VIRTUAL EXPERIENCE
OCTOBER 11-14

2021 Fall
Technical Forum
SCTE® • NCTA • CABLELABS®

# Considerations for Moving Your Access Network to the Cloud (and Back)

A Technical Paper prepared for SCTE by

**Michael O'Hanlon**
Senior Principal Engineer
Intel Corporation – Network Platforms Group
Dromore House, Shannon, Co. Clare, Ireland
+353 (86) 354 3536
michael.a.ohanlon@intel.com

**Eric Heaton**
Platform Solutions Architect
Intel Corporation – Network Platforms Group
2200 Mission College Blvd., Santa Clara, CA
1-408-765-3447
eric.d.heaton@intel.com

**Randy Levensalor,** CableLabs

**Brendan Ryan,** Intel Corporation – Network Platforms Group

**Richard Walsh,** Intel Corporation – Network Platforms Group

**David Coyle,** Intel Corporation – Network Platforms Group

**Pavel Belitskiy,** Intel Corporation – Network Platforms Group

**Subhiksha Ravisundar**, Intel Corporation – Network Platforms Group

# Table of Contents

**Title** **Page Number**

## List of Figures

**Title** **Page Number**

# List of Tables

**Title** **Page Number**

# 1. Introduction

The Cable industry continues to innovate on both technical and business fronts in their Access and Edge networks to meet and exceed ever-ascendant customer demands. Currently, Distributed Access Architecture, DOCSIS 4.0, Generic Access Platform, fiber deep, Wireless Access, and virtualization are all being deployed in various permutations in order to deliver on the promises of the 10G era – high bandwidth, low latency, seamless fail-over, single user profiles, and so forth. Looking at the evolution of the Cable Modem Termination System (CMTS) specifically, what was once a fixed, vertically integrated, HFC CCAP appliance from a single vendor has become a virtualized CMTS (vCMTS) appliance running on commercial off the shelf (COTS) servers. This transformation into the software domain has brought the agility needed to keep up with other upgrades to the network.

Along these lines, the vCMTS is now being broken into smaller component functions packaged in software containers that now can be deployed, managed, and scaled individually. In fact, some of those functions may be shared across other network functions or Access technologies to achieve some aspects of convergence and cost savings – for example, a DHCP server or a network telemetry database. The flexibility of such a decomposed, cloud-ready, vCMTS solution brings a lot of opportunities for the MSO in terms of hardware or software consolidation, placement of a workload to the "best" place in the network, individual scaling, and general positive reactivity to the real time needs of users or the operator.

These characteristics are also the promise from Cloud Service Providers (CSPs) when deploying a given application on their infrastructure. That is, they will manage the hardware, furnish a consistent pane of glass for management, and transparently provide redundancy and scale. This certainly sounds appealing! In fact, the Communications Service Provider (CoSP) industry is already familiar with moving enterprise workloads to the Cloud; but will this approach work for vCMTS – and the DOCSIS MAC dataplane in particular?

Figure 1 shows how a vCMTS solution could be deployed to Cloud infrastructure[1] and tie into the general MSO network. This approach moves the vCMTS functionality from equipment in the Headend/Hub in the RPD case or the Node in the RMD case owned by the network operator to servers and switches deeper in the network owned by a CSP. The upside is that the CSP handles the deployment and management of this hardware and provides a common software infrastructure, distributed backup capability, and other operational benefits to the network operator.

[1] Cloud can refer to Cloud Service Provider hardware located in the cloud providers datacenter or within a Telco or MSO providers premises. Section 1.1**Error! Reference source not found.** outlines various possible locations for cloud hardware.

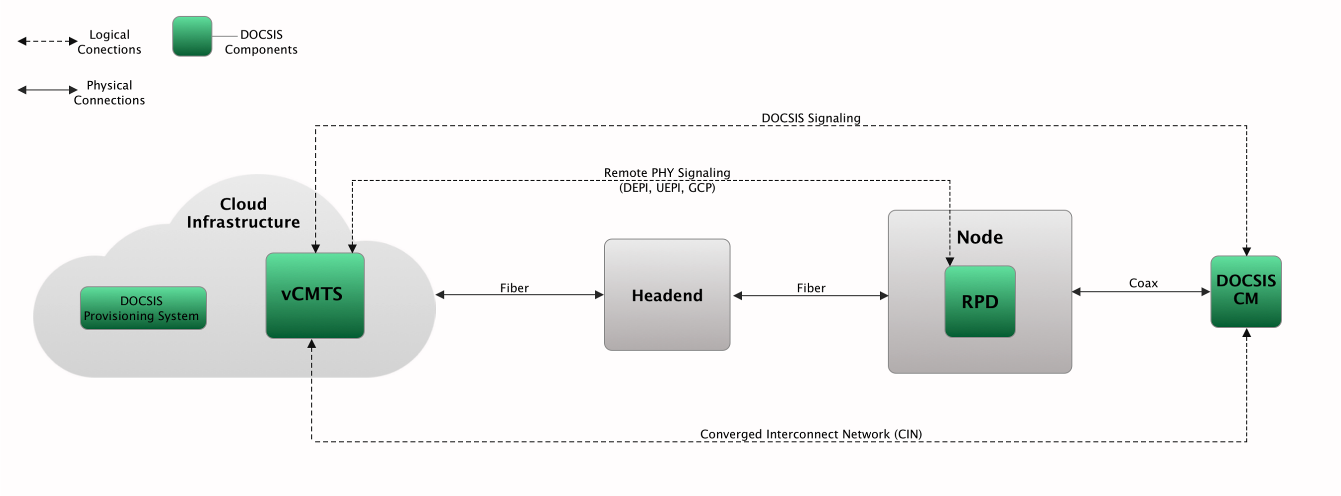


**Figure 1 – vCMTS in the Cloud**

For all the benefits, there are some trade-offs versus having the vCMTS running on local servers and switches in the Headend/Hub. While there are business and strategic considerations for moving a workload to the cloud – for example, costs, core competencies, and competitive circumstances – this paper focuses on technical questions of expected vCMTS dataplane performance and comparison to the same applications running on on-premise equipment. Just to be clear, the benchmarking and subsequent analysis described in this paper only looks at considerations for the DOCSIS dataplane as opposed to control plane elements of a full vCMTS solution (ex. availability of IEEE 1588), which is, itself, a topic ripe for further discussion.

- Section 1 lays out the test infrastructure and process used to evaluate the current Cloud offerings specifically for networking workloads like a vCMTS dataplane
- Section 2 presents the initial single Service Group benchmarking results across several cloud instance types and makes comparisons to on-premise solutions
- Section 3 describes considerations for effective and performant scaling of the vCMTS dataplane within the Cloud to work-around any per-instance performance limits
- Section 4 summarizes this testing and infrastructure analysis and recommends an "ideal cloud instance type" for the vCMTS dataplane or other similar Access workloads

In short, this paper will describe how a vCMTS dataplane workload can be run optimally in the Cloud, lay out benchmarks comparing various cloud products to on-premise solutions, provide a recommendation for an ideal instance type, and detail the full process from start to finish to allow this work to be generalized across other CSP offerings and workloads in the future.

## 1.1. Definition of the Cloud

A cloud instance is essentially a virtual machine that is hosted and managed by a CSP and can be rented to users. Each instance is defined by a set of properties, such as the number of virtual CPUs (including those with specific capabilities or instruction sets), the amount of memory or storage, accelerator options (ex. GPUs, custom ASICs, etc.), and network connectivity. This analysis focuses solely on the offerings from Amazon Web Services (AWS) as being representative of the offerings from other CSPs.

The cloud products from AWS only continue to grow in number and to help customers determine the right match for their needs, AWS groups them into the families shown in Figure 2. Each family, and in many cases, sub-family (not shown), has a general set of workload requirements in mind, such as being optimized for machine learning algorithms or having large storage but low compute requirements. These attributes then translate into the actual instance configuration for compute, hardware accelerators, storage capability, number and size of network interfaces, and so forth.

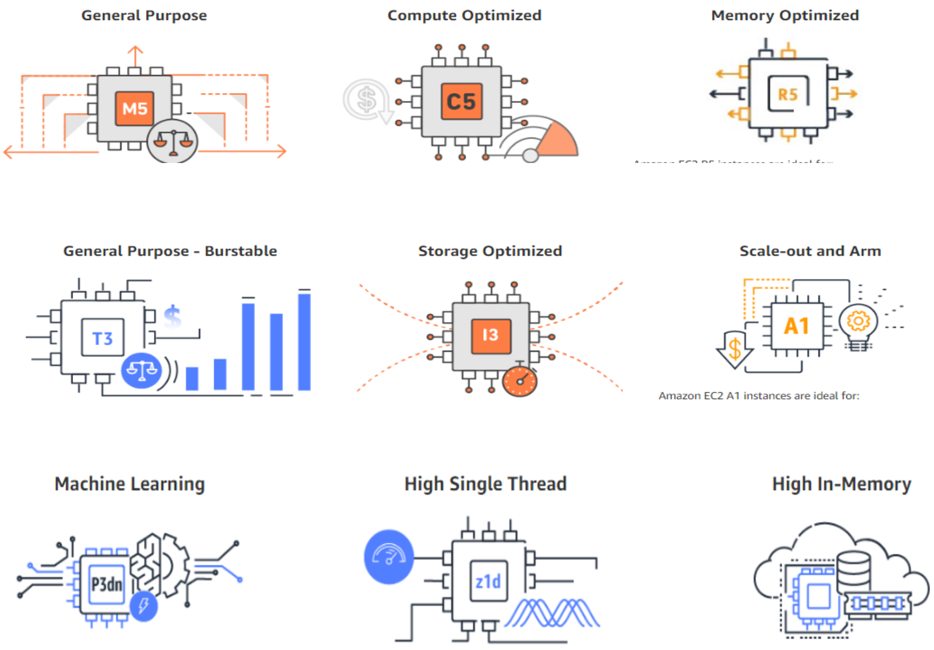
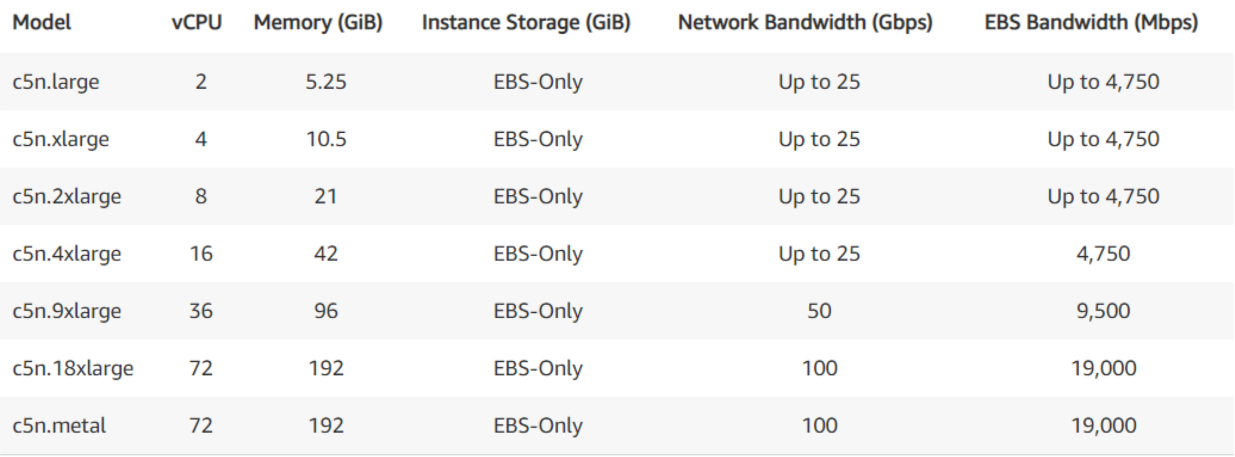


**Figure 2 – AWS Instance Families[2]**

The first criteria to filter the selection is knowing that vCMTS dataplane reference software used for this analysis was written and optimized for an x86 architecture and thus it will run best on instances based on the newest possible x86 processors with AVX-512 and AES-NI instructions. The second criteria is knowing that an on-premise server running this application can saturate 100Gbps per socket of DOCSIS traffic and thus would like to see instances with very high network connectivity.

Reviewing the options – and performing some baseline testing – the Compute Optimized family is most suitable for NFV generally and the vCMTS dataplane specifically because of its powerful compute capabilities. Within the Compute Optimized family is a sub-family known as "c5n" (the "n" is for networking), which pairs x86 compute from 1st Gen Intel® Xeon® Scalable Processors with up to 100Gbps of network bandwidth per instance. The instance configurations within this sub-family are shown in Figure 3.

[2] https://aws.amazon.com/ec2/instance-types/



| Model | vCPU | Memory (GiB) | Instance Storage (GiB) | Network Bandwidth (Gbps) | EBS Bandwidth (Mbps) |
|---|---|---|---|---|---|
| c5n.large | 2 | 5.25 | EBS-Only | Up to 25 | Up to 4,750 |
| c5n.xlarge | 4 | 10.5 | EBS-Only | Up to 25 | Up to 4,750 |
| c5n.2xlarge | 8 | 21 | EBS-Only | Up to 25 | Up to 4,750 |
| c5n.4xlarge | 16 | 42 | EBS-Only | Up to 25 | 4,750 |
| c5n.9xlarge | 36 | 96 | EBS-Only | 50 | 9,500 |
| c5n.18xlarge | 72 | 192 | EBS-Only | 100 | 19,000 |
| c5n.metal | 72 | 192 | EBS-Only | 100 | 19,000 |

**Figure 3 – Instance options within the c5n family**[3]

All c5n instance types run on a single c5n server type that includes dual 18 core 1st Gen Intel® Xeon® 8124M Scalable Processors running at 3.0GHz (with up to 3.5GHz Turbo operation; Turbo cannot be disabled), 192GB of DDR4 memory, and a 100GBE network interface, as shown in Figure 4. The largest instances, c5n.18xlarge and c5n.metal, consume a complete server and hence are not co-resident with other c5n instances. Smaller instances consume sub-portions of the server resources and may be co-resident with other instances. Figure 5 shows the inverse relationship to the instance vCPU count to the number of such instances that will fit on a given c5n physical server. In other words, divide the total vCPU count of the hardware (72) by the number of vCPUs in an instance to calculate how many of the same instance can theoretically be deployed there.

