Getting Real Performance from a Virtualized CCAP

A Technical Paper prepared for SCTE/ISBE by

Mark Szczesniak
Software Architect
Casa Systems, Inc.
100 Old River Road
Andover, MA, 01810
978-688-6706
mark.szczesniak@casa-systems.com

Table of Contents

List of Figures

Introduction

Virtualization of network functions and software defined networking (SDN) control of those functions promises service providers tantalizing benefits including faster time to market for new services, lower costs, and higher customer satisfaction. These benefits are particularly important as competition and user demand continue to rise year after year. But, virtualized solutions need to not only exist but also perform at least as well as their legacy counterparts. This is the challenge service providers are coming up against as virtualization initiatives try to make their way out of the lab and into the field.

The good news is that there are answers for the performance challenge. In the technical paper “Getting Real Performance from a Virtualized CCAP”, Casa Systems will present the underlying performance challenges inherent in implementing converged cable access platform (CCAP) functionality (video, IP voice and data) on the current generation of commercial off-the-shelf (COTS) x86 servers.

These challenges include:

• Guaranteeing performance of shared network function virtualization (NFV) infrastructure

• Designing for security in the face of cryptographic performance constraints in virtualized environments

• Providing maximum packet throughput for virtual network functions (VNFs)

This paper will explore each of these challenges and present the underlying factors driving these issues. Further, this paper will explore alternative solutions available today and options service providers should consider as they introduce virtualized CCAP in their networks. Some of the solution aspects to be explored include:

• Optimum server performance characteristics

• Optimum NFV infrastructure and configuration

• Using software solutions like Linux new API (NAPI) or data plane development kit (DPDK) to enhance server performance

• Enhancing VNFs for maximum performance and throughput

By analyzing the underlying challenges inherent in virtualizing full CCAP functions, and understanding the options available today to help overcome those challenges, this paper seeks to aid the advancement of virtual CCAP in the field.

Content

1. Background

Before virtualization, CCAP vendors provided physical network functions (PNFs), which included all the hardware and software needed for the full CCAP solution. With virtualization, CCAP vendors only provide the VNFs, which run on generic server hardware. A virtualized infrastructure manager (VIM) is used to manage and coordinate the physical resources needed by the VNFs. This means that service providers not only need to select the CCAP VNF; they also need to select a VIM and the servers on which everything will run.

Most virtualization infrastructure is tailored to cloud and enterprise applications, which were focused on allocating central processing unit (CPU), memory and disk to applications. Networking was expected as a given, as networking is always shared among applications. With virtualization of service provider network functions, networking performance becomes vitally important and leads to new challenges for virtualization infrastructure.

CCAP virtualization requires the physical / coax layer to be separate from the server. This adds an additional layer of complexity from the need to address security between the CCAP core and the device where the packet is translated to analog. For remote physical (PHY) layer designs, data packets are already encrypted in the core by the data over cable service interface specification (DOCSIS) media access control (MAC) layer, which uses baseline privacy interface (BPI+) encryption. For remote MAC-PHY designs, internet protocol (IP) encryption may be needed to protect the packets between the core and the remote MAC-PHY device.

To achieve the performance needed from virtualized service provider network applications, including virtualized CCAP (vCCAP), solutions need to move packets as quickly as possible in and out of the virtualization platform as well as rapidly encrypt and decrypt packets.

