ARCHITECTURE STUDY

Maximizing the Performance of DOCSIS 3.0/3.1 Processing on Intel® Xeon® Processors

Intel optimizes virtualized, cable modem termination system (vCMTS) architecture using Intel® Advanced Vector Extensions 512, Intel® AES New Instructions, DPDK, and Intel® QuickAssist Technology.

Authors

Michael O'Hanlon
Principle Engineer

Brendan Ryan
System Architect

Table of Contents

Introduction/Overview

The cable industry's need to deliver higher bandwidth, support multiple access technologies, make better use of deployed fibre, and increase service delivery velocity is driving many operators to consider re-architecting the cable modem termination system (CMTS). One approach is to virtualize the CMTS (vCMTS), which takes all the Data-over-Cable Service Interface Specification (DOCSIS) MAC and PHY features traditionally performed in a physical CMTS appliance and splits them across two main network components: a remote PHY (R-PHY) and a vCMTS platform.

Such a major re-architecting and platform retargeting is the source of much speculation, such as will it meet tomorrow's capacity and performance targets while delivering all the benefits promised by Network Functions Virtualization (NFV)? This paper aims to help service providers and original equipment manufacturers (OEM's) understand the potential of a virtualized architecture by focusing on a DOCSIS MAC data plane running on Intel® architecture and the Data Plane Development Kit (DPDK). It provides an insight into implementation options and establishes an empirical performance data baseline that can be used to estimate the capability of a vCMTS platform running on industry-standard, high-volume servers. Actual measurements were taken on the Intel® Xeon® Scalable processor-based system described in Appendix A.

The data shows that a single Intel Xeon processor core can support many Gbps of downstream MAC traffic, easily satisfying the 5 Gbps channel bandwidth required for an entire DOCSIS service group (SG) with a typical hybrid SG configuration comprising both DOCSIS 3.0 and 3.1 channels. When Intel® QuickAssist Technology (Intel® QAT) acceleration is employed in the system, a single core can even satisfy pure DOCSIS 3.1 configurations with up to five orthogonal frequency-division multiplexing (OFDM) channels (i.e., more than 9.5 Gbps).[1] What is particularly exciting is that a typical dual-processor server blade has something on the order of 40 such cores, and in prototype testing, the vCMTS workload scales nearly linearly, thus achieving compelling performance density for the total platform.

A future paper will discuss the incredible agility Intel architecture offers to accommodate different SG sizes, automated time-of-day capacity adjustments, and power optimizations, utilizing features such as Intel® Turbo Boost Technology and telemetry tools.

A Downstream DOCSIS 3.0/3.1 Pipeline

Figure 1 provides a basic architectural view of a vCMTS with R-PHY.

A vCMTS platform is implemented as a set of virtual network functions (VNFs) running on high-volume, industry-standard servers. These VNFs execute all the DOCSIS protocol functions from the MAC layer upwards, including the operations support systems (OSS) and business support systems (BSS) interfaces. A typical solution design also includes software infrastructure for VNF orchestration, life cycle management, real-time monitoring, and automated failure detection and recovery.

DOCSIS MAC functionality can be broken down into four categories: downstream data plane, upstream data plane, control plane, and system management. From a network perspective, the data plane traffic dominates, with about 5-10 times more traffic and related processing demand in the downstream compared to the upstream. Consequently, the focus of this paper is on downstream traffic with the assumption that upstream would have a similar, if not smaller, per packet cost as downstream.

Intel developed a vCMTS downstream data plane pipeline prototype built on top of the DPDK for the purpose of characterizing peak packet-processing performance on an Intel Xeon Scalable processor-based platform. The Intel-developed pipeline prototype is divided into upper MAC and lower MAC stages, which are executed as separate software threads affinitized to hyper-thread siblings[2] on a single Intel Xeon Scalable processor core. The pipeline is shown in Figure 2, and the stages are described in the following:

Upper MAC Processing Stages

1) Packet Rx
Using a DPDK poll mode driver (PMD), bursts of MAC frames are received from the NIC port and sent directly into an SG's upper MAC thread to begin vCMTS downstream packet processing.

2) Cable Modem Lookup
The DPDK hash API is used to do bulk lookup based on the destination MAC address of the frame to identify the DOCSIS filters, DOCSIS classifiers, service flow queue info, and security associations to be used for the target cable modem.

3) Subscriber Management
The number of active subscriber devices is monitored (tracked by destination IP addresses) and checked against the DOCSIS limit.

4) DOCSIS Filtering
The DPDK Access Control List (ACL) library is used to apply an ordered list of filters (e.g., masks, ranges, etc.) to the frame. These comprise permit and deny filters, and all 16 filters are evaluated per packet.

5) DOCSIS Classification
Packets are classified using an ordered list of classifiers and then are enqueued to one of four service flow queues using the DPDK ACL API.

6) Service Flow Scheduling
The DPDK scheduler API is used to apply rate-shaping, congestion control, and weighted-round-robin (WRR) scheduling to service flow queues. Appendix B contains test environment configuration information and relevant variables.

7) Channel Access Scheduling
The DPDK scheduler API has been adapted to perform channel access scheduling on packets after service-flow scheduling. Channel access scheduling is optimized by performing it in an earlier pipeline stage than is typically done in other implementations. This scheduling stage takes into account the encapsulation overhead added later in the pipeline.[21]