[3] https://aws.amazon.com/ec2/instance-types/

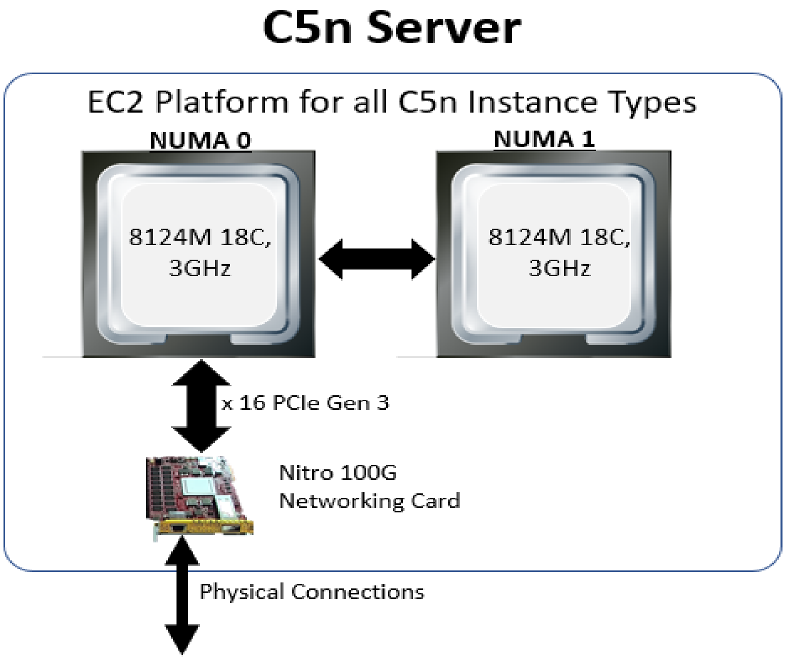


**Figure 4 – AWS c5n server definition**

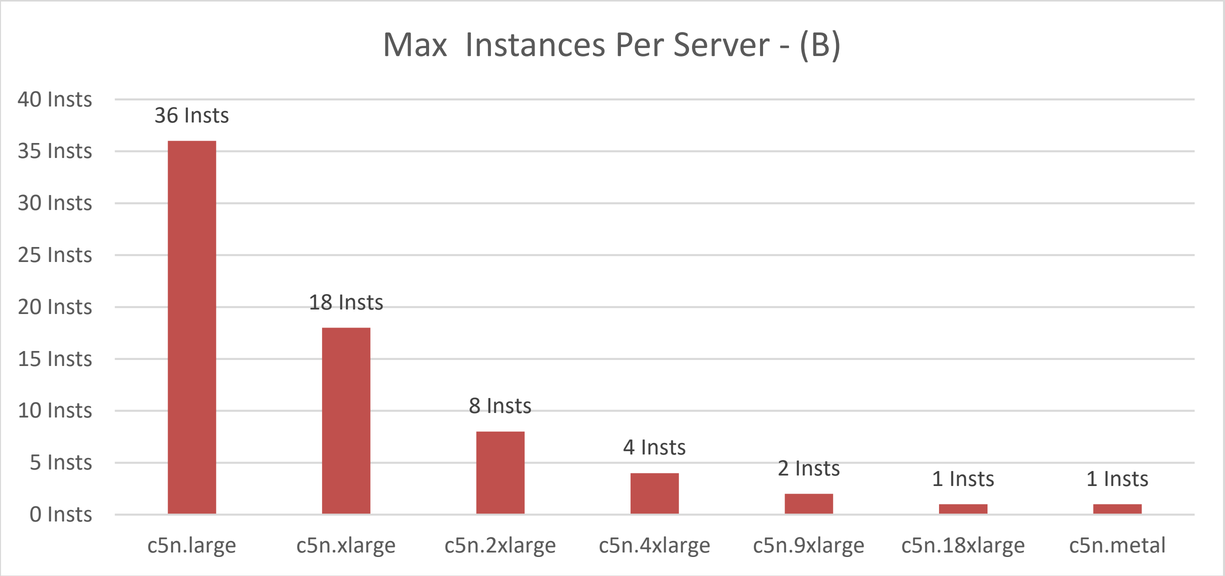


**Figure 5 – Max number of instances per c5n server**

Figure 3 is helpful to get a high-level understanding of the compute and memory resources per instance, however, what else can be said about the network bandwidth? This is key to set performance expectations correctly and perhaps even anticipate bottlenecks for a dataplane application. For example, is the defined bandwidth shared among other instances, are there limitations on packets per second, are there any common packet offload features, how many ports/interfaces can be attached to an instance?

The network interfaces for the c5n family are provided through their own custom AWS Nitro System, which is a "combination of dedicated hardware and lightweight hypervisor[4]". This system includes a set of I/O acceleration cards for Virtual Private Cloud (VPC) and various storage options, a controller card that establishes a root of trust and interfaces to the control plane, and a few other elements designed for secure, accessible, and performant access to all cloud resources.

The main I/O card for network connectivity is called the Elastic Network Adapter (ENA), as shown in Figure 6. The ENA is a custom 100Gbps NIC supporting up to 15 physical interfaces (similar to VFs) per instance and up to 32 packet queues[5] per interface.

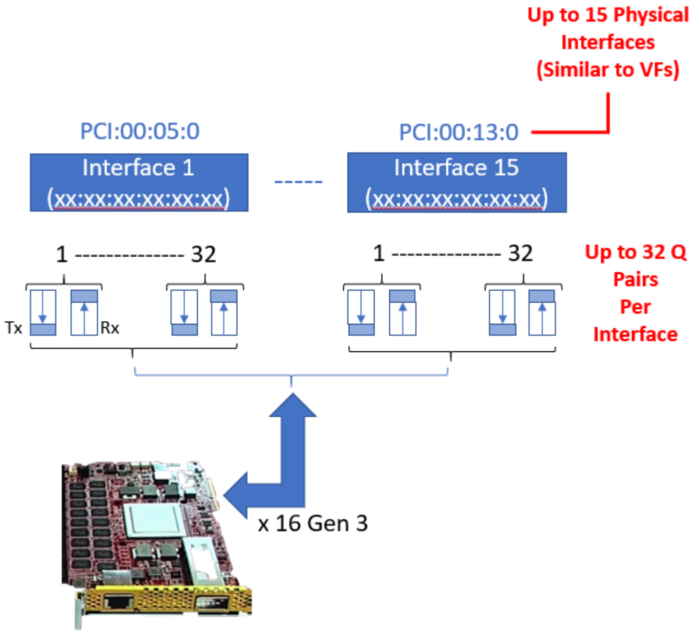
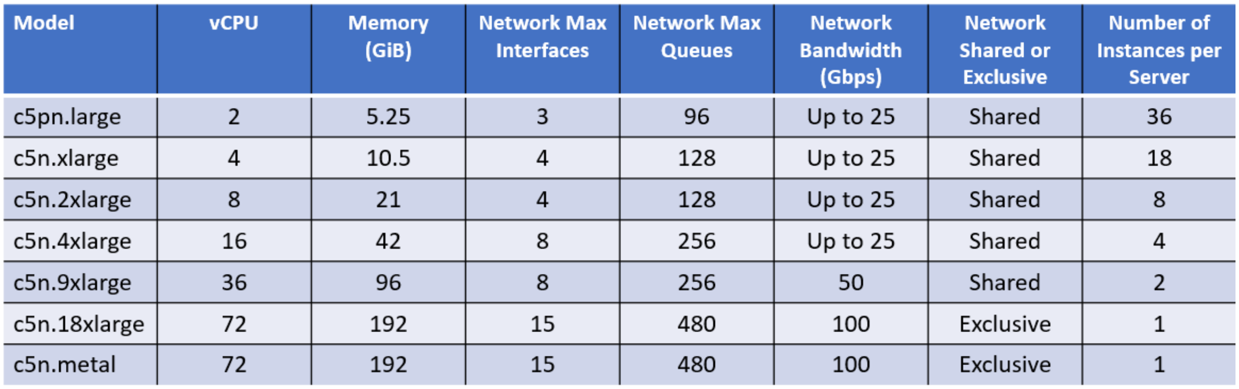


**Figure 6 – Network connectivity via AWS Nitro ENA – c5n.18xlarge example**

Even though all the c5n instance types make use of this 100Gbps ENA hardware, they do not have equal access to the bandwidth, the number of interfaces, the number of queues, or even if they need to share the bandwidth with other instances or not. Instances with less vCPUs and memory will have less networking bandwidth, less network interfaces (VFs), and less packet queues. This is important to note because of per-interface and per-queue performance limits that are discussed during later analysis.

Taking the public c5n instance listing and adding these key aspects for understanding the networking-related feature set yields the updated table shown as Figure 7.

[4] https://aws.amazon.com/ec2/nitro/
[5] https://aws.amazon.com/blogs/aws/new-c5n-instances-with-100-gbps-networking/



| Model | vCPU | Memory (GiB) | Network Max Interfaces | Network Max Queues | Network Bandwidth (Gbps) | Network Shared or Exclusive | Number of Instances per Server |
|---|---|---|---|---|---|---|---|
| c5pn.large | 2 | 5.25 | 3 | 96 | Up to 25 | Shared | 36 |
| c5n.xlarge | 4 | 10.5 | 4 | 128 | Up to 25 | Shared | 18 |
| c5n.2xlarge | 8 | 21 | 4 | 128 | Up to 25 | Shared | 8 |
| c5n.4xlarge | 16 | 42 | 8 | 256 | Up to 25 | Shared | 4 |
| c5n.9xlarge | 36 | 96 | 8 | 256 | 50 | Shared | 2 |
| c5n.18xlarge | 72 | 192 | 15 | 480 | 100 | Exclusive | 1 |
| c5n.metal | 72 | 192 | 15 | 480 | 100 | Exclusive | 1 |

**Figure 7 – Adding ENA functionality to c5n instance definition**

Finally, the software associated with the AWS Nitro system is generally lightweight since much of the control, management, and security of resources is done in the hardware of the various IO cards. That said, benchmarking shows notable differences in the performance of the standard ENA driver versus a standard DPDK-based PMD NIC driver and will be explored later.

Knowing how ENA resources are allocated to a given instance type along with ENA driver behavior will help explain the varied results across vCMTS benchmarking test configurations presented in subsequent sections. For example, is performance capped due to the compute capabilities of the instance, a bottleneck to and from memory, oversubscribing the ENA adapter in some way, or some other reason?

By definition, a networking application implies that data is being transported from one place to another, being “processed” (i.e., in this case, in the aforementioned cloud instances), and then, in most cases, sent somewhere else to provide some overall functionality. For this type of system, the topology of the network, complexity of the data transport layer, and location of compute resources, will all contribute to the expectations and limitations of deterministic behavior and a low end-to-end latency.

Early cloud options may have been limited, with servers and switches hosted in a far-off datacenter, but there are many more options today. As their real estate and infrastructure footprint has grown to accommodate a wide variety of workloads and markets, AWS offers compute, network, storage, and service products that can be located pretty much anywhere – including on a customer premise in various Enterprise and IOT form factors. Figure 8 shows several of the options relevant for Access network applications. The closer one gets the equipment and processing to end users, the lower possible latencies and higher potential for determinism for the application, but also the less possible abstraction and higher costs for hosting and management.

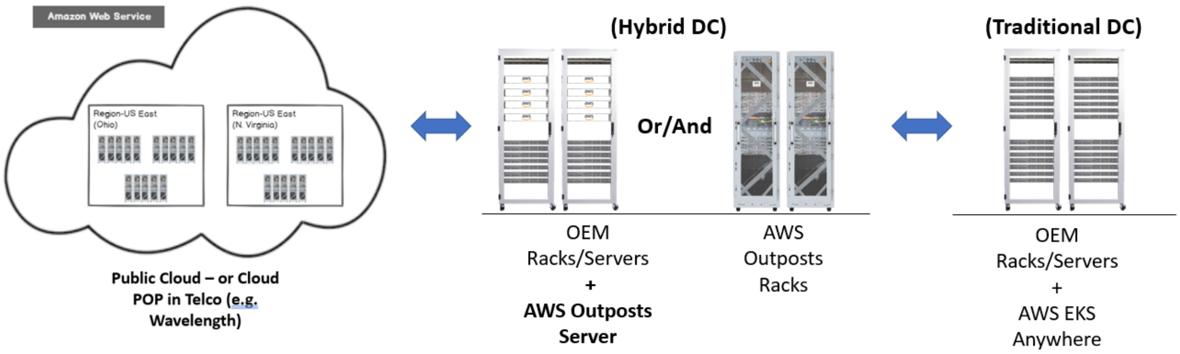

Amazon Web Service
Region-US East (Ohio)
Region-US East (N. Virginia)
Public Cloud – or Cloud POP in Telco (e.g. Wavelength)
(Hybrid DC)
Or/And
OEM Racks/Servers + AWS Outposts Server
AWS Outposts Racks
(Traditional DC)
OEM Racks/Servers + AWS EKS Anywhere

**Figure 8 – AWS application continuum**

AWS Regions define a physical location in the world where AWS clusters data centers[6]. Each Region is made up of so-called Availability Zones (AZs), which are one or more data centers logically connected for multiple layers of load balancing and redundancy. Some regions include ancillary Local Zones (LZs), which place AWS infrastructure and services even closer to population centers, enterprise customers, and IT centers to allow for single-digit millisecond response times. These concepts together allow for the scale, high availability, and relative localization CSP offerings are prized for and fit the needs of most applications.