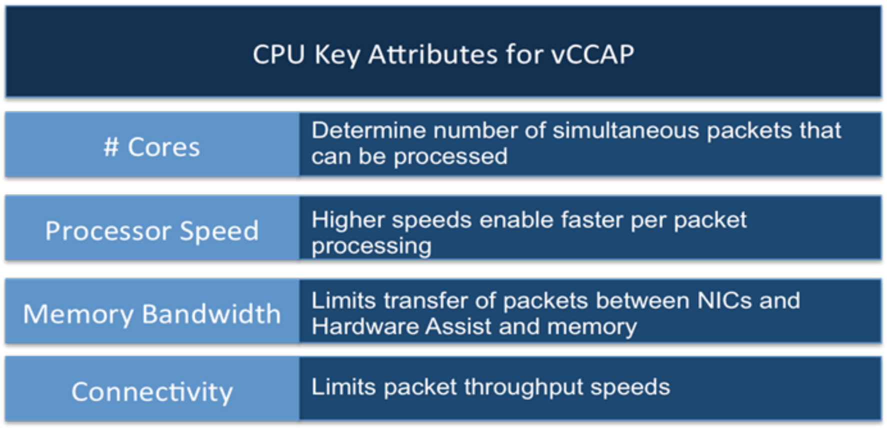
2. Hardware

There are many things to consider when buying a server for NFV. In addition to aspects like disk space, memory and CPU that one normally considers, network interface controller (NIC) bandwidth and non-uniform memory access (NUMA) are important for NFV. The space and number of available slots in the server may limit the number and type of NICs available. A multiple CPU server will have multiple NUMA regions with limited bandwidth between these regions. One may also want to consider adding hardware assist as a valid tradeoff to adding more servers. Hardware assist can be provided via accelerators integrated within the CPU or platform controller hub (PCH) of the server itself, or through specialized silicon (i.e., field programmable gate array (FPGA), application-specific standard product (ASSP), or application-specific integration circuit (ASIC)) added via peripheral component interconnect express (PCIe) add-in cards. These solutions can be used for CPU intensive tasks such as encrypting/decrypting packets or computing cyclic redundancy check (CRC) / checksum where the full packet needs to be traversed by the CPU.

Disk space is not normally an issue for NFV. Most VNFs will need some disk space for the VNF virtual machines (VMs) as well as some configuration storage, but neither consumes very much disk space. If analytics is part of the VNF, disk space may need further consideration, as analytics normally requires a large amount of data.

Memory is important for the control plane to store session/connection information. Some memory is also needed for packet buffers for forwarding, but buffer count should be limited to reduce packet jitter and delay. Even though the control plane may need a lot of memory, the amount of memory a server can have is normally 1-2 orders of magnitude greater than an embedded system, so running out of memory should not be an issue for VNFs.

The remaining choices which are relevant to server selection (e.g., CPU, NICs, and hardware assist) all impact packet throughput.



CPU Key Attributes for vCCAP	
# Cores	Determine number of simultaneous packets that can be processed
Processor Speed	Higher speeds enable faster per packet processing
Memory Bandwidth	Limits transfer of packets between NICs and Hardware Assist and memory
Connectivity	Limits packet throughput speeds

Figure 1 - CPU Key Attributes for vCCAP

When choosing the CPU, one should consider the number of cores, the speed, the memory bandwidth and connectivity, as shown in Figure 1. More cores give the ability to process more packets simultaneously. High speeds will process each packet faster. Memory bandwidth can limit the transfer of data between NICs and hardware assist and memory and thus limit the ability for the core to process packets. Most CPUs use PCIe for connectivity, though some have built-in NICs. Servers have PCIe slots to allow users to add extra capabilities to the servers. These slots are normally used for connectivity (NIC PCIe cards), but can also be used for other types of expansion such as hardware assist. PCIe is the current standard for expandability. Most servers have a mix between PCIe 2.0, which can transmit up to up to 4 Gbps per lane, and PCIe 3.0 which can transmit up to 7.877Gbps per lane. PCIe 4.0 will double PCIe 3.0 speeds when it is available.

There are CPUs on the market, which have 40 PCIe 3.0 lanes, and the expectation is that others will soon come to market with 48 PCIe 3.0 lanes. Some of those lanes may be used for standard interfaces such as universal serial bus (USB) and disk connectivity. Others are made available via LAN on motherboard (LOM) / daughter card and PCIe slots. LOMs / daughter cards are different terminologies to describe a server specific card that provides some NIC functionality. Most PCIe slots are either 8 or 16 lanes. An 8 lane PCIe 3.0 slot would be able to transmit 63Gbps, which could effectively handle a 50Gbps interface. A 16 lane PCIe 3.0 slot would be able to transmit 126Gbps, which could effectively handle a 100Gbps interface.

Some CPUs have a special bus for multi-socket designs. Most servers support 1 or 2 sockets, though some will support 4 sockets. Most servers with multiple CPUs use NUMA and thus need to use the inter-CPU bus for some memory accesses. Because the bus between the sockets is limited, it is better not to move data between sockets too frequently.

Most servers come with a few 1G NICs by default and have a couple of options for adding or upgrading NICs. They normally either have a LOM / daughter card as the first option for NICs or have built-in NICs. If more NICs are needed, the PCIe slots can be populated with NIC cards. For NFV, PCIe slots will be needed to maximize packet throughput. The number of PCIe slots used will depend on the number and size of slots available and whether hardware assist is used, as hardware assist will also require PCIe slots.

