## Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Description</th>
<th>Revision Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install-guide for vCMTS reference dataplane release v20.10.0</td>
<td>October 2020</td>
</tr>
</tbody>
</table>
Related Information

This install guide relates to v20.10.0 of the Intel vCMTS reference dataplane package which may be downloaded from Intel 01.org at the following link: https://01.org/access-network-dataplanes

Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATHMAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web site at www.intel.com.

© 2020 Intel Corporation. All rights reserved. Intel, the Intel logo, and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

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

1 Introduction ....................................................................................................................... 7
1.1 System overview ............................................................................................................. 7
  1.1.1 vCMTS Reference Dataplane Pipeline ................................................................. 10
  1.1.2 vCMTS Reference Dataplane NFV Stack .............................................................. 11
  1.1.3 CPU Core Management .......................................................................................... 12
  1.1.4 Power Management Features ............................................................................... 13
1.2 Network Interface Configuration .................................................................................... 16
1.3 Memory Module Configuration ...................................................................................... 17
1.4 System Configuration ................................................................................................... 18
1.5 Release Package Contents ............................................................................................ 19

2 Installation Guide ............................................................................................................. 23
2.1 vCMTS dataplane server preparation ............................................................................ 23
  2.1.1 Configure system BIOS settings .......................................................................... 23
  2.1.2 Check NIC Firmware and Driver versions ............................................................. 24
  2.1.3 Check disk space settings ...................................................................................... 25
  2.1.4 Check OS and Linux kernel versions ..................................................................... 26
  2.1.5 Disable automatic Linux package updates ............................................................. 27
  2.1.6 Change file size limit settings .............................................................................. 27
  2.1.7 Configure Linux GRUB settings .......................................................................... 28
  2.1.8 Install public key for remote access to vCMTS traffic-generator server ............... 29
  2.1.9 Load the vCMTS reference dataplane package .................................................... 29
  2.1.10 Configure proxy servers .................................................................................... 30
  2.1.11 Install Linux packages ...................................................................................... 31
2.2 vCMTS traffic-generator server preparation .................................................................. 32
  2.2.1 Configure system BIOS settings .......................................................................... 32
  2.2.2 Check NIC Firmware and Driver versions ............................................................. 33
  2.2.3 Check disk space settings ...................................................................................... 34
  2.2.4 Check OS and Linux kernel versions ..................................................................... 35
  2.2.5 Disable automatic Linux package updates ............................................................. 36
  2.2.6 Change file size limit settings .............................................................................. 36
  2.2.7 Configure Linux GRUB settings .......................................................................... 37
  2.2.8 Install public key for remote access to vCMTS dataplane server ......................... 38
  2.2.9 Load the vCMTS reference dataplane package .................................................... 38
  2.2.10 Configure proxy servers .................................................................................... 39
  2.2.11 Install Linux packages ...................................................................................... 40
  2.2.12 Install the vCMTS platform management tool ................................................... 40
2.3 Configure vCMTS platform settings ............................................................................. 42
  2.3.1 vCMTS dataplane platform configuration ............................................................ 42
  2.3.2 vCMTS traffic-generator platform configuration ................................................ 43
Introduction