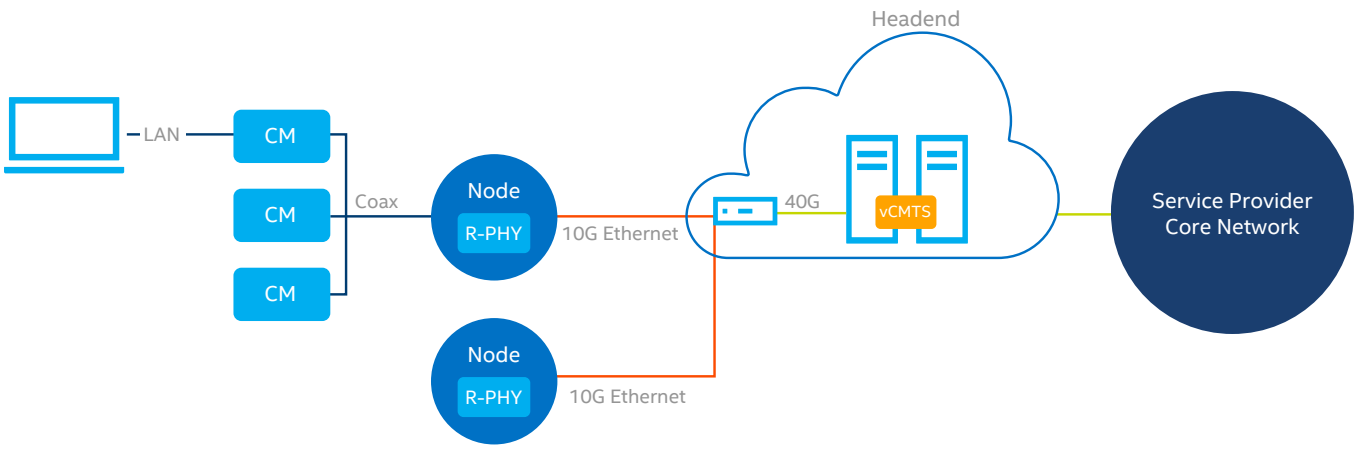

Figure 1. Basic Virtual Cable Modem Termination System (vCMTS) Architecture.

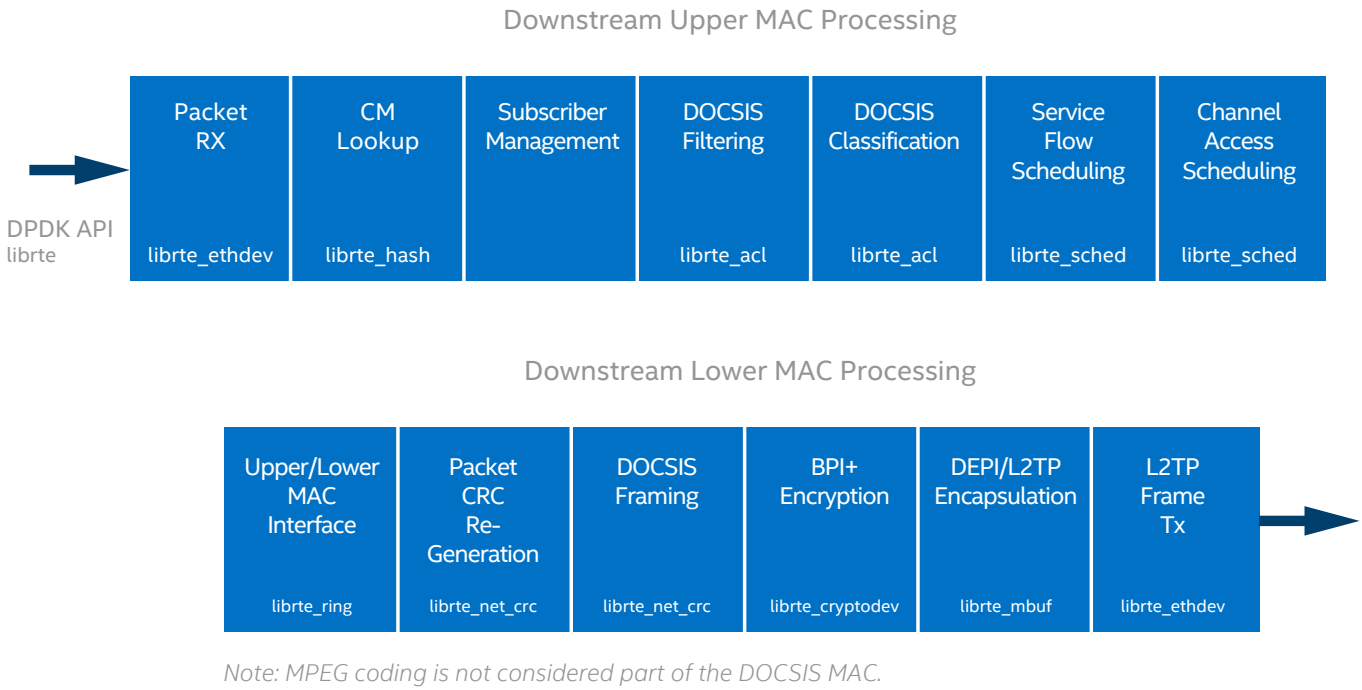

Figure 2. The Intel-Developed Downstream Data Plane Pipeline Prototype

Lower MAC Processing Stages

8) Lower MAC Interface
A DPDK ring is used to transfer packets between upper MAC and lower MAC threads.

9) Packet CRC Re-Generation
If required, the 32-bit Ethernet cyclic redundancy code (CRC) of the packet is regenerated using the DPDK CRC API. This is an optimized library based on Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions.

10) DOCSIS Framing
DOCSIS MAC headers are generated, including header check sequence, for prepending to packets. The DPDK CRC API is used to generate the DOCSIS header check sequence.

11) Baseline Privacy Interface (BPI+) Encryption
128-bit Advanced Encryption Standard (AES) encryption is performed on packets based on cable modem security association using DPDK crypto APIs for DOCSIS. This API supports AES and Data Encryption Standard (DES) BPI+ encryption for both software-only and hardware offload to Intel QAT.

12) Downstream External Phy Interface (DEPI) Encapsulation
DOCSIS frames are converted to Packet Streaming Protocol (PSP) segments, concatenated using DPDK mbuf chaining, and encapsulated into L2TP DEPI frames of maximum transmission unit size. PSP segments are fragmented across DEPI frames so all transmitted frames are of maximum transmission unit (MTU) size in order to ensure maximum utilization of the R-PHY link.