In a 1-rack unit (RU) server, there are normally either 2 or 3 slots. In the 3 slot models, all the slots are normally low-profile (LP) or half-height (HH) which leaves less space for ports. In addition, some of the slots are x8 lanes and some are x16 lanes. For a 2 RU server, as illustrated in Figure 2, there can be up to 7 slots, most of which are normally full height, but most are also x8 lanes because the CPUs do not provide enough lanes for 7 slots of PCIe connectivity. Because the standard interfaces normally take about 16 lanes, a 2 CPU server may have 64 lanes for connectivity.

Since most servers that support 2 CPUs are sold in both 1 CPU and 2 CPU configurations, the standard interfaces are normally all connected to the first socket, thus limiting the PCIe lanes available for NICs (and for Hardware Assist) to 24. The second socket will normally be able to use all 40 lanes for NICs (and Hardware Assist). In addition, a 1RU server also may not have the slot connections designed for uniformity in that the first socket is attached to 2 slots and the LOM / daughter card and the second socket is attached to 1 slot. This means that the first socket will have 8 lanes per slot and 8 lanes for the LOM / daughter card and the second socket will have 16 lanes for one slot and 24 lanes unused. 2RU servers do not have this second issue as they have space for more slots to use up to all 40 lanes.

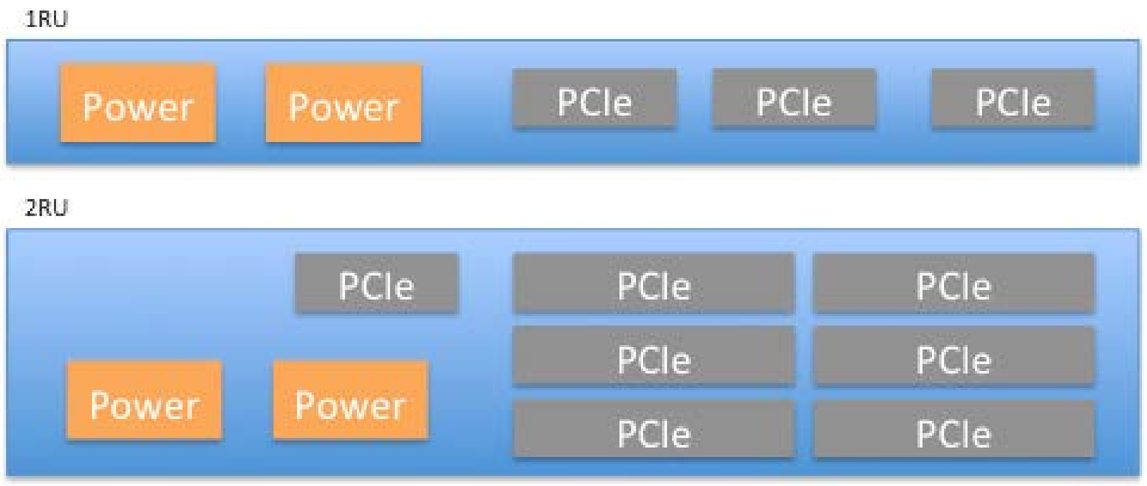


Figure 2 - 1RU vs 2RU PCIe Slots

For maximum throughput, packets should ingress, be processed and egress via the same socket and PCIe lanes attached to the socket. As the CPUs are normally the same across the sockets, it is optimal if the slots / PCIe lanes are allocated the same across sockets (i.e. first socket with 8 lanes for standard interfaces, 16 lanes for daughter card / LOM and 16 lanes for 1 slot and the second socket with 8 lanes for standard interfaces and 16 lanes for each of 2 slots). This symmetric design increases the bandwidth from 250Gbps to 400Gbps for the server. Though 200Gbps is probably a lot more than a high end CPU could process, it allows for half of the bandwidth to be taken by hardware assist (for crypto). Future servers are expected to address the importance of input / output (I/O) balance by providing PCIe connections split equally across both processors.

Additionally, it is important to consider the PCIe generation supported by the slot as some servers have both Gen2 and Gen 3 slots. There can also be a difference between the physical width of a PCIe slot and the width supported electrically. Some slots can physically take an x16 PCIe card, but only 8 PCI lanes are electrically connected to the PCIe controller. The connectivity of the PCIe slots to the PCIe controller can also impact throughput, as some slots are connected to PCIe controllers integrated inside the CPU as some are connected to ones inside the PCH. The latter creates a longer path to the CPU and memory.