But vCMTS or other Access dataplane workloads have more difficult-to-achieve requirements in terms of latency, jitter, and overall throughput than a typical Enterprise application. AWS is responding to this market by offering additional cloud deployment options. The first is called AWS Wavelength, which installs and maintains AWS equipment right inside the Edge Data Centers, Next Generation Central Offices, or Headends/Hubs owned by Communications Service Providers. This is a co-location network edge deployment model[7] and has "better together" benefits for the CSP and the network operator in being able to offer extremely low latency hosting or services for end users.

The second option, known as AWS Outposts, are packaged AWS hardware with AWS management infrastructure delivered onsite to a customer. The hardware could be a single server or a set of full racks, made for any instance type. From the technical perspective, AWS Wavelength is, perhaps, a scaled version of AWS Outposts that also includes an explicit agreement to resell or co-resell the use of this infrastructure to Enterprise customers as opposed to using to support the network directly. Given this paper focused on Access workloads (and vCMTS in particular) for an MSO, AWS Wavelength or AWS Outposts would provide approximately the same baseline performance in terms of latency and jitter.

The final option is to use non-AWS hardware running EKS, a Kubernetes offering from AWS, enabling a seamless workload orchestration across public, hybrid and private cloud. This provides an option to use

[6] https://aws.amazon.com/about-aws/global-infrastructure/regions_az/
[7] https://builders.intel.com/docs/networkbuilders/strategies-for-implementing-edge-services-in-the-10g-cable-network.pdf

familiar well proved and known quantity hardware where performance density and determinism are guaranteed while still availing of the flexibility and scalability offer by the cloud.

In short, AWS Regions, AZs, LZs, Wavelengths, and Outposts, as well as EKS Anywhere provide a gamut of options for hosting cloud instances and realizing a cloud deployment.  The general interfaces and procedures to access and manage these resources is the same whether the associated server is local or (varying degrees of) remote.

In the context of the vCMTS dataplane and similar Access/Edge workloads, the main technical factors (i.e., ignoring cost) for choosing one over another are:

1. Meets or exceeds minimum required latency
2. Meets or exceeds maximum possible jitter
3. Maintains deterministic behavior for throughput/performance
4. Supports DOCSIS Time Protocol or underlying IEEE 1588 Precision Time Protocol (PTP)[8]
5. If going with onsite solution like Outpost, have the space, power, and skillset to host the infrastructure
6. Allows for required level of high availability/redundancy

Latencies and jitter are going to depend on distances, mediums, and network complexity (ex. switch hops, congestion, etc.) between the cloud site and end users.  Deterministic behavior of the workload (i.e. beyond jitter on the network) will rely on having consistent access to compute, memory, and network resources with no major contention with other software applications.  AWS provides several facilities for managing workload placement, which can constrain the variables for achieving solid and consistent performance. Two of these schemes are employed in this study.

The first scheme is Amazon ECS placement groups and specifically Cluster Placement Groups, which are logical groups of tasks or services[9] that can run on any type of AWS cloud location.  In this way, a cluster can be defined to only run instances for the vCMTS dataplane and packet generators to achieve high and consistent throughput between deployed machine instances. Instances will be created within collocated Servers and Racks – maximizing communications efficiency. In this model is not possible to target specific servers within specific racks to host machine instances, AWS will manage the actual placement and hence actual location may vary as Instances are stopped and started.

The second scheme is the use of an Amazon EC2 Dedicated Host, which is a physical server with EC2 instance capacity fully dedicated for (one's) use. Dedicated Hosts support different configurations (physical cores, sockets and VCPUs) which allow (one) to select and run instances of different families and sizes depending on business need[10,11].  Though specifically designed for handling special Software licensing, in the context of this vCMTS dataplane analysis, it was used to run a number of c5n instances on the same physical server as shown in Figure 9.

[8] Technically DTP is part of vCMTS control plane and not strictly considered in this paper
[9] https://docs.aws.amazon.com/AmazonECS/latest/developerguide/clusters.html
[10] https://aws.amazon.com/ec2/dedicated-hosts/faqs/
[11] Note that there may be limited quantities and quotas for dedicated hosts in various regions.

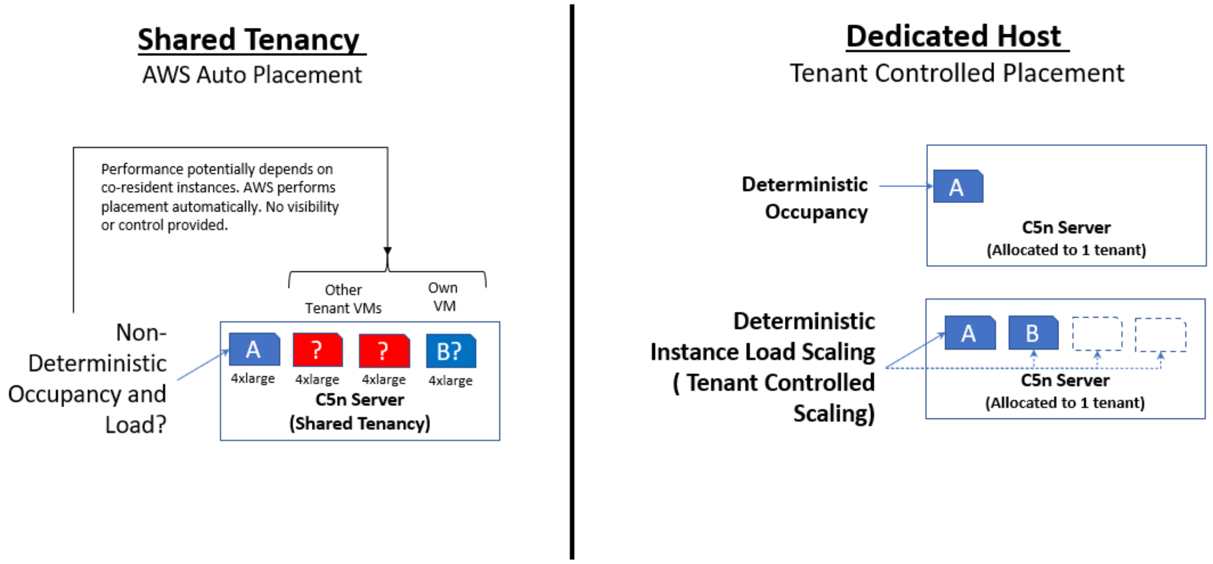

Shared Tenancy
AWS Auto Placement
Performance potentially depends on co-resident instances. AWS performs placement automatically. No visibility or control provided.
Other Tenant VMs
Own VM
Non-Deterministic Occupancy and Load?
A
?
?
B?
4xlarge
4xlarge
4xlarge
4xlarge
C5n Server
(Shared Tenancy)
Dedicated Host
Tenant Controlled Placement
Deterministic Occupancy
A
C5n Server
(Allocated to 1 tenant)
Deterministic Instance Load Scaling ( Tenant Controlled Scaling)
A
B
C5n Server
(Allocated to 1 tenant)

**Figure 9 – AWS auto placement versus Dedicated Host**

All that said, the cloud-based test setups described in the next section generally make use of AWS Regions as well as Dedicated Hosts to gather data and figure out the limits at each end of the spectrum of physical options. The other locations described in this section can be considered for future work but generally employ the exact same server hardware. As such, learnings in terms or throughput, scaling, and server capacity explored in this paper will translate to these locations too.

## 1.2. Application Porting and Test Setup

Intel had previously developed and released a reference vCMTS dataplane implementation to demonstrate the capability of various server hardware for the Cable Access network[12]. The reference package includes the vCMTS dataplane itself, packet generation and sink software, a cloud-native software infrastructure based on Kubernetes, and scripts for on-premise deployments. While there were some modifications that needed to be made for the cloud case (discussed later), all benchmarking was done using this vCMTS dataplane and existing packet generation software.

The on-premise tests were done on a server with the same CPU family that underlies the c5n instances – 1st Gen Intel® Xeon® Scalable Processors using the Skylake architecture. While this is an older platform than what is used for current vCMTS deployments[13], this apples-to-apples comparison makes it easier to identify underlying causes of unexpected behavior of the cloud infrastructure.

Figure 10 shows how the vCMTS dataplane and test software is deployed via Kubernetes across two different servers, one for the packet generation and sink facilities and one as the vCMTS dataplane device under test (DUT). These systems are connected back-to-back via Intel® 810 Series NICs and each SG is essentially assigned a dedicated 10GBE port (or equivalent VF). The traffic packet generation and

[12] https://01.org/access-network-dataplanes/overview
[13] https://networkbuilders.intel.com/solutionslibrary/maximizing-vcmts-data-plane-performance-with-3rd-gen-intel-xeon-scalable-processor-architecture

packet sink capabilities for a given vCMTS pod are provided by the same Traffic-Sim (DPDK pktgen) instantiation per SG.

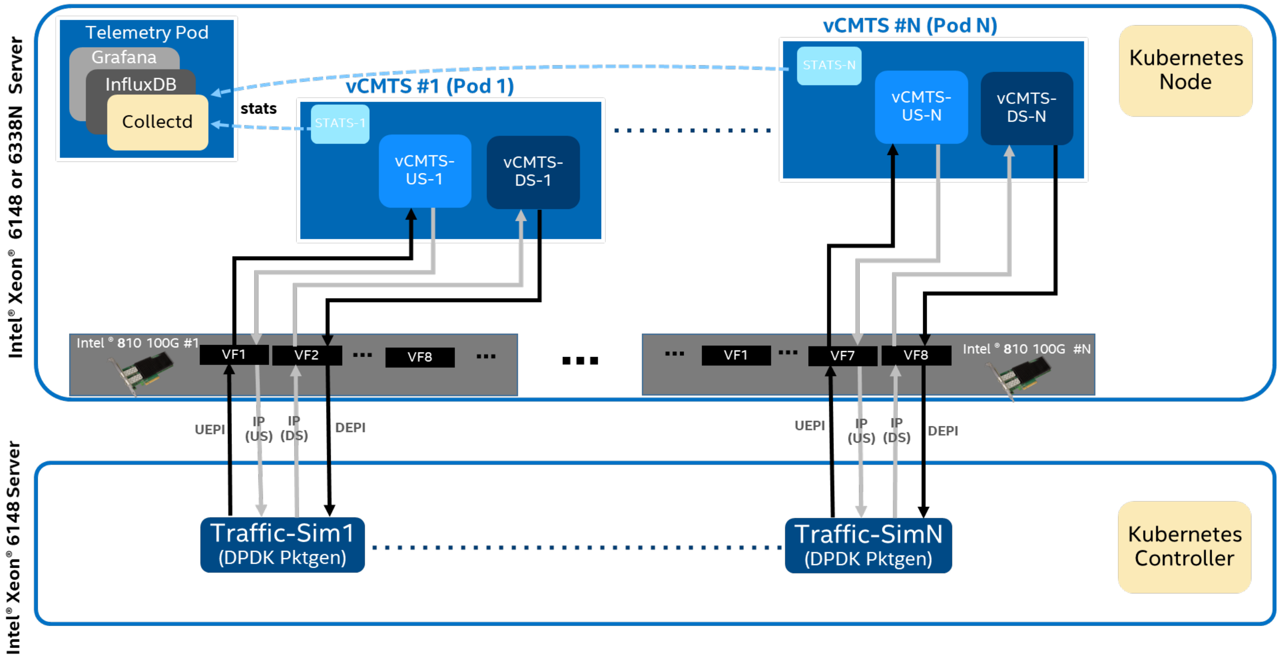


**Figure 10 – On-premise vCMTS performance analysis setup**

Figure 11 shows the test setup used for the AWS c5n performance analysis. At the high level, it is generally the same as the on-premise test setup insofar that the same vCMTS application and DPDK pktgen software is being used, however, there are a few key differences.