13) Packet Tx
Bursts of L2TP DEPI frames are transmitted to the NIC using the DPDK PMD.

Test Environment

The test environment consists of a vCMTS platform and software traffic generator, as shown in Figure 3.

The vCMTS platform is based on a server blade with dual Intel® Xeon® Platinum 8180 processors and three 4 x 10G Intel® Ethernet Converged Network Adapter X710-DA4 network interface cards (NICs).

The software infrastructure utilized Docker* containers to host the DOCSIS MAC processing for individual SGs, allowing them to be instantiated and scaled independently. To maximize performance determinism, each SG instance was pinned[3] to a single, isolated[4] processor core.

The traffic generation platform, equipped with an Intel® Xeon® processor E5-2699 v4 and three 4x10G Intel® Ethernet Controller X710-DA4 NICs, uses DPDK Pktgen to generate IPv4 and Logical Link Control (LLC) traffic of various packet sizes. The traffic is an Internet mix (IMIX) with a weighted average packet size of 998 bytes, comprising 84 byte packets (15%), 256 byte packets (8%), and 1,280 byte packets (75%). Destination MAC and IP addresses are crafted for cable modems, filters, and classifiers configured in vCMTS instances.

The vCMTS application configuration and associated packet generation modelled 300 modems per SG and an average of five IP addresses per subscriber. In addition, channel bonding was also modelled for the DOCSIS 3.0 channels with overlapping channel bonding groups.

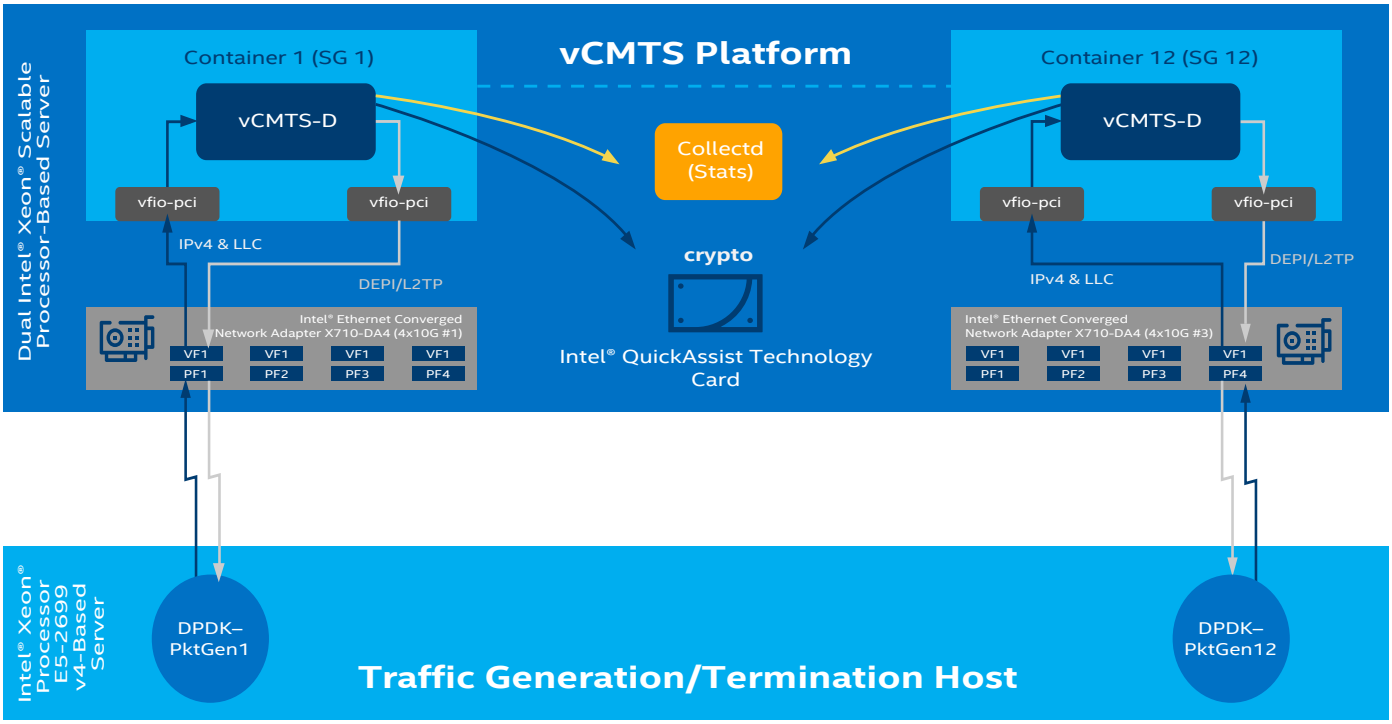

Figure 3. Test Environment Used to Measure vCMTS Performance

Results: Single Processor Throughput

Performance tests were run on the pipeline with various DOCSIS channel configurations. Figure 4 shows the single processor core throughput for two such example deployments over a range of packet sizes.

The first configuration (labeled "32xSC-QAM + 2xOFDM") represents an evolutionary step towards full DOCSIS 3.1 support and includes a mixture of DOCSIS 3.0 and 3.1 channels. The second configuration (labelled "6xOFDM") shows the potential performance for a pure DOCSIS 3.1 deployment with six OFDM channels; and in this case, the impact for crypto acceleration offload is also shown.

Throughput is shown for a range of packet sizes. With increasing packet size, the header-to-payload processing ratio declines, generally resulting in higher overall packet throughput. Software-based encryption and CRC regeneration (when needed) will require more computing resources for larger packet sizes.

With increasing packet size, the header-to-payload processing ratio declines, generally resulting in higher overall packet throughput. As shown in the graph above, a single core achieves the physical throughput limits for several DOCSIS channel configurations.

The IMIX bar shows the achievable throughput range when running the IMIX packet mixture. Although the calculated weighted average packet size is approximately 1,000 bytes, the actual measurements suggest the effective throughput is somewhere in the range of 800 to 900 byte packets.