Another limitation on the number of NICs is the size of the slot. The LOM / daughter cards and full height slots can normally support up to 4 ports. Low profile / half height slots can normally only support up to 2 ports. So, a 1RU server could support up to 10 ports. A 2RU server could support up to 30 ports. When you put the size restriction and lane restriction together, the 1RU server could support a max of 100Gbps via (10) 10Gbps ports or 400Gbps with (4) 100Gbps ports in a symmetric server. An asymmetric 1RU server would be limited to (3) 50Gbps (limited by the x8 lanes) and (1) 100Gbps ports. The 2RU server would be limited to (30) 10Gbps ports or (8) 50Gbps ports (limited by the x8 lanes). Since max NIC bandwidth for both 1RU and 2RU servers is 400Gbps, a rack of 1RU servers could provide twice the bandwidth of a rack of 2RU servers if the NICs were the limiting factor.

The NIC bandwidth may also be limited by the use of hardware assist. For most applications, if hardware assist is needed, the hardware assist bandwidth should match the NIC bandwidth and thus NIC bandwidth is cut in half in a symmetric server and is reduced even more in an asymmetric server. The asymmetric 1RU server only has 1 slot and thus cannot handle both a NIC card and a hardware assist card connected to the second socket. This means to use the second socket with hardware assist, all packets would need to traffic the quick path interconnect (QPI) bus, which would limit throughput, as shown in Figure 3.

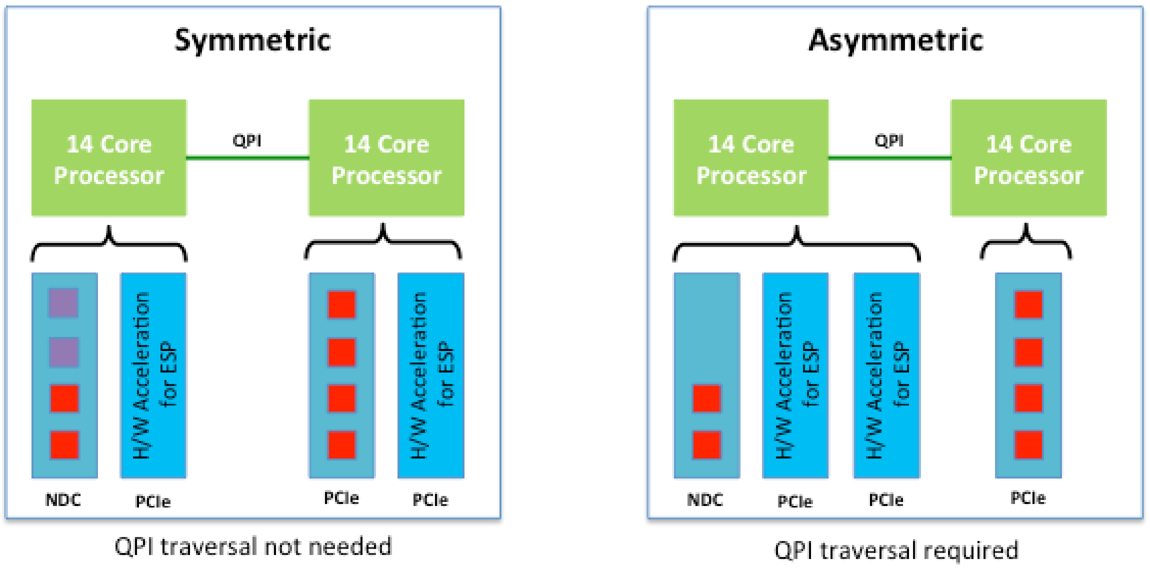
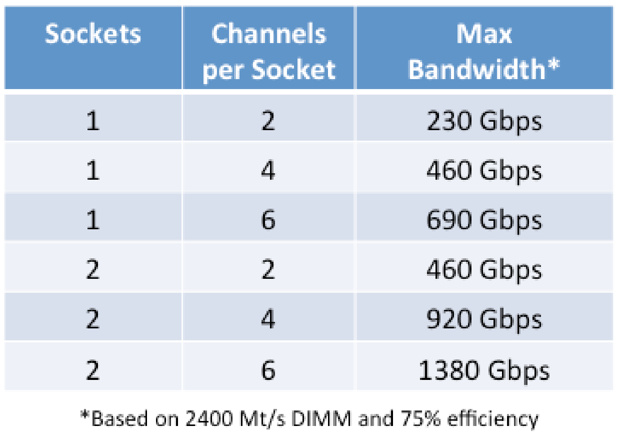


Figure 3 - Symmetric vs Asymmetric QPI Traversal

Another limiting factor is the number of memory channels per CPU. The high-end servers usually provide all the memory channels that the CPU can support. Current server models support 4 – 6 channels. The 6-channel servers have a 50% boost in memory bandwidth, as shown in Figure 4. Even if the server supports the full number of channels, there may or may not be memory in each channel.



