	20.56	10.19	23.47	11.66	22.36	11.09	20.21	10.01	19.45	9.65
512 bytes	22.08	5.44	35.20	8.66	33.80	8.31	29.03	7.15	33.90	8.33	30.58	7.52	33.60	8.26	31.08	7.64
1024 bytes	35.48	4.35	34.46	4.23	34.85	4.27	34.35	4.21	36.18	4.47	36.10	4.45	35.23	4.33	35.80	4.38
1518 bytes	34.66	2.87	35.48	2.94	35.47	2.93	34.20	2.82	35.67	2.95	35.20	2.92	35.60	2.94	35.47	2.94

SR-IOV EAST/West

	1 VM		2 VMs		3 VMs		4 VMs		5 VMs		6 VMs		7 VMs		8 VMs	
Packet Size	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS
64 bytes			7.97	16.90	8.52	17.70	7.96	16.60	8.53	17.80	8.53	17.80	7.98	16.40	9.12	18.98
128 bytes			20.00	20.16	19.99	20.12	20.01	20.18	20.01	20.14	19.97	20.01	20.33	20.56	18.64	18.80
256 bytes			34.06	16.88	34.86	17.28	34.84	17.22	34.48	17.12	34.88	17.32	35.63	17.71	34.80	17.30
512 bytes			37.20	9.16	37.63	9.24	36.49	8.99	38.09	9.35	36.51	9.00	37.31	9.18	36.58	8.97
1024 bytes			43.65	5.34	44.44	5.44	43.55	5.35	44.81	5.48	44.43	5.46	43.60	5.34	43.61	5.36
1518 bytes			43.18	3.56	43.56	3.60	43.96	3.64	43.25	3.55	42.38	3.49	43.93	3.64	43.99	3.61

VHOST EAST/West

	1		2 VMs		3 VMs		4 VMs		5 VMs		6 VMs		7 VMs		8 VMs	
Packet Size	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS	Gbps	MPPS
64 bytes			4.99	10.36	7.48	15.56	9.98	20.77	12.51	25.97	14.90	30.96	17.38	36.20	19.85	41.46
128 bytes			9.87	9.86	14.84	14.90	19.73	19.86	25.65	24.83	29.31	29.58	34.15	34.42	39.06	39.38
256 bytes			18.57	9.10	27.75	13.80	37.01	18.34	46.32	22.91	55.06	27.36	64.12	31.81	73.36	36.41
512 bytes			31.70	7.82	47.44	11.67	63.00	15.53	78.34	19.36	93.48	23.01	108.54	26.73	123.75	30.40
1024 bytes			49.94	6.12	72.51	8.86	96.85	11.79	120.25	14.78	144.66	17.74	165.80	20.26	189.82	23.30
1518 bytes			57.43	4.73	82.62	6.84	110.28	9.11	139.42	11.53	165.27	13.75	189.27	15.73	216.41	17.73

5.2 Final Notes on Test Setup

The setup for this testing was a general configuration; it was not optimized for any specific type of interface (VHOST or SR-IOV). There are a number of settings that could have been configured to make each one more efficient than what is shown in this document. If that was the case, over 33 Mpps could easily achieved in a VNF VM using SR-IOV (North/South traffic). However, the configuration to achieve this reduces overall performance for VHOST.

The purpose of this document is not to show the absolute best performance possible in a given configuration. Rather, it is to demonstrate that using SR-IOV in a NFV solution needs to be considered carefully. SR-IOV is potentially a great solution for North/South traffic, but not so for East/West traffic.

Nine VMs were set up for these tests, which was at the limit of the system under test in the way it was configured (each VM having pinned CPU cores, huge pages and so on.). One could likely setup a system that supports more efficient VHOST and SR-IOV interfaces concurrently by using fewer VMs and not making them all have the same configuration.

The tools used were not necessarily the best available for doing performance testing; they are all software based. While they are terrific tools, they are not the same as commercial products. Additionally, each VNF VM only used one queue and one CPU core for each test. DPDK performance can be improved by adding more queues and cores.

Lastly, the kind of testing done may not be something that reflects actual traffic. Using the DPDK **pktgen** application to blast packets is good for showing flow, but that traffic is not under any kind of flow control, either hardware or software (such as TCP Window sizes).

6.0 Additional Resources

Intel SR-IOV Explanation Video:

http://www.youtube.com/watch?v=hRHsk8Nycdg

Intel SR-IOV Primer Document:

http://www.intel.com/content/www/us/en/pci-express/pci-sig-sr-iov-primer-sr-iov-technology-paper.html

White Paper - Open vSwitch Enables SDN and NFV Transformation

https://soco.intel.com/servlet/JiveServlet/download/38-636829/open-vswitch-enables-sdn-and-nfv-transformation-paper.pdf

Bandwidth, Packets per Second, and Other Network Performance Metrics:

http://www.cisco.com/c/en/us/about/security-center/network-performance-metrics.html

LEGAL

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps,

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade. Intel® Turbo Boost Technology requires a PC with a processor with Intel Turbo Boost Technology capability. Intel Turbo Boost Technology performance varies depending on hardware, software and overall system configuration. Check with your PC manufacturer on whether your system delivers Intel Turbo Boost Technology. For more information, see www.intel.com/technology/turboboost.

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 purchase. For more complete information about performance and benchmark results, visit www.intel.com/performance. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software, or configuration may affect your actual performance.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

© 2017 Intel Corporation. All rights reserved.