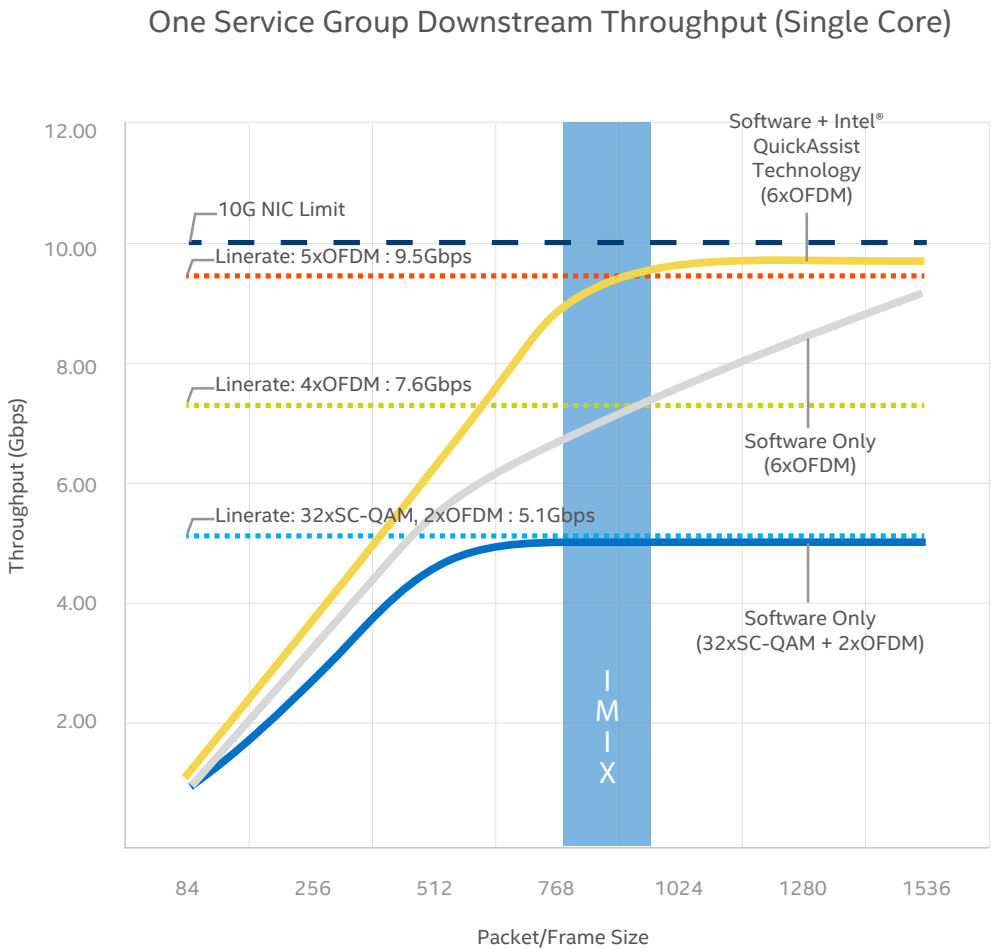

Figure 4. Single Processor Core Throughput for Various Packet Sizes[5,6]

32xSC-QAM+2xOFDM Configuration[7]

The first channel configuration scenario with 32 legacy channels (single-carrier quadrature amplitude modulation SC-QAM, DOCSIS v3.0 or earlier) and two OFDM channels, has a theoretical bandwidth limit of approximately 5.1 Gbps. As seen by the solid blue line in Figure 4, the per core performance for an un-accelerated MAC (i.e., no Intel QAT crypto acceleration) starts to saturate the channel configuration at packet sizes between 500 and 700 bytes. The IMIX bar confirms a core running at this frequency (2.5 GHz) is more than capable of handling the line rate for this channel configuration.

Figure 5a shows the CPU cycle consumption[8] for the downstream pipeline stages supporting 32xSC-QAM+2xOFDM channels when processing 1 KB packets on a single Intel Xeon processor core. Two scenarios are presented, one with and one without Intel QAT crypto acceleration, illustrating how the use of hardware offload can reduce the cost of encryption and total CPU cycles by almost 20 percent.[9]

Encryption, specifically AES-based BPI+ encryption in this case, is one of the two big CPU cycle consumers (approximately 20 percent) in this pipeline, with the other being service flow and channel access scheduling.

A key feature of software encryption is it can be characterized as a per byte cost, hence the cost of encrypting a packet is directly proportional to the size of a packet. For 1 KB packets, the cost for AES is about 1,500 bytes or 1.5 cycles per byte, assuming the use of Intel® AES New Instructions (Intel® AES NI) multi-buffer technology embedded in the DPDK crypto library. The library implements packet batching techniques to make optimal use of Intel AES-NI.

AES supersedes the legacy DES, which is no longer used for new cable deployments; however, DES is used in older versions of DOCSIS, which runs on approximately 10 to 20 percent of devices today. The amount of software-based DES processing is an important platform consideration since it can require 20 to 30 times more computing resources than software-based AES processing. Intel plans to provide an optimized implementation of DES implementation using the Intel AVX512 instructions that will be several times faster than existing software libraries, but it will still be close to an order of magnitude more expensive (i.e., in CPU cycles) than software AES.[12]