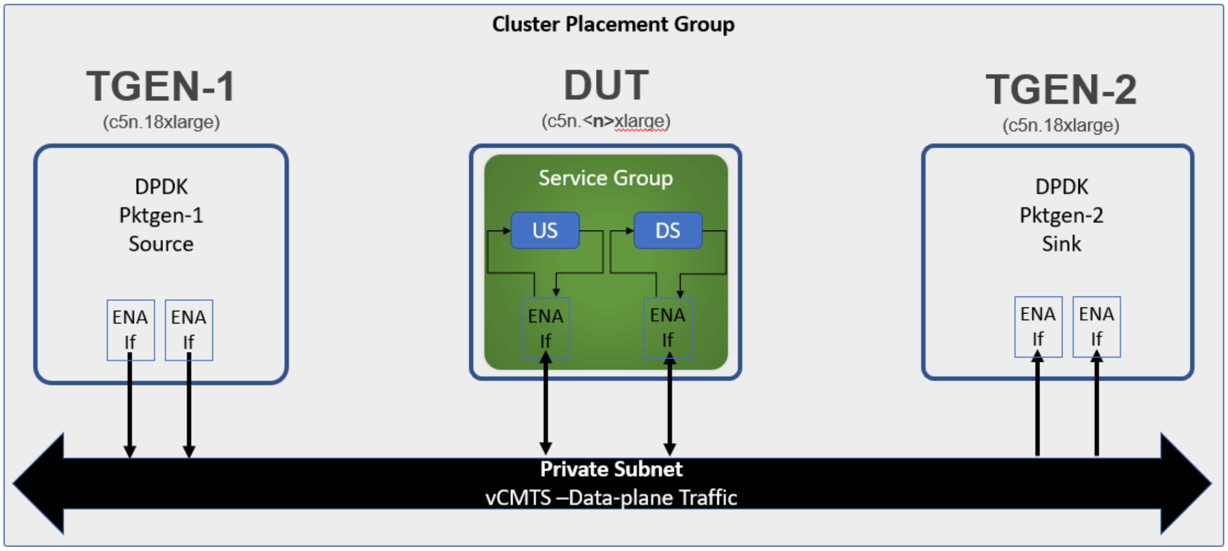


**Figure 11 – vCMTS performance analysis set-up on AWS**

From a system point of view, the packet generator and packet sink are running in different instances due to traffic throttling in the Nitro hardware observed during initial testing. That is, even though a single c5n.18xlarge instance has the raw compute capability to generate and receive the required number of packets, testing found there was a bottleneck through the networking sub-system that limited the bi-directional throughput. There will be more discussion of this finding later in the paper as it relates to scaling of the vCMTS dataplane itself.

On the infrastructure front, Kubernetes and the on-premise install scripts are not needed as the AWS Nitro system has its own hypervisor and application management facilities. In fact, the AWS cloud management tools were used to instantiate the desired instances and connect them via a Virtual Private Cloud (VPC) per the network diagram in Figure 12. Correspondingly, the existing reference testbed – which assumed that physical systems are connected back-to-back or through a simple switch – was modified so that L2TP tunnel and Cable-modem IP addresses were adapted to the sub-net properties of this VPC. At the network device level, the DPDK PMD for the Intel® NICs were switched to use the ENA PMD. This change also required the use of the DPDK out-of-tree IGB_UIO kernel module instead of the standard VFIO kernel module due to performance degradation with the ENA PMD.

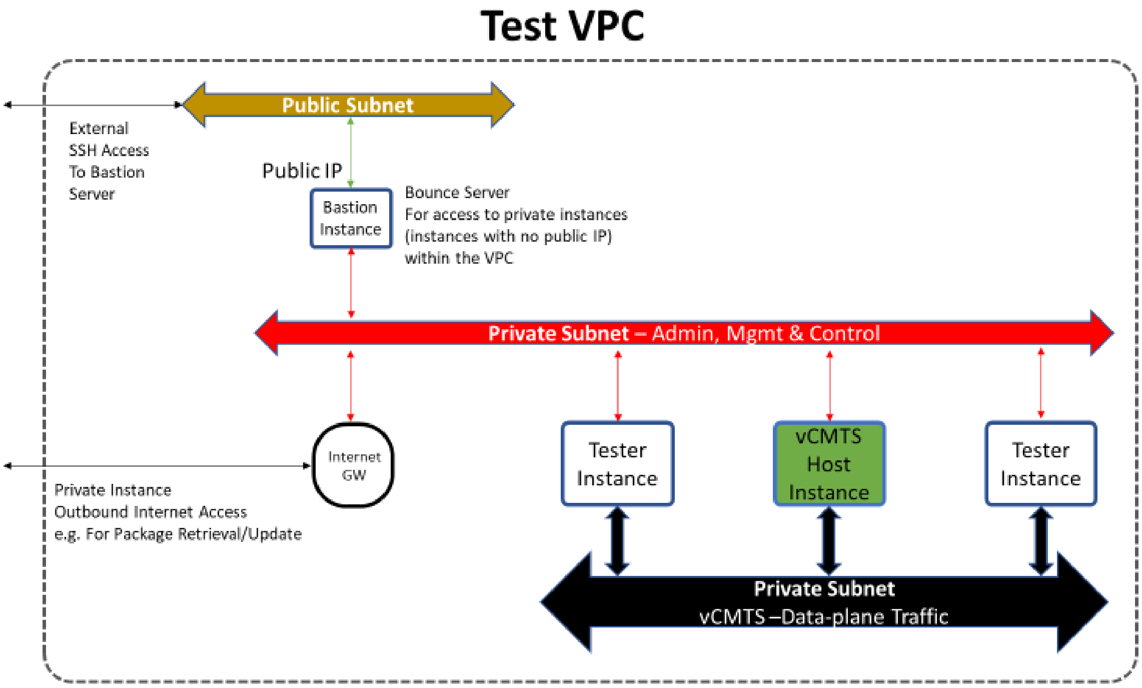


**Figure 12 – VPC network architecture for vCMTS dataplane testing**

All benchmarking, both on-premise and cloud setups, used the same general process as defined by RFC-2544[14], but limited to steady state load testing across a wide variety of packet sizes. In addition to the packet sizes recommended by RFC-2544, an iMix representative of MSO network traffic was also employed to replicate real-world circumstances. All tests were bi-directional in nature (i.e., sending traffic in both upstream and downstream directions simultaneously) and maximum throughputs were measured under zero packet loss conditions. Upstream traffic was configured to be a minimum of 10%

[14] https://datatracker.ietf.org/doc/html/rfc2544

of Downstream traffic for cloud performance tests. All tests employ the same number of (virtual) cable modems and use the same filter rules. In some situations, supplemental data, like per-pipeline-stage CPU cycle-counts, were captured later to the original test to help explain certain observations. Such data will be discussed in the relevant sections below.

## 2. Single-SG Throughput Analysis

Initial testing established single-SG performance across both the on-premise systems and the following AWS c5n instance types: c5n.metal, c5n.4xlarge, and c5n.2xlarge. The SG was chosen as the base unit of interest because a vCMTS dataplane would be scaled across vCPU cores based on the number of service groups required for a given network footprint.

Since all c5n instances make use of the same c5n server hardware it was expected that the single-SG performance would be similar across instances. It was also expected that this cloud performance would be similar to what is achievable on the on-premise platform. In both cases, there were surprises.

### 2.1. Comparing Instance Behavior to On-Premise

As mentioned in Section 1.2, bi-directional upstream and downstream throughput was measured for each test configuration with zero packet loss with traffic of several packet sizes and iMix types. Figure 13 plots the results for a c5n.metal instance.

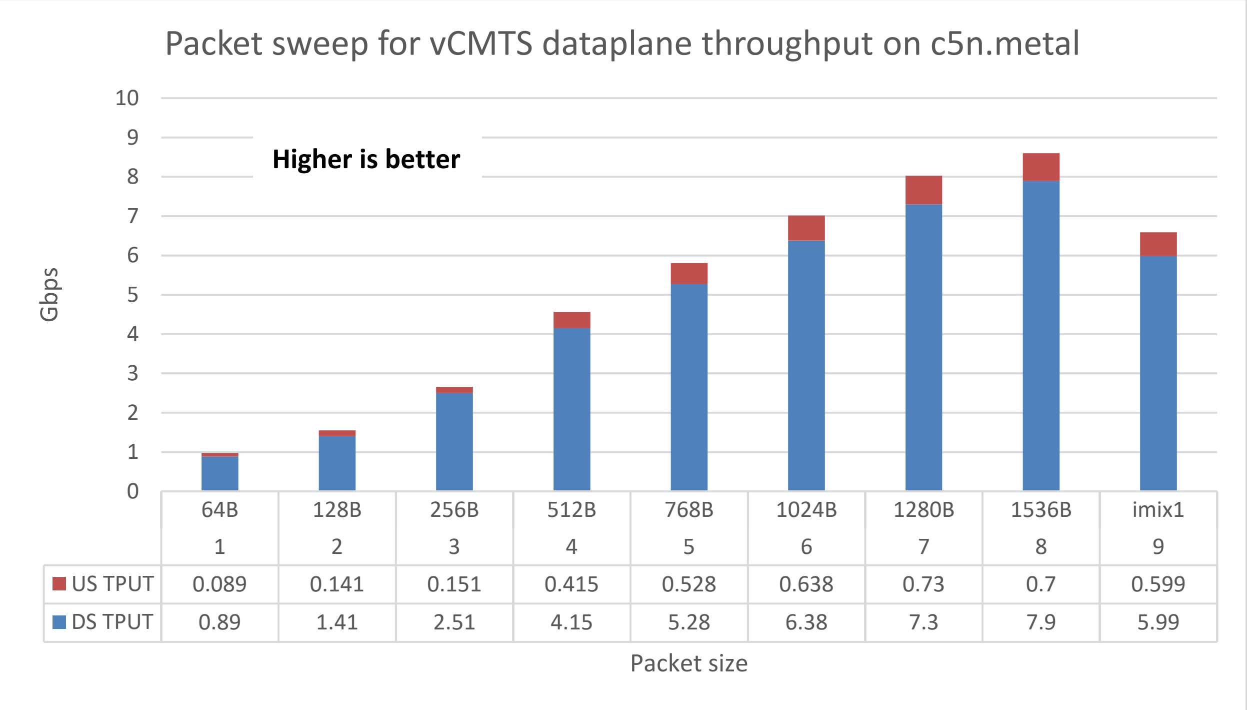
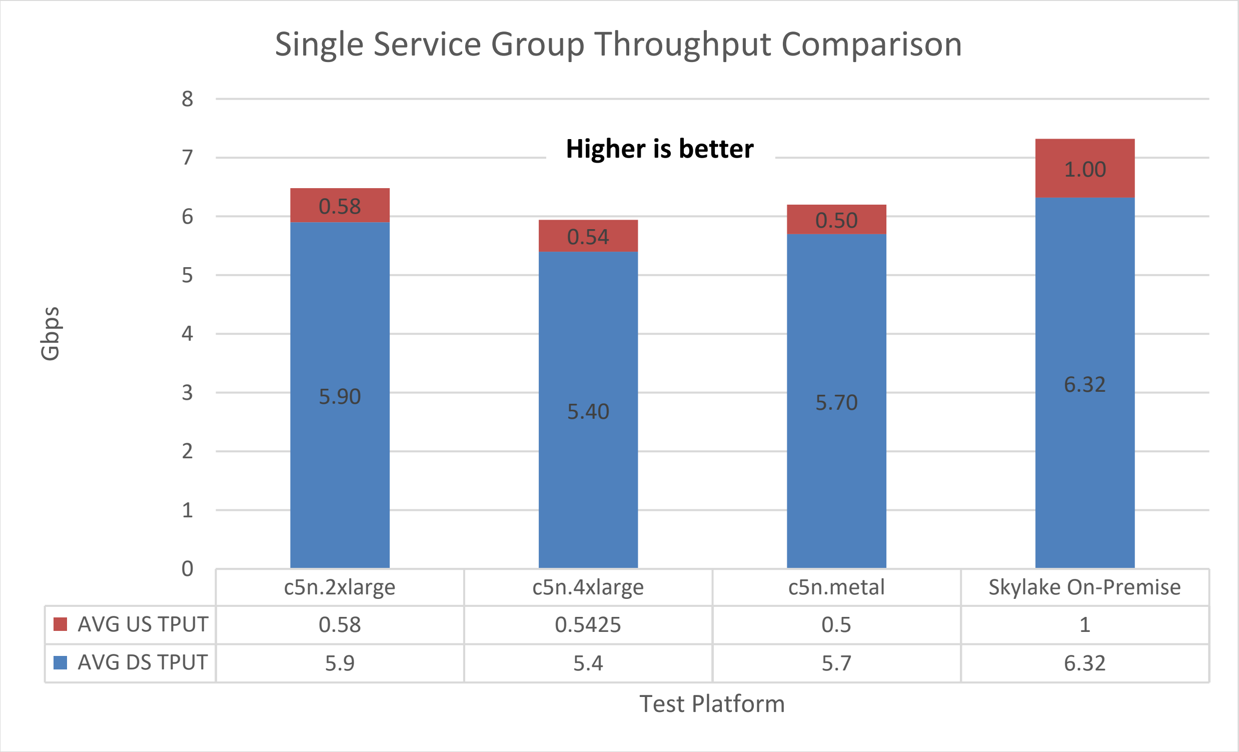


**Figure 13 – Max bi-directional throughput for single vCMTS SG (c5n.metal)**

The performance variance across packet sizes is expected and is a common phenomenon among any network function because smaller packet sizes at a certain network speed give less time to process that packet than larger ones. This behavior is seen in the other instances and both the on-premise platforms.

Since looking at any given packet size will yield the same themes, from here the analysis will use iMix data only. Figure 14 shows the iMix upstream and downstream aggregate throughput results across all tested instance types and on-premise platforms.



| | c5n.2xlarge | c5n.4xlarge | c5n.metal | Skylake On-Premise |
|---|---|---|---|---|
| 🟥 AVG US TPUT | 0.58 | 0.5425 | 0.5 | 1 |
| 🟦 AVG DS TPUT | 5.9 | 5.4 | 5.7 | 6.32 |

**Figure 14 – Comparing per-SG vCMTS Dataplane Performance for iMix**

The first observation is that the on-premise solution yielded the best performance at 7.32 Gbps of total upstream and downstream throughput. This was 13% better than the best cloud instance (c5n.2xlarge) at 6.48 Gbps. The result is surprising given that the cloud instances are running on CPUs with a higher base clock frequency than the CPU in the on-premise system, and the fact that the cloud instances may be able to take advantage of CPU Turbo operation for an even bigger boost (the on-premise servers disable Turbo in the name of determinism). Without mitigating factors in the memory or network, the performance of the vCMTS dataplane tracks linearly with the CPU speed. Thus, the cloud instances running at a minimum of 3.0GHz should get 25% better throughput than the 2.4GHz on-premise servers, but instead they are far in deficit.