Sockets	Channels per Socket	Max Bandwidth*
1	2	230 Gbps
1	4	460 Gbps
1	6	690 Gbps
2	2	460 Gbps
2	4	920 Gbps
2	6	1380 Gbps

*Based on 2400 Mt/s DIMM and 75% efficiency

Figure 4 - Channels per socket vs maximum bandwidth

When purchasing memory for the server, there is a choice between more modules with less memory each or fewer modules with more memory each. The modules with more memory are usually more economical, but they can reduce memory performance. For example, if you want to buy 64G of memory, you could buy (1) 64G dual in-line memory module (DIMM), (2) 32G DIMMs, (4) 16G DIMMs, (8) 8G DIMMs, or (16) 4G DIMMs. Purchasing (8) 8G DIMMs would allow for 1 DIMM per memory channel, thus giving the best memory bandwidth, but it also means that if you need to upgrade, you may need to replace your DIMM as opposed to just adding more.

The information above is all based on rack servers, but there are also chassis / blade systems. These systems can be denser in terms of CPU, but tend not to have the expansion capabilities to add NICs nor hardware assist, thus limiting throughput. In a normal datacenter, CPU horsepower is more important and thus the chassis / blade servers seem to be the better choice. But for network function virtualization, throughput is more important and thus separate servers seem to be the better choice.

Another thing to consider is how the servers are connected. A rack of 30 servers with (10) 10G ports would need a TOR (Top of Rack) switch with (300) 10G ports, whereas, if each server had (1) 100G port, the TOR switch would only need (30) 100G ports. The 10G TOR would be at least 6RU to support the same bandwidth as a 1RU 100G switch for inter-server connections. Either TOR switch would also need uplinks, which would add another 1-2RU to either switch depending on the ratio of uplink traffic to inter-server traffic. The extra size for uplink traffic is due to the fact that the current selections of lower-end 100G switches do not have higher speed uplinks. A 48 port 10G switch can have (5) 100G uplinks and not need to oversubscribe the (48) 10G ports used for server connections. A 32 port 100G switch would only be able to provide 16 ports for servers and 16 ports for the uplink without oversubscription. The 100G switch still has higher overall bandwidth of 1.6Tbps vs. 480Gbps and less cabling (32 vs. 53).

3. Virtualization Infrastructure

There are many different choices for a VIM, with VMware's vSphere and OpenStack being the most prevalent. Many vendors implement their own version of OpenStack with proprietary extensions. Some of these extensions are designed specifically for NFV to help VNFs scale and perform better than just using standard open-sourced OpenStack.

VMs need to be placed where they can access memory and NICs attached to the same NUMA region as the CPU. If not, the QPI bus between CPUs will need to be used, which is limited. This bus is limited because it is expected that most accesses would not need to traverse between NUMA regions. A core in one NUMA region that requires access to memory of the other NUMA region will be required to access the memory via the QPI bus. If this were an update, there would be 2 accesses to the bus (1 for the read and 1 for the write). This can be accomplished either by only giving the VM access to devices in 1 NUMA region or by enhancing the processes in the VM to understand NUMA and group its resources by NUMA region.

VMs can take a bit of time to scale up, so it is worthwhile to spin up an extra VM before it is needed, such that it will be ready when it needed.

When scaling VNFs that have flow state to multiple servers, a load-balancer is needed to direct packets to the correct VM. The TOR can do this if it is capable of the hash function that will work for the specific VNF. If not, the TOR will need to spray packets across multiple load-balance VMs that can then hash and send the packets to the correct VM to process the flow.

4. Software

Forwarding performance is not just affected by the hardware design of the server as described above. The software running on the server also affects it. Without virtualization, by default, all packets pass through the Linux kernel, which can be very slow, thus limiting packet throughput to under 10Gbps in a server even though 100Gbps NICs are available for servers. This is because the Linux kernel has not been optimized for forwarding. Also, packets that need to get to user space need to have 2 full packet copies when transitioning from kernel to user and then back to the kernel. These full packet copies take up a lot of time for the CPU. Additionally, the kernel uses interrupts to handle incoming packets, which have context switch overhead for each and every packet.

Data plane development kit (DPDK) technology has solved these issues with poll-mode drivers that run in user-space and with optimized libraries for memory and packet management. Packets are no longer handled in the kernel when using DPDK, as shown in Figure 5.