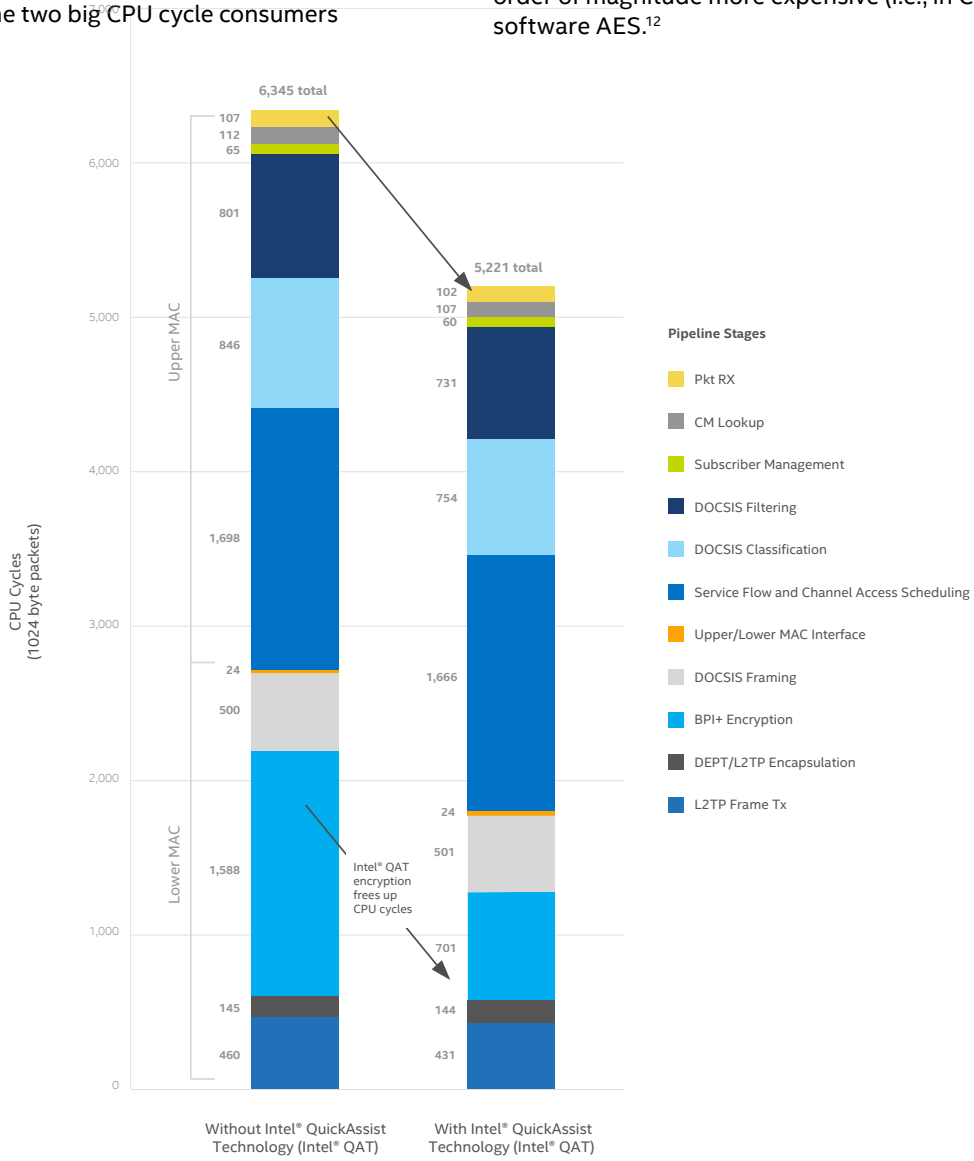

Figure 5a. Downstream Data Plane Pipeline Supporting 32xSC-QAM+2xOFDM: Processor Cycles[10,11]

The second technology that can be used for encryption acceleration is Intel QAT integrated into the Intel® C620 Series Chipset family or added to a system via a PCI add-in Card. This technology performs hardware-accelerated crypto functions and effectively reduces the CPU cycle cost of encryption regardless of packet size or encryption type (AES or DES). In this pipeline measurement, the cost is 668 cycles, which is less than half the cost of Intel's software-only implementation.[13]

All that said, the impact of Intel QAT crypto acceleration is not shown in Figure 4 as it is negligible for this scenario. Software-based AES encryption using Intel AES NI is sufficient to support the data rate. On the other hand, it is very important to consider if an SG configuration requires legacy DES support. Intel QAT encryption acceleration can be used to maintain the blue line curve shown because the cost of AES and DES encryption amounts to the same static, fixed value when offloaded to Intel QAT.

The second largest consumer of cycles (approximately 20 percent) is service flow and channel access scheduling. At 1,024 byte packets, the scheduler kept all channels filled with enough packets to saturate channel bandwidth. Note that the scheduler cost of around 1,600 cycles is a peak cost due to the inefficiency of the implementation and the level of scheduler complexity required for the legacy 32xSC-QAM channels, as shown in Figure 5a. When compared to scheduler cost below the saturation point, the scheduler consumes significantly more cycles once the channel is saturated. This observation is left for further study. Service flow and channel access scheduling has the potential to be a hotspot in many implementations. Future work will take a look at how this area can be optimized.

6xOFDM Configuration[14]

The second channel configuration is a pure DOCSIS 3.1 deployment with six OFDM channels that produce a theoretical cumulative bandwidth of 11.3 Gbps; however, the effective bandwidth is limited to 10 Gbps by the Ethernet NIC port itself. The un-accelerated MAC (Software Only) curve indicates performance can achieve 7 Gbps at IMIX. The accelerated curve, benefitting from Intel QAT encryption acceleration, is much steeper than the software only curve, achieving a higher 5xOFDM line rate saturation somewhere between 800 and 900 byte packets. At IMIX this represents a 33 percent improvement in throughput.