How to explain this difference?

To answer this question, CPU cycle counts for each stage of the DOCSIS MAC pipeline were collected for both on-premise and cloud instance test systems. This was done in runs separate from the throughput measurements so one test did not affect the other. Figure 15 compares the results and sheds light on where each system is spending its CPU budget.

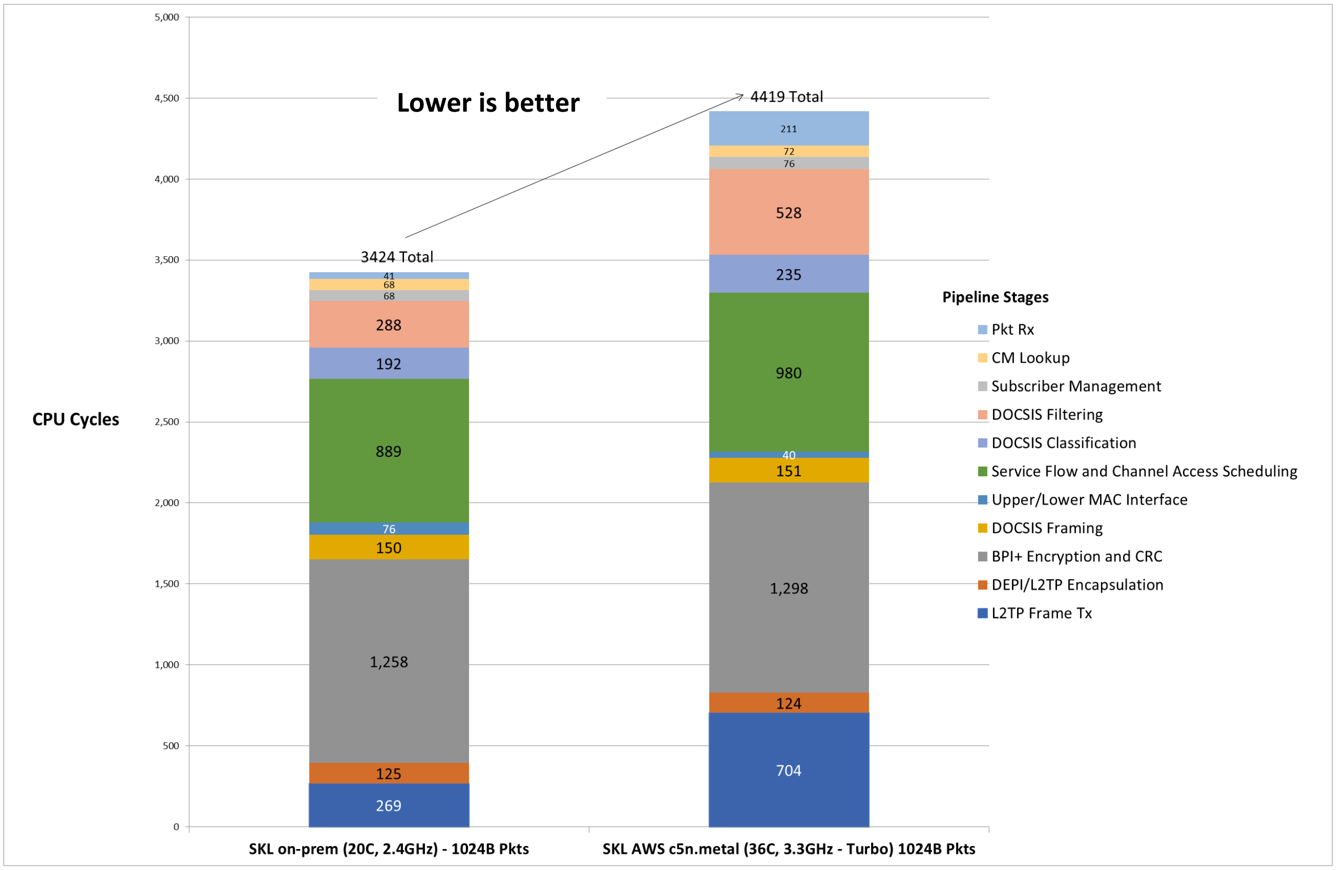


**Figure 15 – DOCSIS MAC downstream CPU cycle count comparison – On-premise vs Cloud**

Many of the pipeline stages have similar cycle counts. This means that both the cloud instances and the on-premise servers spent equal amounts of compute capability in those stages. However, the c5n instances took over 400%, 160%, and 150% longer in the packet receive, the packet transmit, and the DOCSIS filtering stages, respectively. Together this accounts for 29% more cycles and thus would process 29% less packets without even accounting for other possible bottlenecks.

The software part of the AWS Nitro system is generally lightweight; however, this data indicates that the ENA networking driver has quite a bit more overhead for basic receive and transmit functionality than a typical DPDK PMD in Linux. It is not clear what portion of this overhead can be won back with optimization to the ENA software driver or Nitro hypervisor (by AWS) and what portion is due to immutable behavior in the ENA/Nitro hardware itself.

The additional cycles for the DOCSIS filtering stage are not fully explained insofar that all tests use the same number of cable modems and the same filtering rules (i.e., parameters that are known to change the processing needs of this algorithm), but the current theory is that the significantly greater number of CPU cycles spent on packet receive is having some secondary effect on a shared resource in the CPU, for example, the L3 cache, instruction pipelines, or memory controller, and causing additional latency through the system.

The other main observation from Figure 14 is that the throughput was not consistent across the cloud instance types even though they are all running on the same underlying hardware and the throughput for a single service group should not be saturating the available network bandwidth. The c5n.2xlarge and c5n.metal data is approximately the same, within 5% variation, but the c5n.4xlarge performance was more than 10% worse.

## 2.2. Use of AWS Dedicated Hosts

Given that all c5n instances run on the same physical hardware and do not hit hard vCPU, memory, or network constraints to support a single SG, what could be the reason for the c5n.4xlarge performance difference? Looking back at Figure 5, four c5n.4xlarge instances can be instantiated on a given c5n server at any one time, and there are no guarantees where a given instance will be scheduled within an AWS Region (even if employing clusters). In other words, there could be other workloads from other AWS customers trying to make use of the same CPU L3 caches, memory, network interfaces, and other shared resources critical to vCMTS dataplane performance running on the same c5n server. These types of applications are known as noisy neighbors.

Per Section 1.1, AWS does provide a way to limit noisy neighbor risk via an AWS EC2 Dedicated Host. With a Dedicated Host, a customer reserves a whole server type (in this case a c5n) for themselves and therefore has full control on what instance types will be deployed there and what workloads will be running within those instances. Figure 16 shows how the initial single-SG test setup was modified from the original of Figure 11 to incorporate a Dedicated Host running the vCMTS dataplane in a c5n.4xlarge instance.

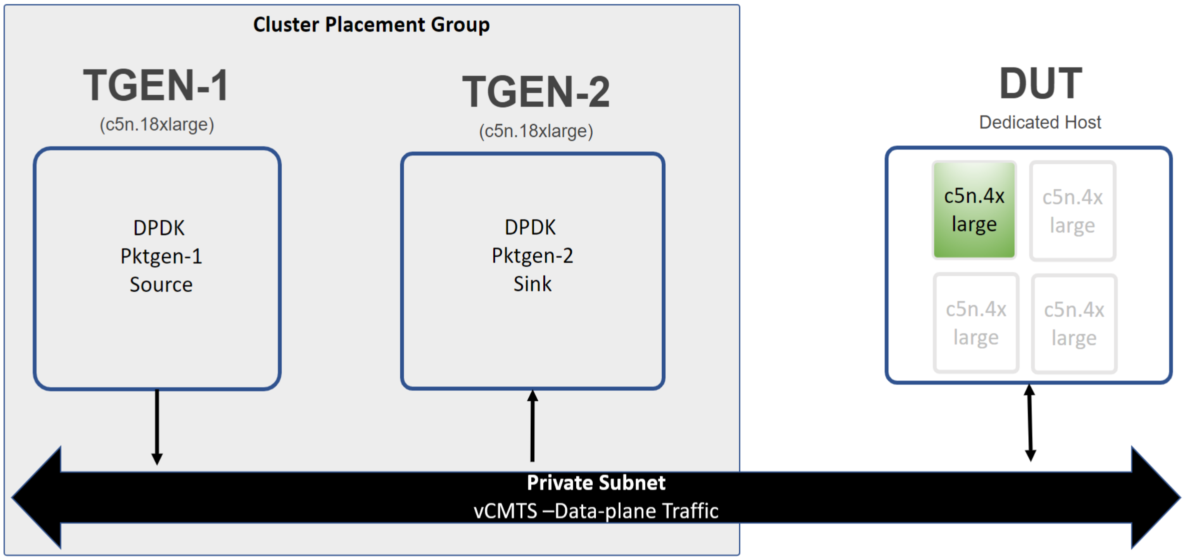


**Figure 16 – vCMTS test setup with Dedicated c5n Host**

Figure 17 shows the new c5n.4xlarge single-SG throughput test results alongside all the previous data.

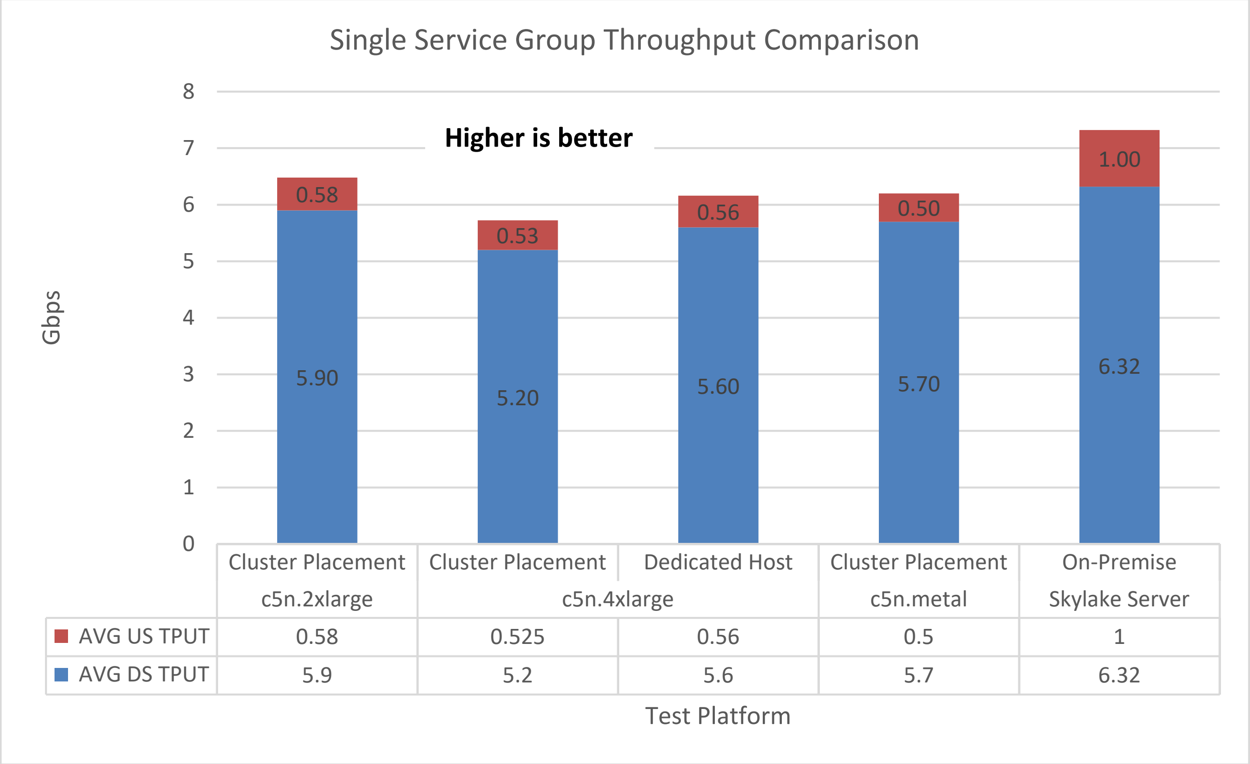


**Figure 17 – Affect of using a dedicated host on per-SG Performance**

It is clear that using the Dedicated Host helped bring the c5n.4xlarge results in-line with the performance of the other c5n instance types, for example, within 1% of the c5n.metal bi-directional throughput. By definition a c5n.metal instance takes up a whole c5n server, so it makes sense that the behavior of a single known application in that environment would be consistent. However, in the c5n.4xlarge case, there are no guarantees on where and how AWS will schedule such instances in real time from any customer.

It is interesting to note, anecdotally, that during the initial single-SG cluster-based testing the c5n.4xlarge throughput would be in line with the performance of the other instances as if it was running on a Dedicated Host. However, it was uncommon and there was no discernable pattern to when the throughput would be higher and when it would be lower. A network operator cannot rely on a random best-case scenario when dimensioning cloud resource needs for critical Access infrastructure.

That said, why did the testing not show this same behavior in the c5n.2xlarge instance case? It, too, can share a c5n server with other instances; per Figure 5 a c5n server can host up to 8 c5n.2xlarge instances and chances are that at some point there will be a noisy neighbor application running in one of them. Without knowing the full details of the AWS instance scheduler, the main lesson is that without full control of the hardware performance cannot be guaranteed. With enough tests it is expected that the c5n.2xlarge instances would also see the same performance variability as measured in the c5n.4xlarge case.