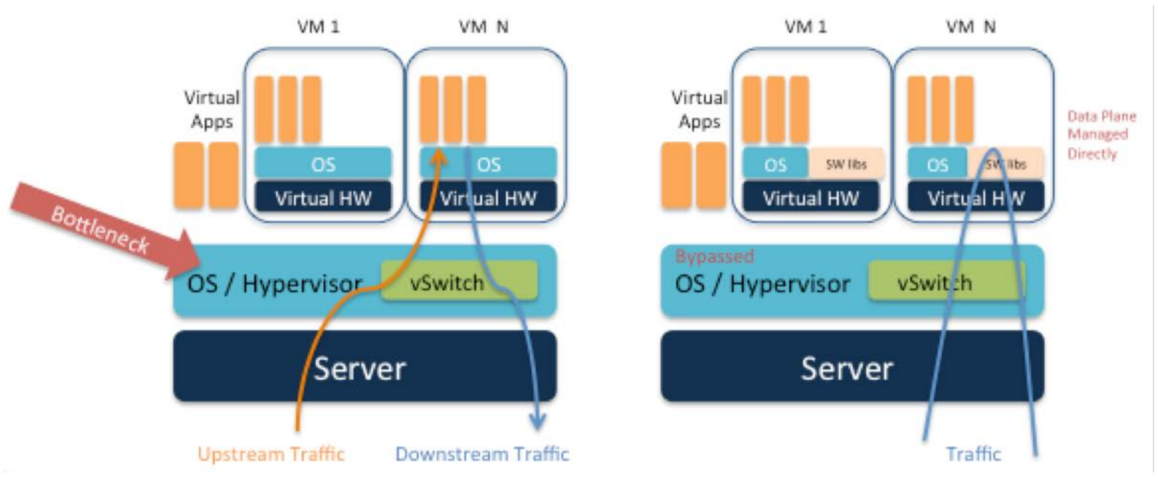
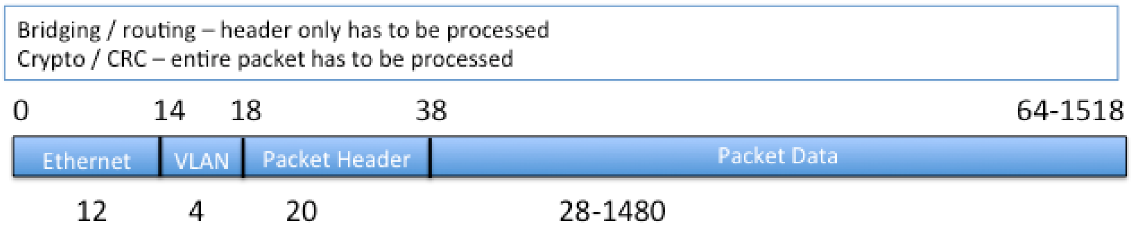


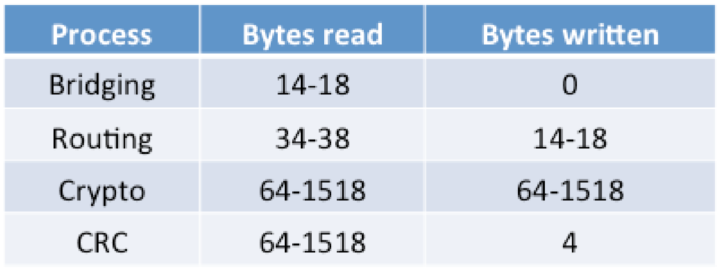
Figure 5 - Direct Management of Data Plane

They are put into a buffer that the user-mode process is polling to process packets. No interrupts will be generated and thus no context switch on a received packet. When virtualization is added, the vSwitch can become the bottleneck. By default, the vSwitch normally runs in the host OS's kernel, including Open vSwitch (OVS), which is used by OpenStack. This can limit server throughput to less than 10Gbps.

There are a few different options for fixing this issue. One is to get an accelerated vSwitch to speed it up. There are also DPDK extensions that will speed it up as well. Another option is to bypass the vSwitch by using single root input / output virtualization (SR-IOV) or Ironic. SR-IOV bypasses the vSwitch and gives packets directly to the VM. Ironic runs the machine natively on the server instead of within a VM. Ironic is OpenStack’s terminology for setting up a bare metal server with a specified image and networking. It is limited in that the NICs are determined by the physical hardware and not by the VM setup.