Figure 5b shows the cycle cost breakdown for the configuration with 6xOFDM channels and no SC-QAM channels. Noteworthy is the scheduler uses approximately

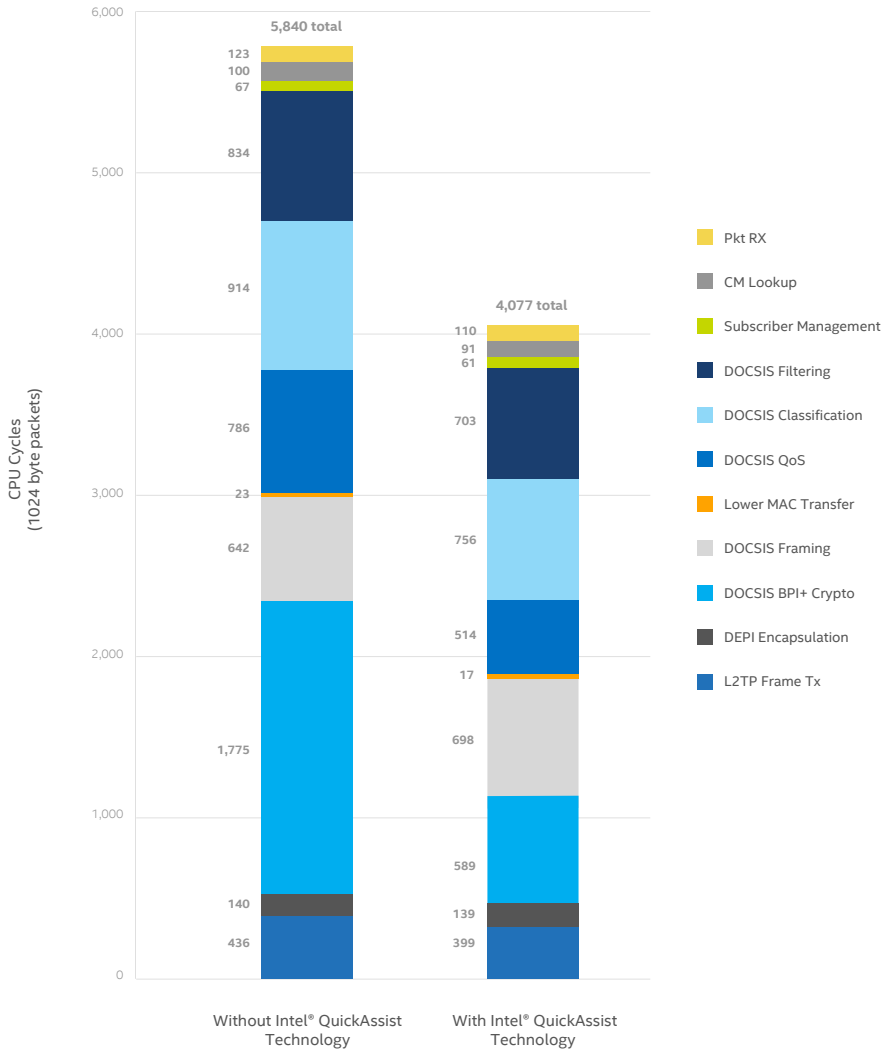

Figure 5b. Downstream Data Plane Pipeline Supporting 6xOFDM channels: Processor Cycles[15,16]

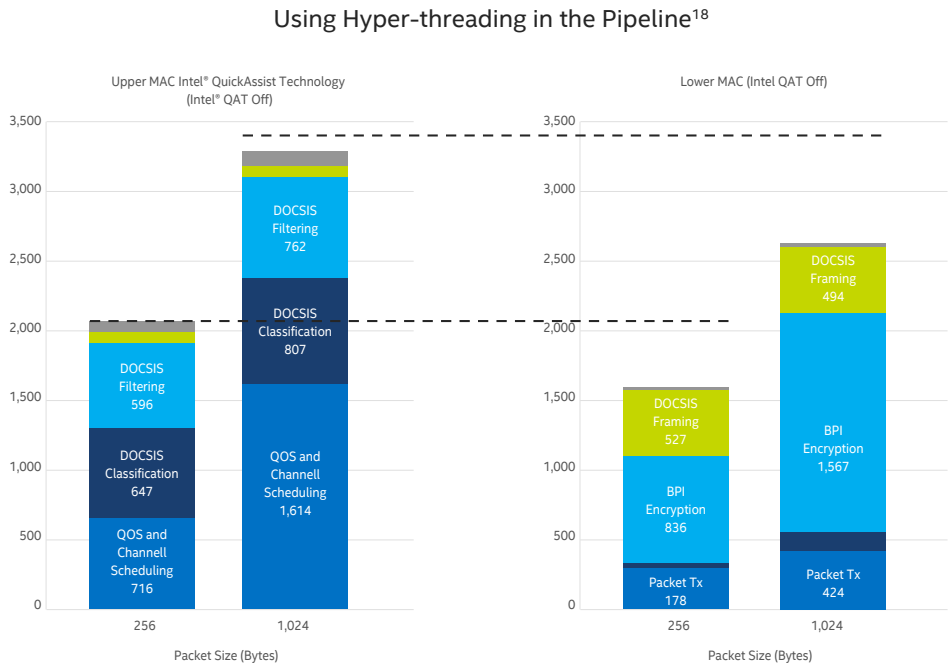

Figure 6. CPU Cyle Comparison of Upper and Lower MAC for Processing 32xSC-QAM+2xOFDM Channels[18]

50 percent fewer CPU cycles than the 32xSC-QAM+2xOFDM configuration, which has a positive impact on the overall cost of processing a packet by reducing it below 6,000 cycles. The impact of Intel QAT is more pronounced in 6xOFDM channels channel configuration than the 32xSC-QAM+2xOFDM configuration, reducing the overall packet cost by about 30 percent.

Note also that Intel QAT crypto acceleration has a positive impact on the cycle cost of several other functional blocks.

Using Hyper-threading in the Pipeline[17]

Figure 6 shows the split of functionality between the upper and lower MAC, and the associated per functional block cost for two different packet sizes processed for 32xSC-QAM+2xOFDM channels. For both packet sizes, the upper MAC is cumulatively more costly than the associated lower MAC. The impact of this is seen when the upper and lower MAC are deployed on sibling hyperthreads (i.e., hyper-threads of the same processor core), as they are in all the data provided in this document. Since the upper MAC takes longer to execute than the lower MAC, it determines the throughput per service group, assuming there are no other limiting factors, such as port speed or channel bandwith configuration. For example, 256 byte packets cost approximately 2,000 cycles for the upper MAC, at 2.5 GHz, this equates to (2.5/2000)*256 bytes *8=2.6 Gbps, which is inline with the measured throughput for 256 byte packets shown in Figure 4. For 1,024 byte packets, the same calculation suggests a throughput of 6 Gbps; however, the channel configuration limits the platforms throughput to 5.1 Gbps.