Why wouldn't a user always employ a Dedicated Host then? While the benefits are clear, there are some trade-offs to using this type of cloud infrastructure that would need to be factored into a total cost of ownership calculation. First, AWS limits the number of dedicated hosts per account to two per AWS

Region. Perhaps exceeding that stated limit is due to the fact they their original intention is to support legacy software licensing models, or perhaps it puts the user into the realm of deploying their own AWS Outpost servers? Regardless, it is not an approach that scales on its own. In addition, for every dedicated host available a user will pay a higher per time unit fee versus equivalent EC2 product offerings (ex. c5n.metal) at the time of this writing[15].

While it was not originally designed for networking applications, the AWS Dedicated Host concept is very useful for workloads that prize deterministic behavior and/or have hard requirements on system resources to achieve key performance indicators (KPIs). However, their utility for large scale deployments may be limited due to higher cost and capacity constraints.

# 3. Per-Instance Throughput Analysis

The per-SG analysis uncovered functional differences between c5n instance types and comparable on-premise solutions for the vCMTS dataplane and subsequently, insights into the c5n hardware itself, how those servers are shared among multiple customers, and key insights for managing performance expectations.

The next phase of testing is designed to determine the following for each c5n instance type:

1. Scaling capability (i.e., throughput per SG)
2. Maximum throughput possible via multiple SGs
3. The limiting factor for performance (ex. vCPU, memory, networking)

This information will be used to deepen the technical understanding of each instance type and, ultimately, to plan effective vCMTS dataplane application scaling strategies.

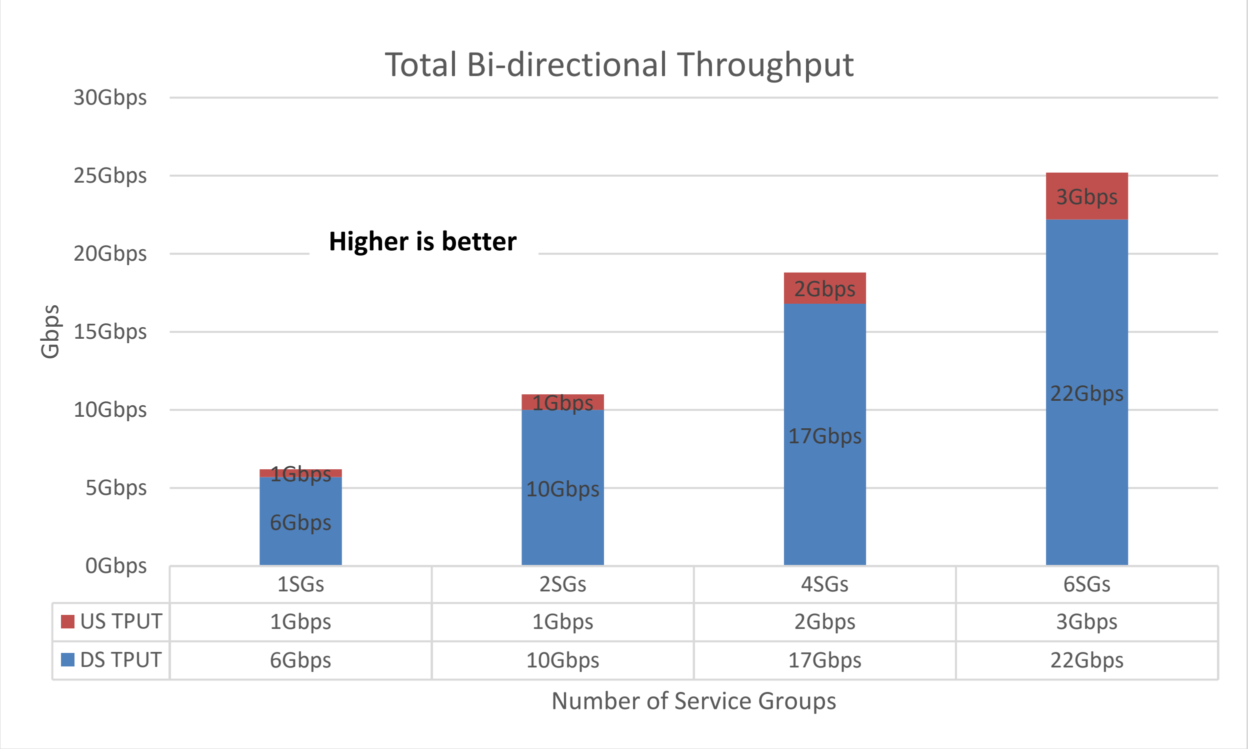
## 3.1. Finding Performance Limitations

This scaling analysis uses the same testing set up that was used for the single-SG benchmarking. Data will be collected from c5n.metal, c5n.4xlarge, and c5n.2xlarge instance types, as well as the on-premise platform. In the ideal case, good scaling means that the throughput per SG stays as constant as possible as more SGs are added into the system. However, the reality is that the scaling is typically non-linear, in the negative direction, for any system with shared resources such as all of these under test.

The vCMTS dataplane reference application requires two network interfaces per SG (one for upstream and one for downstream), so the maximum number of SGs for a given instance type is the total number of network interfaces minus 1 for an administration port, then that number divided by two. For example, the c5n.4xlarge instance is allowed up to 8 total interfaces, so with 1 interface used up for administration, the maximum possible number of SGs for that setup is 3.

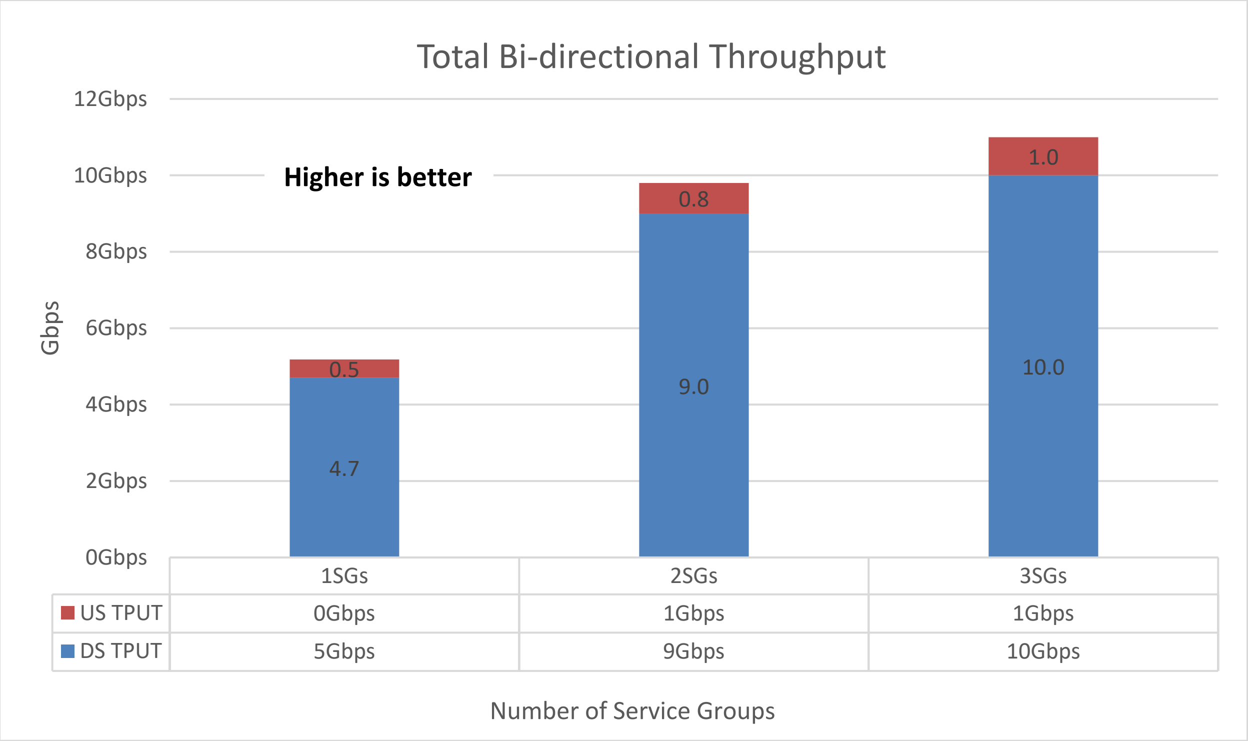
Figure 18 and Figure 19 below show maximum service-group scaling for c5n.metal and c5n.4xlarge instance types, respectively. The data for the c5n.4xlarge instance type was collected using the standard cluster setup versus using a Dedicated Host. Scaling data is not shown for the c5n.2xlarge instance since its 4 network interfaces limit it to supporting 1 SG at a time.

[15] https://aws.amazon.com/ec2/pricing/



| | 1SGs | 2SGs | 4SGs | 6SGs |
|---|---|---|---|---|
| 🟥 US TPUT | 1Gbps | 1Gbps | 2Gbps | 3Gbps |
| 🟦 DS TPUT | 6Gbps | 10Gbps | 17Gbps | 22Gbps |

**Figure 18 - Max bi-directional throughput for vCMTS service-group scaling (c5n.metal)**



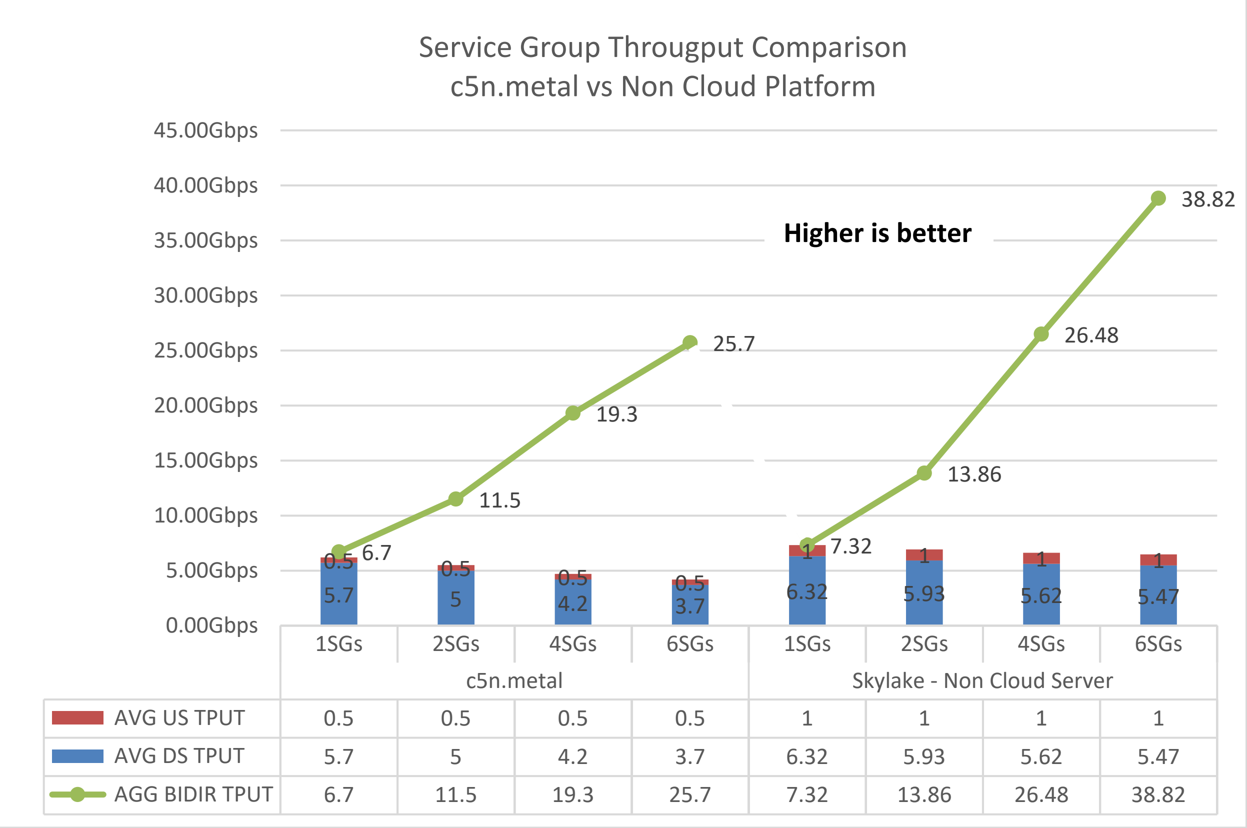
| | 1SGs | 2SGs | 3SGs |
|---|---|---|---|
| 🟥 US TPUT | 0Gbps | 1Gbps | 1Gbps |
| 🟦 DS TPUT | 5Gbps | 9Gbps | 10Gbps |

**Figure 19 - Max bi-directional throughput for vCMTS service-group scaling (c5n.4xlarge)**

The total bi-directional throughput of the c5n.metal instance shown in Figure 18 is not linear, but it tracks similarly to an on-premise solution insofar that it gets progressively worse as more SGs are added. The data points start at a single SG achieving 6.2 Gbps, then going to 2 SGs at 5.5 Gbps each and 4 SGs at 4.7 Gbps each, finally landing at 4.2 Gbps per SG in the 6 SG case. With 3 vCPUs supporting each SG, only 18 vCPUs were used out of the 72 available to achieve the total 25.2 Gbps bi-directional throughput for this instance type. Thus, the maximum throughput per instance is limited by the number of networking ports. Each additional two ports would allow the addition of another SG.