The type of actions performed on the packet can also have an affect on performance. For normal bridging and routing, only the header of the packet needs to be viewed, so only the header actually needs to be pulled into memory. But for operations like crypto and CRC, the whole packet needs to be read and thus needs to be transferred to memory, as shown in Figure 6.


Bridging / routing – header only has to be processed
Crypto / CRC – entire packet has to be processed
0
14
18
38
64-1518
Ethernet
VLAN
Packet Header
Packet Data
12
4
20
28-1480



Process	Bytes read	Bytes written
Bridging	14-18	0
Routing	34-38	14-18
Crypto	64-1518	64-1518
CRC	64-1518	4

Figure 6 - Bytes Processed per Function

Although CPU can perform the crypto and CRC algorithms, the time it takes to transfer the whole packet into memory and traverse the packet will take a lot of cycles. Adding more cores can compensate for the need for extra cycles, but adding hardware assist is more economical than adding cores. Adding crypto cards to existing servers is much less expensive than adding more servers, both in terms of capital expenditure (CAPEX) and operating expenses (OPEX). The amount of savings will depend on the ratio of cores needed for crypto to the cores needed for the VNF processing of packets.

For the different VNFs required by a vCCAP, the crypto core overhead is between 50% and 250%. There are some optimized crypto libraries than can help increase crypto performance, but they still do not approach the performance of dedicated crypto hardware. Another option for increasing CRC performance is to perform incremental CRC, but this requires access to the input CRC, which is not always available to VMs.

Another consideration is how packets traverse infrastructure and VMs to provide the functionality of the VNF. If a VNF were comprised of 1 VM that processes the packets in 1 pass, a packet would go from TOR to Server to TOR, as shown in Figure 7. A 1 Gbps stream of packets would require a 1Gbps connection from the TOR to the server and from the server back to the TOR. If however, the VNF required 3 VMs to process each packet, the 1Gbps stream could require up to 3Gbps of connection from TOR to server and server to TOR. If all 3 VMs were in the same server, the traffic could be optimized to stay in the server, but if the second VM were on a different server than the other 2, all 3Gbps would be needed. This may not be an issue for low bandwidth applications, but when the application scales up to 100 Gbps streams, requiring 300Gbps to handle the packet within the rack can be costly. The 100/300Gbps is the aggregate throughput of all flows through the application (and thus through all VMs).

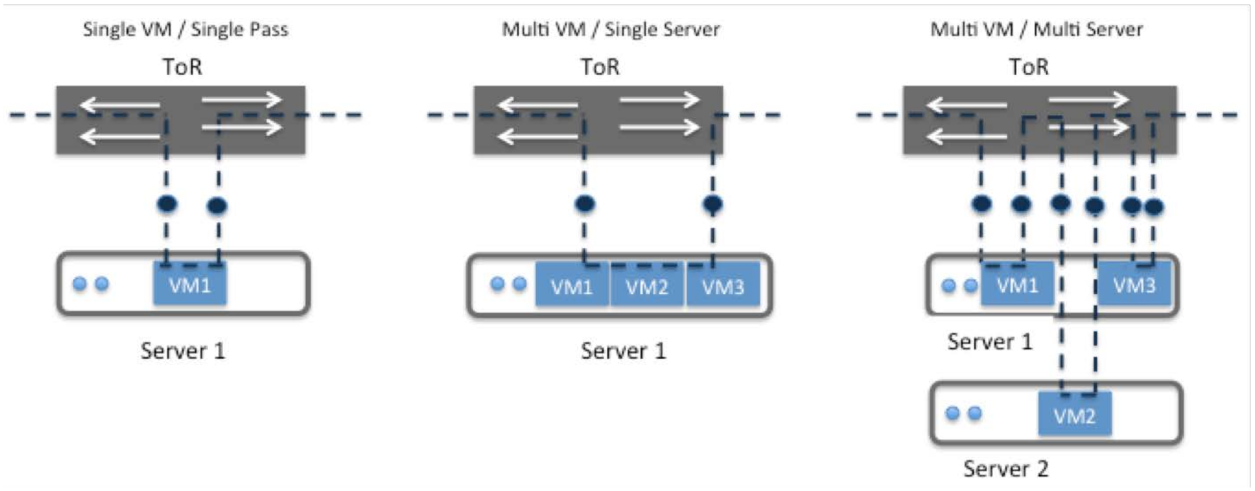


Figure 7 - VM and Server Impact on TOR Traversal

Also, when scaling to high bandwidth streams, there needs to be a way to direct packets to the correct cores within the correct VM from outside the server. A single core will only be able to direct up to 20-30Gbps of traffic to other cores before its queues overflow and packets are dropped, so sending it a 100Gbps stream means dropping 70-80% of the traffic. Most higher speed NICs support Receive Side Scaling (RSS) receive queues and use a hash to select to which VF (Virtual Function) to send packets. This also means VMs will need to use VFs and not the physical interfaces.