For the 32xSC-QAM+2xOFDM configuration with AES, the lower MAC execution cost is effectively hidden for the packet sizes shown when it is running on the second thread of the same core. Interestingly, encryption offloading or further optimization of the the lower MAC, in this case, has no affect on performance. However, if the cost of the lower MAC were to exceed the upper MAC, as when there is a significant amount of DES traffic; or if the upper MAC becomes less costly than the lower MAC, as when processing large packets for OFDM channels; then optimizing of the lower MAC would have an impact (refer to Figure 4 and figure 5b).

Results: System Scalability and Server Sizing

For cable system architects who wish to size server requirements, Figure 7 shows platform DOCSIS 3.1 (6xOFDM with AES encryption) throughput as it scales with the number of SGs when passing IMIX traffic. The SGs are supported on 1 to 12 processor cores, residing on one of the two processors on this dual processor platform. For up to eight cores, the platform performance scales linearly.

Using the information in Figure 7, it is possible to calculate how many Intel Xeon processor cores are needed for a wide variety of cable system requirements.

When dimensioning a system, other workloads should be taken into account. For instance, the operating system, upstream scheduler, control plane, telemetry logging processes, high availability monitors, and other functionality may need to be hosted on the platform, thus consuming more processor cores.

Example of Platform Dimensioning

In order to calculate the performance density of a platform, it is important to account for all the CPU resources consumed by all the applications and services needed to host the virtualized application. Platforms and implementations will differ, but the following provides a simple example of how this might be done at a very basic level:

1. CPU/core allocation for non-data plane platform elements will consume approximately four cores. The following is a selection (not an exhaustive list) of some of the elements that should be included:
   a. Operating system
   b. Orchestration framework and high availability infrastructure
   c. Platform telemetry and service assurance
   d. Application control plane VNFs
2. CPU allocation to data plane elements – consuming approximately 1.5 cores
   a. Downstream data plane for a single SG – approximately one core
   b. Upstream data plane and upstream scheduler for a single SG – approximately half a core (e.g., one hyper-thread)

Note: *These estimates are basic rules of thumb given the compute needs of a SG data plane can vary considerably for each physical deployment, and it may be possible to deploy more than one SG to a core if the SGs are lower speed.*

The following applies these rules to a dual processor platform populated with an Intel® Xeon® Gold 6148 processor, running at 2.4 GHz with a total of 40 cores and 12[20] PCIe* x8 Gen3 ports.

- Allocate four cores (per #1 above) to non-data plane tasks, leaving 36 cores
- Allocate 36 cores to SG's data plane (i.e., 1.5 cores per SG), supporting 24 SGs

The scalability graph in Figure 7 suggests 12 cores of a single processor system can support approximately 80 Gbps at IMIX packet mix without the use of Intel QAT crypto acceleration. Assuming a frequency differential of 4 percent or 100 MHz is negligible and no impact of reduced quantity of L3 cache (in line with lower core count), the system could support 160 Gbps without acceleration and consume a minimum of four of the 12 PCIe x8 ports for NIC interfacing, depending on the chosen NIC redundancy model.

If Intel QAT crypto acceleration is employed, 12 cores is likely to exceed 100 Gbps for IMIX, yielding a platform throughput of 200 Gbps. To support this throughput, it would be necessary to add four more PCIe x8 ports to handle the encryption traffic, bringing the total used port count to eight. To achieve a throughput of greater than 100 Gbps per processor, an additional four PCI slots (two NICs and two crypto accelerator cards) would be required.

Scalability of vCMTS Architecture

More and more network architects are adopting software-based solutions running industry-standard, high-volume servers to increase agility, flexibility, and cost competitiveness. These benefits now present themselves with an architecture that offers a high level of scalability and helps avoid vendor lock-in associated with proprietary, hardware-based solutions. This scalability is possible because the vCMTS data plane VNFs running on each core are relatively independent from each other; and thus, more SGs can be handled by spinning up new VNF instances on available processor cores. When no cores are available, another server can be easily added to the chassis to increase capacity.

This paper demonstrates how the scalability of Intel Xeon processor-based platforms can be applied to today's mix of channels (SC-QAM and OFDM) as well as future OFDM-focused implementations. With this data, network architects can better assess the hardware cost of implementing a vCMTS on industry-standard, high-volume servers.

For more information on networking solutions using Intel® technologies, please visit networkbuilders.intel.com

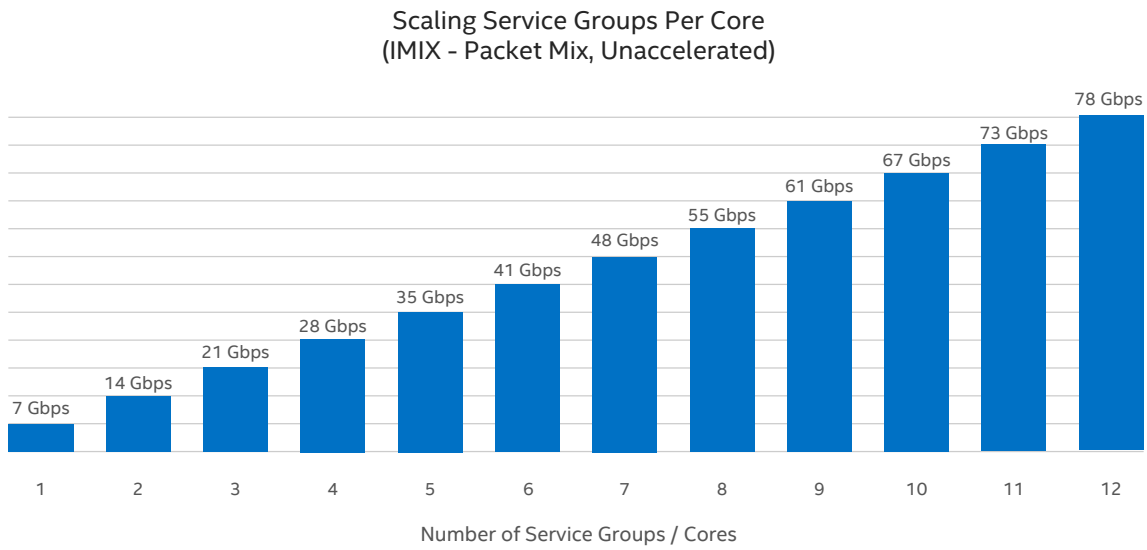

Figure 7. Platform Throughput for IMIX and Different Number of Service Groups[19]