The story is different in the c5n.4xlarge case. There is decent scaling up to 2 SGs, with the throughput per SG at 5.2 Gbps and 4.9 Gbps in the 1 SG and 2 SG cases, respectively. However, the total throughput for the instance seems to be capped at about 11 Gbps, devastating the throughput per SG in the 3 SG case (bringing it down to 3.67 Gbps per SG). Prior testing showed that each network interface can only pass a maximum of ~4.0Mpps bidirectional or ~9.0Mpps unidirectional traffic, so total throughput for the system will be limited for small packet sizes. The iMix traffic in this testing averages ~1000 bytes in the downstream direction and ~300 bytes in the upstream direction and thus well outside of realm of possibility for saturating the theoretical 25 Gbps network bandwidth for the c5n.4xlarge instance type.

Bringing the on-premise platform into the picture, Figure 20 compares c5n.metal instance scaling to on-premise scaling and highlights the reasons for a huge difference in overall performance.



| | c5n.metal | | | | Skylake - Non Cloud Server | | | |
|---|---|---|---|---|---|---|---|---|
| | 1SGs | 2SGs | 4SGs | 6SGs | 1SGs | 2SGs | 4SGs | 6SGs |
| 🟥 AVG US TPUT | 0.5 | 0.5 | 0.5 | 0.5 | 1 | 1 | 1 | 1 |
| 🟦 AVG DS TPUT | 5.7 | 5 | 4.2 | 3.7 | 6.32 | 5.93 | 5.62 | 5.47 |
| ● AGG BIDIR TPUT | 6.7 | 11.5 | 19.3 | 25.7 | 7.32 | 13.86 | 26.48 | 38.82 |

**Figure 20 – Comparing on-premise to cloud scaling**

There are three reasons why the on-premise platform scales better and achieves much higher overall performance than the c5n.metal instance. First is that the number of vCPU cycles needed per packet is much higher for c5n instances due to large receive and transmit overhead in the ENA driver as described in Section 2.1. Second, the slope tracking the aggregate bi-directional throughput for the on-premise case in Figure 20 is much steeper versus the one tracking the c5n.metal performance, meaning that its average throughput per SG goes down at a slower rate (as seen in the per-SG bars). Finally, per prior analysis[16], the on-premise solution is only limited by the number of vCPUs and the amount of L3 cache; in the case of the Intel® Xeon® 6148 Scalable Processor with 20 physical cores (40 vCPUs) and 27.5 MB of L3 cache, the total achievable throughput per server with two sockets is ~100Gbps versus the ~25 Gbps for the c5n.metal case.

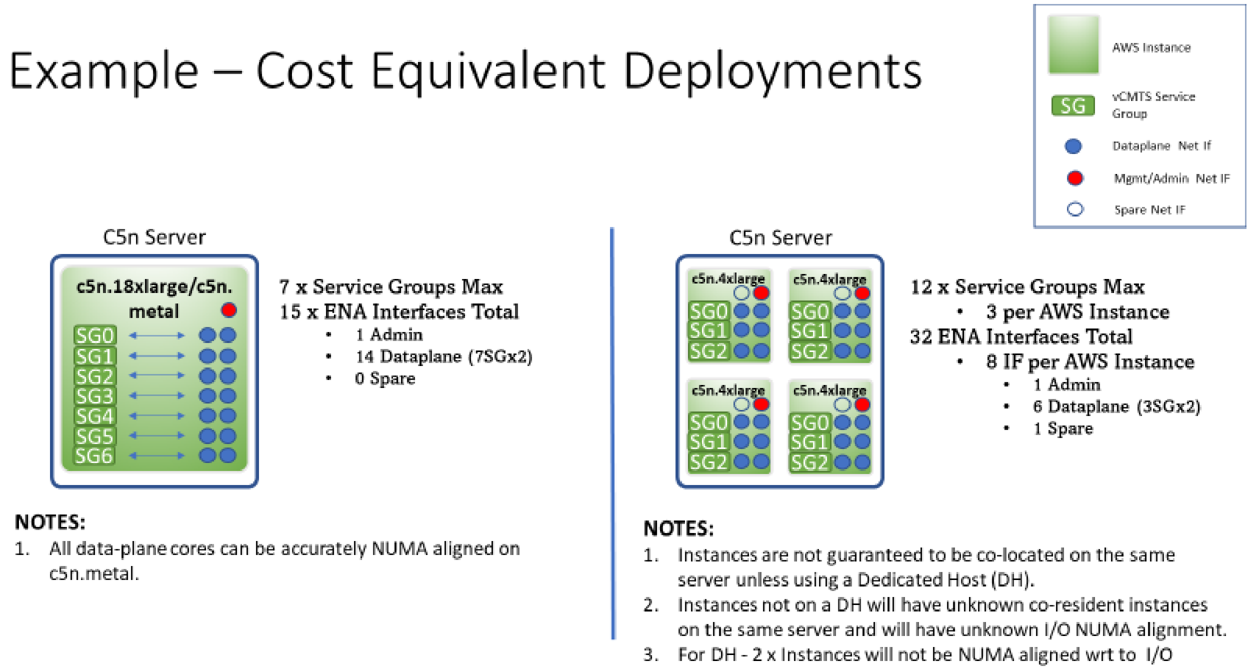
The main take-away is that the maximum vCMTS dataplane throughput for every c5n instance type is being limited by the networking sub-system. Specifically, the c5n.metal instance is limited by the number of networking ports and the c5n.4xlarge instance is limited by maximum packets per second in the ENA. Either way, this means that c5n instances, as defined today, have a mismatched compute-to-networking ratio and thus cloud resources that are being paid for (i.e., vCPUs) are not being used when the system is targeted at network-heavy applications.

### 3.2. Breaking Through Per-Instance Bottlenecks

Looking for clues in Figure 7, the compute to networking mismatch can be minimized with careful planning across multiple instances. In other words, can the c5n.metal bottleneck due to lack of network ports be solved by deploying multiple other c5n instance types with a better vCPU to port ratio (even if those instances have different per-instance limitations of their own)?

Figure 21 illustrates the example by comparing a single c5n.metal instance to four c5n.4xlarge instances, both of which fully occupy a single c5n server. These two system configurations are cost-equivalent in the financial sense, but also in terms of having the same amount of aggregate network bandwidth (100 Gbps) and a similar number of vCPUs (72 and 64). However, due to the greater total number of network ports available, the four c5n.4xlarge set up can support up to 12 SGs while the c5n.metal instance is limited to 7 SGs.

[16] https://networkbuilders.intel.com/solutionslibrary/maximizing-vcmts-data-plane-performance-with-3rd-gen-intel-xeon-scalable-processor-architecture



Example – Cost Equivalent Deployments

🟩 AWS Instance
SG vCMTS Service Group
● Dataplane Net If
● Mgmt/Admin Net IF
○ Spare Net IF

7 x Service Groups Max
15 x ENA Interfaces Total
- 1 Admin
- 14 Dataplane (7SGx2)
- 0 Spare

NOTES:
1. All data-plane cores can be accurately NUMA aligned on c5n.metal.

12 x Service Groups Max
- 3 per AWS Instance

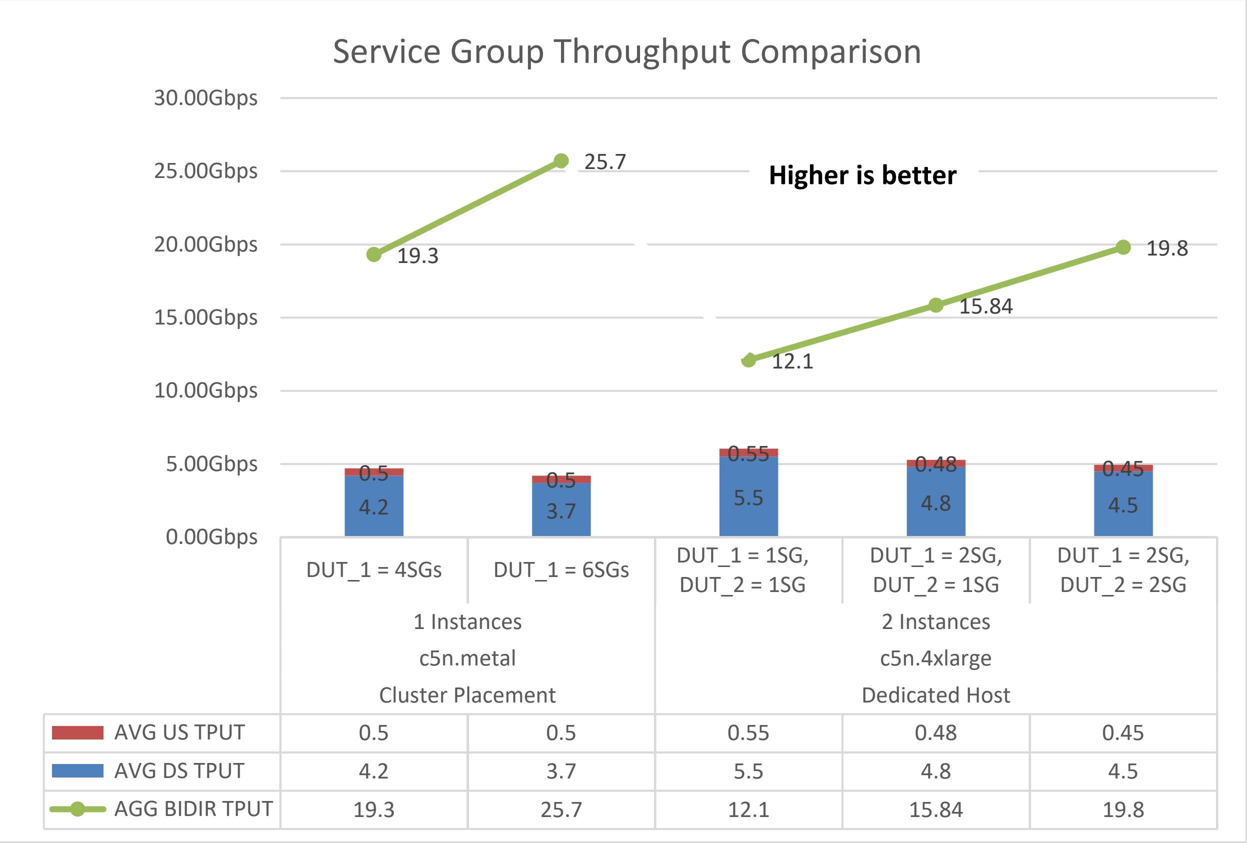
32 ENA Interfaces Total
- 8 IF per AWS Instance
  - 1 Admin
  - 6 Dataplane (3SGx2)
  - 1 Spare

NOTES:
1. Instances are not guaranteed to be co-located on the same server unless using a Dedicated Host (DH).
2. Instances not on a DH will have unknown co-resident instances on the same server and will have unknown I/O NUMA alignment.
3. For DH - 2 x Instances will not be NUMA aligned wrt to I/O

**Figure 21 – Example of cost equivalent c5n instance configurations**

Per the scaling data in the previous section, each c5n.4xlarge instance has a maximum possible throughput of 11 Gbps and therefore the maximum theoretical throughput for this system is 44 Gbps vs the 25.7 Gbps measured for the c5n.metal instance, a possible 71% improvement.

Taking the effect of real-life scaling factors into account, Figure 22 shows the total throughput measurements of two c5n.4xlarge instances. And while there is some degradation from the theoretical maximum, the data still demonstrates how compelling this approach can be. Note that due to organizational logistics (i.e., not any issue with the AWS infrastructure), benchmarking was completed with only two of the full four instance configuration.



| | DUT_1 = 4SGs | DUT_1 = 6SGs | DUT_1 = 1SG, DUT_2 = 1SG | DUT_1 = 2SG, DUT_2 = 1SG | DUT_1 = 2SG, DUT_2 = 2SG |
|---|---|---|---|---|---|
| 🟥 AVG US TPUT | 0.5 | 0.5 | 0.55 | 0.48 | 0.45 |
| 🟦 AVG DS TPUT | 4.2 | 3.7 | 5.5 | 4.8 | 4.5 |
| ● AGG BIDIR TPUT | 19.3 | 25.7 | 12.1 | 15.84 | 19.8 |

**Figure 22 – Throughput of two c5n.4xlarge instances to one c5n.metal**

Starting with these measurements of 19.8 Gbps for two c5n.4xmetal instances supporting 2 SGs each, we see a 10% scaling penalty to the theoretical maximum of 22 Gbps. Conservatively assuming a further scaling penalty of 15% when going from two to four c5n.4xlarge instances, one could expect up to 33.6 Gbps of bi-directional traffic, which is over 76% of the theoretical maximum and still a 30% improvement over the c5n.metal capacity at the same cost.

Table 1 summarizes the vCMTS dataplane performance for each test setup alongside the resource usage for each scenario to give an idea of the constraint that is causing the bottleneck. It also calculates what percentage of the instance resources are being used and what is left unused – specifically for the vCMTS dataplane at least.

| Instance Type | vCPU | Ethernet | Ports | Maximum vCMTS tput | Tput limitation | Used Resources | Unused Resources | % Unused Resources |
|---|---|---|---|---|---|---|---|---|
| **Single c5n.metal** | 72 | Shared 100G | 15 | 25.7Gbps with 6 SGs | Number of ports | 18 vCPUs, 25.7Gbps, 12 ports | 54 vCPUs, 74.3 Gbps, 3 ports | 75% vCPUs, ~74% network |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Four c5n.4xlarge** | 64 | Shared 100G | 32 | 33.6 Gbps[17] with 12 SGs | pps per instance | 36 vCPUs, 33.6 Gbps, 25 ports | 28vCPUs, 66.4 Gbps, 7 ports | ~43% vCPUs, 66% network |
| **Two c5n.4xlarge** | 32 | Shared 50G | 16 | 19.8 Gbps with 6 SGs | pps per instance | 18 vCPUs, 19.8 Gbps, 13 ports | 14 vCPUs, 30.2 Gbps, 3 ports | ~43% vCPUs, ~60 Gbps network |

**Table 1 - Scaling via multiple smaller instances**

Clearly, aggregate performance of four c5n.4xlarge instances (based on the measurements from two c5n.4xlarge instances) is better than a single c5n.metal instance from both absolute throughput and percentage of resource usage perspectives. This configuration drives better utilization of vCPU cores by having a greater aggregate number of network interfaces to work with. That said, all c5n platforms used for vCMTS dataplane applications will not make use of all available compute and network capability.