There are other new challenges arising from virtualization. Applications may need to evaluate session setup and modification rates as well as messaging speeds. In embedded systems, some applications use shared memory to speed up IPC (Inter-Process Communication) message handling. In a virtualized environment, applications may grow to more than one virtual machine and thus can no longer take advantage of shared memory.

Applications may need to rework their interactions to send larger amounts of data less often. They should also move to asynchronous communication to allow other processing to occur while waiting for replies. These are good practices for any multi-threaded or multi-process application, but were not always necessary in an embedded software environment. Certain VNFs, such as vCCAP may need clock synchronization for functionality such as upstream scheduling. Servers do not usually have a sufficient clock for driving synchronization protocols such at IEEE 1588 such that the master clock will need to come from outside the server. The vCCAP will need to be a slave to that master clock.

5. Wrap-up:

When looking at a server for forwarding performance, like that required by a vCCAP, NIC bandwidth, memory bandwidth and CPU core bandwidth need to be considered. If the application needs 1 core to process 10Gbps of traffic, then each core can match a 10G NIC and only 10 cores are needed in a 1RU server filled with 10G NICs. This would be reasonable for a low-end server with low-end CPUs (6 cores per CPU). Or the 10 cores could be used to fill a 100G NIC per socket, thus (2) 10-core CPUs would be needed. Current top of line servers with 22 cores can support (2) 100G NICs per socket. As long as 4 memory channels per socket have memory, the memory bandwidth is enough for the 200Gbps of traffic per socket. If the application requires 4 cores to process 10G, the top of line server would be needed to support just 100G of traffic (10 10G NICs or 2 50G NICs). The choice for 10G NIC will only work if hardware assist is not needed (as it would take away 2 PCIe slots leaving only space for 6 NICs).

Conclusion

The control plane pieces of VNFs should be able to run well on any server that meets the memory and disk requirements, but the data plane pieces of VNFs need the networking piece to work well. This requirement greatly reduces the number of servers from which to choose. The control plane and data plane parts of the VNF should therefore be split so each can be scaled independently. A VNF will get the most throughput per RU in a 1RU server with 2 (hardware assist needed) or 4 (no hardware assist needed) 100G ports with symmetric allocation of PCIe slots to CPU sockets. Either the VIM will need to allocate VMs for a VNF with all resources from the same NUMA region or the VNF will need to understand NUMA regions and be able to group resources per NUMA region. Data plane VMs either need to be connected via an accelerated vSwitch or use SR-IOV to bypass the vSwitch. They should use DPDK to bypass the kernel.

Abbreviations

AES-NI	Advanced encryption standard instruction set – new instructions
API	Application program interface
ASIC	Application-specific integration circuit
ASSP	Application-specific standard product
BPI	Baseline privacy interface
CAPEX	Capital expenditure
CCAP	Converged cable access platform
COTS	Commercial off-the-shelf
CPU	Central processing unit
CRC	Cyclic redundancy check
DIMM	Dual in-line memory module
DOCSIS	Data over cable service interface specification
DPDK	Data plane development kit
FPGA	Field programmable gate array
Gbps	Gigabits per second
HH	Half height
I/O	Input / output

IP	Internet protocol
IPC	Inter-process communication
LAN	Local area network
LOM	LAN on motherboard
LP	Low profile
MAC	Media access control layer
NAPI	New API
NFV	Network function virtualization
NIC	Network interface controller
NUMA	Non-uniform memory access
OPEX	Operating expenditure
OS	Operating system
OVS	Open vSwitch
PCIe	Peripheral component interconnect express
PCH	Platform controller hub
PHY	Physical layer
PNF	Physical network function
QPI	Quick path interconnect
RSS	Receive side scaling
RU	Rack unit
SCTE	Society of Cable Telecommunications Engineers
SDN	Software defined networking
SR-IOV	Single root input / output virtualization
Tbps	Terabits per second
TOR	Top of rack
USB	Universal serial bus
VF	Virtual function
vCCAP	Virtualized CCAP
VIM	Virtualized infrastructure manager
VM	Virtual machine
VNF	Virtual network function