Appendix A: System Configuration

vCMTS Server	
Hardware	
Platform	Intel® Server Board S2600WFT
CPU	Intel® Xeon® Platinum 8180 Processor, 2.5 GHz, 28 Cores
Memory	12x8GB DDR4
Hard Drive	Intel® SSD DC S3520 Series (480G)
Network Interface Card	3 x Intel® Ethernet Converged Network Adapter X710-DA4 Quad-Port 10GbE
Crypto Acceleration Card	2 x Intel® QuickAssist Adapter 8970
Software	
Host OS	Fedora 24, Linux* Kernel v4.11.12
Linux Container	Docker* v1.10.3
DPDK	DPDK v17.08
vCMTS	Intel® vCMTS-d v17.10

vCMTS Traffic Generator	
Hardware	
Platform	Intel® Server Board S2600WTTR
CPU	Intel® Xeon® DP E5-2699 v4, 2.2 GHz, 22 Cores
Memory	8 x 8GB DDR4
Hard Drive	1TB SATA Seagate* HDD
Network Interface Card	3 x Intel® Ethernet Converged Network Adapter X710-DA4 Quad-Port 10GbE
Crypto Acceleration Card	2 x Intel® QuickAssist Adapter 8970
Software	
Host OS	Fedora 24, Linux* Kernel v4.11.12
DPDK	DPDK Pktgen v3.3.8
Traffic Generator	Fedora 24, Linux* Kernel v4.11.12

Appendix B: Test Environment Configuration Information and Relevant Variables

CM Lookup & Subscriber Management	300 Subscribers, average of five devices (or IP addresses) per subscriber
DOCSIS Filtering	10% matched (permit traffic), 90% unmatched
DOCSIS Classification	10% matched (enqueue to one of three specific service-flow queues), 90% unmatched (enqueue to default service-flow queue)
Service-Flow Scheduling	16 service-flow queues per subscriber (four active)
Channel Scheduling	2xOFDM (1.89Gbps) and 32-SC-QAM (42.24Mbps) Channels or 6 x OFDM (1.89 Gbps) Channels
Packet CRC	0% re-generated, 100% not re-generated
Encryption	100% AES, 0% DES

Appendix C: Acronyms and Definitions

Acronym or Term	Definition
AES	Advanced Encryption Standard
BPI	Baseline Privacy Interface
CM	Cable Modem
CMTS	Cable Modem Termination System
CRC	Cyclic Redundancy Code
DEPI	Downstream External Phy Interface
DES	Data Encryption Standard
DOCSIS	Data Over Cable Service Interface Specification
MAC	Media Access Control
R-PHY	Remote Phy
vCMTS	Virtualized Cable Modem Termination System

[1] Intel test results using the system configuration shown in Appendix A.

[2] Hyper-threaded siblings are hardware threads of execution contained within the core sharing of the same set of core resources. Each hyper-thread run its own software thread.

[3] Pinning process to a core is an operating system method to ensure a process always executes on a specific processor core and is never migrated to another core even if those other cores are idle. This avoids costly core context switch costs (e.g., cache refills, TLB refills) and improves determinism

[4] Isolated cores are processor cores that are excluded from the Linux* scheduler. As such, these cores run a single process, which is never swapped out. Likewise, these cores are typically not included in the interrupt load balancing pool.

[5] Intel test results using the system configuration shown in Appendix A.

[6] Internet mix (IMIX) downstream traffic profile: Weighted average packet size is 998 bytes, comprising 84 byte packets (15%), 256 byte packets (8%), and 1,280 byte packets (75%).4. Running a complete data plane on a single core is not always possible due to nature of the application. For example, the scale and complexity of the application pipeline may require it to be partitioned across cores.

[7] Intel test results using the system configuration shown in Appendix A.

[8] Intel test results using the system configuration shown in Appendix A.

[9] Intel test results using the system configuration shown in Appendix A.

[10] Intel test results using the system configuration shown in Appendix A.

[11] Note the test environment is a loaded system from which measurements are taken during execution. Inherently, there will be a margin of error due to the action of taking the measurements. Results shown are for a reference implementation of the pipeline and not a production MAC. Functional costs will vary due to different implementations and deployment threading models. These numbers should be treated strictly as a reference only.

[12] Intel test results using the system configuration shown in Appendix A.

[13] Intel test results using the system configuration shown in Appendix A.

[14] Intel test results using the system configuration shown in Appendix A.

[15] Intel test results using the system configuration shown in Appendix A.

[16] Intel test results using the system configuration shown in Appendix A.

[17] Intel test results using the system configuration shown in Appendix A.

[18] Intel test results using the system configuration shown in Appendix A.

[19] Intel test results using the system configuration shown in Appendix A.

[20] Each Intel® Xeon® 6148 Scalable processor has 48 lanes of PCIe Gen3, and consequently a dual processor system has 96 PCIe lanes. This equates to 12 x8 lanes. See http://mark.intel.com/products/120489/Intel-Xeon-Gold-6148-Processor-27_5M-Cache-2_40-GHz -

[21] Note that a Channel Access Scheduling patch to the DPDK Scheduler was developed for the purpose of the vCMTS-d prototype.

© 2017 Intel Corporation Printed in USA 1017/FS/ICMCSW/PDF ♻ Please Recycle 336616-001US