So, what can be done with those unused resources? First, given that one type of bottleneck to scaling is the number of Ethernet ports per instance, a software vendor could modify the workload to use less ports overall. For example, the vCMTS dataplane application dedicates a port for US and a port for DS for every SG to optimize its transmit and receive logic, but this scheme could be modified to make better use of this constrained resource. Even if the new code to manage port usage adds more cycles per packet, the change would be well worth the effort in terms of maximizing overall throughput.

Second, unused vCPUs could be used for applications that do not require network access and do not otherwise interfere with the vCMTS dataplane resources, for example, to pre-process network or system telemetry. Alternatively, those unused vCPUs could be used in a high availability scheme, with backup vCMTS dataplane instantiations pointed to relevant databases and network ports but not actually processing packets until the primary applications fail.

No matter what is done with resources unused by the vCMTS dataplane, this testing shows that it is possible to use multiple smaller instances (in this example, c5n.4xlarge) to overcome sub-optimal compute-to-networking ratios and achieve better total bi-directional throughput for c5n-based infrastructure.

## 4. Conclusions

There is a lot of interest in moving applications found in the modern network into cloud infrastructure for all the typical reasons: offloading deployment and configuration of hardware, managing all infrastructure and over the top services though a consistent pane of glass, and having all needs for scale and multiple levels of redundancy taken care of by someone else. These are good and compelling reasons.

However, not all applications are created equal; each one must be evaluated to see if it makes financial, business, and technical sense to move from on-premise equipment to the cloud. This paper focused on only technical considerations related to making best use of cloud infrastructure for vCMTS dataplane

[17] Four c5n.4xlarge performance estimated based on the two c5n.4xlarge lab measurements

processing, with its challenging requirements for high throughput, cryptographic elements, low latency, and consistent performance, and leaves the rest to other venues. Only AWS product offerings were evaluated in this context, but the same methodology and criteria can be used to choose the best cloud infrastructure from any CSP.

The results were clear that even though the AWS c5n family of cloud instances was shown to be best among AWS offerings because of its high capability with Intel® Xeon® Scalable Processors paired with 100 Gbps network interfaces, there is a huge gap in their performance relative to a similar on-premise deployment. Per testing, there is too much overhead and a lack of performance guarantees that make it costly or operationally prohibitive to widely deploy the vCMTS dataplane to the cloud.

That said, AWS and other CSPs are continually innovating and introducing new cloud products; it is inevitable that they will address some of the observations described in this paper:

- Networking capability was not matched to the compute capability, limiting overall throughput
  - Networking ports per instance was generally too low
  - Compute cycles to support packet receive and transmit on the ENA was too high
  - Published bandwidth capacities had too many caveats to achieve in real-world tests (ex. packet per second limitations in the ENA capped iMix performance)
- Lack of resource isolation and/or control of workload placement leads to non-deterministic performance

A cloud instance (or rather, family of instance types) that addresses these issues would increase overall system performance, reduce custom software development efforts, assure customers with expected and consistent behavior, and generally give maximum value to the resources and services a network operator is purchasing to run an important part of their network.

In the meantime, even if not suitable for a large-scale vCMTS deployment, the convenience of cloud infrastructure still makes it interesting for other use cases, including the expansion of test capability, the enabling of remote collaboration, as a temporary solution for surge traffic, or to expand fail-over capacity.

Again, the most value from this work is in the education and process. The particulars (ex. test data for a c5n instance type) may have a limited lifespan, but the evaluation framework will help network operators make informed decisions on CSP offerings for any dataplane applications they are looking to move into the cloud now and into the future. On the flip side, it may help a CSP create the perfect cloud product optimized for the Access network.

### 4.1. Future Work

While cloud infrastructure in various forms has been around a long time and has proven its value in many business arenas, it is still early days for applying its underlying technologies and processes to the high demand needs of Access and Edge workloads like the vCMTS dataplane used in this study. There are many compelling topics in this domain demanding further research; some of these are: to expand this analysis to future AWS offerings (especially the next generation of the c5n family), to expand this analysis to similar products from other CSPs, to study questions around latency in the network by, for example, varying the location of the packet generator and sink relative to the dataplane function, and evaluate cloud offerings for vCMTS control plane needs.

# Abbreviations

| | |
|---|---|
| AWS | Amazon Web Services |
| AZ | AWS availability zone |
| bps | bits per second |
| CMTS | cable modem termination system |
| CoSP | communications service provider |
| COTS | commercial off the shelf |
| CSP | cloud service provider |
| DAA | distributed access architecture |
| DPDK | data plane development kit |
| DTP | DOCSIS time protocol |
| DUT | device under test |
| ECS | AWS elastic container service |
| ENA | AWS elastic network adaptor |
| KPI | key performance indicator |
| ML | machine learning |
| MSO | multi service operator |
| NIC | network interface card |
| PCIe | PCI Express |
| PMD | poll mode driver |
| pps | packets per second |
| PTP | IEEE 1588 precision time protocol |
| SCTE | Society of Cable Telecommunications Engineers |
| SG | service group |
| TCO | total cost of ownership |
| VF | virtual functions (in context of a PCIe device) |
| VPC | virtual private cloud |

# Bibliography & References

Amazon Web Services (2021). “Amazon EC2 Overview”, https://aws.amazon.com/ec2/ (Accessed: 25 August 2021).

Amazon Web Services (2021). “Amazon EC2 Instance Types”, https://aws.amazon.com/ec2/instance-types/ (Accessed: 25 August 2021).

Amazon Web Services (2021). “AWS Nitro System”, https://aws.amazon.com/ec2/nitro/ (Accessed: 25 August 2021).

Barr, J. (2018). “New C5n Instances with 100 Gbps Networking”, https://aws.amazon.com/blogs/aws/new-c5n-instances-with-100-gbps-networking/ (Accessed: 25 August 2021).

Intel Corporation (2021). “Intel vCMTS Reference Dataplane and NFV Stack”, https://01.org/access-network-dataplanes/overview (Accessed: 25 August 2021).

Ryan, B. et. al. (2021). “Maximizing vCMTS Data Plane Performance with 3rd Gen Intel® Xeon® Scalable Processor Architecture”, https://networkbuilders.intel.com/solutionslibrary/maximizing-vcmts-data-plane-performance-with-3rd-gen-intel-xeon-scalable-processor-architecture (Accessed: 25 August 2021).

Heaton, E. (2020). “Strategies for Implementing Edge Services in the 10G Cable Network”, https://builders.intel.com/docs/networkbuilders/strategies-for-implementing-edge-services-in-the-10g-cable-network.pdf (Accessed: 25 August 2021).

Bradner, S. and McQuaid, J. (1999). “Benchmarking Methodology for Network Interconnect Devices”, https://datatracker.ietf.org/doc/html/rfc2544 (Accessed: 25 August 2021)

# Appendix A – On-Premise Test Configuration

| Test Environment Configuration Information and Relevant Variables | |
|---|---|
| CM Lookup & Subscriber Mgmt | 300 subscribers per service group, 4 IP addresses per subscriber |
| DOCSIS Filtering | 6 filter groups, 2 filter groups associated with each cable-modem<br>16 filter rules per filter group<br>10% matched, 90% unmatched (default action - permit) |
| DOCSIS Classification | 16 rules per subscriber,<br>10% matched - enqueue to one of 3 service-flow queues<br>90% unmatched - enqueue to default service-flow queue |
| Downstream Service-Flow Scheduling | 8 service-flow queues per subscriber (4 active) |
| Downstream Channel Scheduling | 6 x OFDM (1.89 Gbps) Channels, 2 x channel-bonding groups.<br>Or 2 x OFDM (1.89 Gbps) and 32-SC-QAM (42.24 Mbps) Channels, 4 x channel-bonding groups<br>NOTE: channel-bonding groups are distributed evenly across cable-modems |
| Upstream Bandwidth Scheduling | Upstream Scheduler not used.<br>Upstream bandwidth pre-allocated in grants of 2KB per service ID.<br>Bandwidth grants balanced evenly across 300 cable-modems. |
| Ethernet CRC | Downstream: 100% CRC re-generation<br>Upstream: 0% CRC verification<br>NOTE: CRC relates to inner frames |
| Encryption | 100% AES, 0% DES |
| Packet IMIX Distribution | Upstream 65% : 84B, 18% : 256B, 17% : 1280B<br>Downstream 15% : 84B, 10% : 256B, 75% : 1280B |

| vCMTS DUT | |
|---|---|
| **Hardware** | |
| Platform | Advantech Skylake SKY-8201L1 |

| | |
|---|---|
| CPU | Intel® Xeon® Gold 6148, Dual Socket, 20C @ 2.4GHz<br>Microcode : 0x2006a08 |
| Memory | 12 x 16GB DDR4-2667 |
| Hard Drive | Intel® SSD (480G) |
| Network Interface Card | 4 x Intel® Ethernet Converged Network Adapter 810 100GbE |
| Crypto Acceleration Card | 4 x Intel® QuickAssist Technology Adapter 8970 |
| **Software** | |
| Host OS | Ubuntu 20.04, Linux Kernel v5.4.x |
| DPDK | DPDK v20.08 |
| vCMTS | Intel vCMTS Reference Data plane v20.10 |
| Date Tested | July 9th, 2021 |

| **vCMTS Traffic Generator** | |
|---|---|
| **Hardware** | |
| Platform | Intel® Wildcat Pass S2600WTTR |
| CPU | Intel® Xeon® E5-2699 v4, Dual Socket, 22C @ 2.2GHz<br>Microcode: 0xb000036 |
| Memory | 6 x 16GB DDR4-2400 |
| Hard Drive | Intel® SSD (480G) |
| Network Interface Card | 4 x Intel® Ethernet Converged Network Adapter 810 100GbE |
| **Software** | |
| Host OS | Ubuntu 20.04, Linux Kernel v5.4.x |
| DPDK | DPDK v20.08 |
| Traffic Generator | DPDK Pktgen v19.10 |

# Appendix B – AWS c5n Test Configuration

| Test Environment Configuration Information and Relevant Variables | |
|---|---|
| CM Lookup & Subscriber Mgmt | 300 subscribers per service group, 4 IP addresses per subscriber |
| DOCSIS Filtering | 6 filter groups, 2 filter groups associated with each cable-modem<br>16 filter rules per filter group<br>10% matched, 90% unmatched (default action – permit) |
| DOCSIS Classification | 16 rules per subscriber,<br>10% matched - enqueue to one of 3 service-flow queues<br>90% unmatched - enqueue to default service-flow queue |
| Downstream Service-Flow Scheduling | 8 service-flow queues per subscriber (4 active) |
| Downstream Channel Scheduling | 6 x OFDM (1.89 Gbps) Channels, 2 x channel-bonding groups.<br>Or 2 x OFDM (1.89 Gbps) and 32-SC-QAM (42.24 Mbps) Channels, 4 x channel-bonding groups<br>NOTE: channel-bonding groups are distributed evenly across cable-modems |
| Upstream Bandwidth Scheduling | Upstream Scheduler not used.<br>Upstream bandwidth pre-allocated in grants of 2KB per service ID.<br>Bandwidth grants balanced evenly across 300 cable-modems. |
| Ethernet CRC | Downstream: 100% CRC re-generation<br>Upstream: 0% CRC verification<br>NOTE: CRC relates to inner frames |
| Encryption | 100% AES, 0% DES |
| Packet IMIX Distribution | Upstream 65% : 84B, 18% : 256B, 17% : 1280B<br>Downstream 15% : 84B, 10% : 256B, 75% : 1280B |

| vCMTS DUT | |
|---|---|
| **Hardware** | |
| Platform | AWS c5n family |
| CPU | Intel® Xeon® Platinum 8124M, Dual Socket |

| | |
|---|---|
| Memory | Up to 196GB, depends on instance type |
| Hard Drive | N/A |
| Network Interface | Up to 100GBE, depends on instance type |
| **Software** | |
| Host OS | Ubuntu 20.04, Linux Kernel v5.4.x |
| DPDK | DPDK v20.08 |
| vCMTS | Intel vCMTS Reference Data plane v20.10 |
| Date Tested | July 30th, 2021 |

| | |
|---|---|
| **vCMTS Traffic Generator** | |
| **Hardware** | |
| Platform | AWS c5n family |
| CPU | Intel® Xeon® Platinum 8124M, Dual Socket |
| Memory | Up to 196GB, depends on instance type |
| Hard Drive | N/A |
| Network Interface | Up to 100GBE, depends on instance type |
| **Software** | |
| Host OS | Ubuntu 20.04, Linux Kernel v5.4.x |
| DPDK | DPDK v20.08 |
| Traffic Generator | DPDK Pktgen v19.10 |

## Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates.  See backup for configuration details.  No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.