2.4 Install Kubernetes ..........................................................44
  2.4.1 Install and configure docker on the vCMTS dataplane server ........44
  2.4.2 Install and Configure Docker on the vCMTS traffic-generator server ....45
  2.4.3 Install Kubernetes controller node on the vCMTS traffic-generator server ....47
  2.4.4 Install Kubernetes node on the vCMTS dataplane server ........47
  2.4.5 Verify Kubernetes installation........................................48
2.5 Connect NIC Interfaces.....................................................51
2.6 Install vCMTS dataplane node software ..................................52
  2.6.1 Generate openssl certificates ......................................52
  2.6.2 Install QAT drivers..................................................52
  2.6.3 Install Intel® IPSEC MB Library.................................53
  2.6.4 Install DPDK.......................................................53
  2.6.5 Install Linux* bare-metal specific software .......................53
  2.6.6 Install Kubernetes specific software .............................54
2.7 Install vCMTS traffic-generator node software ........................57
  2.7.1 Install DPDK.......................................................57
  2.7.2 Install Linux* baremetal specific software .......................57
  2.7.3 Install Kubernetes specific software .............................57
2.8 Configure vCMTS dataplane..............................................59
  2.8.1 Configure vCMTS dataplane service-group options...............62
2.9 Run vCMTS dataplane and traffic-generator software ................64
  2.9.1 Start infrastructure and application instances ...................64
  2.9.2 How to stop infrastructure and application instances .............65
2.10 System verification.......................................................66
  2.10.1 Check Kubernetes dashboard ....................................66
  2.10.2 Attach to application instances ................................67
  2.10.3 Query service-group options ................................68
  2.10.4 Start dataplane traffic .........................................68
  2.10.5 Check the Grafana dashboard ..................................69
  2.10.6 vCMTS CLI Tool................................................75
2.11 System reboot ................................................................76
3 Appendix............................................................................77
  3.1 DOCSIS DDP Forwarding Sample Application ..........................77
    3.1.1 Description..........................................................77
    3.1.2 User Guide..........................................................78
  3.2 How to Verify/Create Grafana Data-source for vCMTS Telemetry ....83
List of Figures

Figure 1 Intel vCMTS Reference Dataplane – Kubernetes Orchestrated Platform ................. 8
Figure 2 Intel vCMTS Reference Dataplane – Bare-metal Platform...................................... 9
Figure 3 Intel vCMTS Reference Dataplane – Packet-processing Pipeline ........................... 10
Figure 4 Intel vCMTS Reference Dataplane – NFV Stack.................................................... 11
Figure 5 Intel vCMTS Dataplane Reference Platform – Layout for 24 Core Dual Processor 12
Figure 6 Intel vCMTS Dataplane Reference Platform – Power Management Architecture 13
Figure 7 Intel Speed Select Technology - Base Frequency Feature........................................ 14
Figure 8 Kubernetes WebUI Login Screen............................................................................. 66
Figure 9 Intel vCMTS Dataplane Reference Platform – Kubernetes WebUI Dashboard.......... 67
Figure 10 Grafana Dashboard Login Screen........................................................................... 71
Figure 11 Intel vCMTS Dataplane Reference Platform – System Summary Dashboard......... 72
Figure 12 CPU Utilization Graph on Grafana Platform Metrics Dashboard.......................... 73
Figure 13 CPU Utilization Query on Grafana Platform Metrics Dashboard.......................... 74

List of Tables

Table 1 System BIOS settings – vCMTS Dataplane server ................................................... 23
Table 2 Linux GRUB settings – vCMTS Dataplane server ..................................................... 28
Table 3 System BIOS settings – vCMTS Traffic-generator server......................................... 32
Table 4 Linux GRUB settings – vCMTS Traffic-generator server.......................................... 37
Table 5 vCMTS dataplane configurable service-group options ............................................. 59
Table 6 vCMTS dataplane fixed service-group options......................................................... 61
Table 7 Service-group bandwidth guide................................................................................ 62
Introduction

This document describes, step by step, how to install and run the Intel vCMTS reference dataplane system with either a Kubernetes-orchestrated Linux Container or bare-metal Linux* environment. This includes a DPDK Pktgen based cable traffic generation system for upstream and downstream traffic simulation.

This release of the Intel vCMTS reference dataplane also includes a sample application, which demonstrates how the Dynamic Device Profile (DDP) feature of the Intel 800 series NIC can be used to enable Cable-specific NIC pre-processing to steer DOCSIS MAC traffic using the Flow-director NIC feature for upstream traffic and the Receiver Side Scaling (RSS) NIC feature for downstream traffic.

Instructions to install and run this application (docsis-ddp-fwd) are provided in the Appendix section 3.1 DOCSIS DDP Forwarding Sample Application

1.1 System overview

The Intel® vCMTS reference dataplane environment consists of a vCMTS dataplane node and a traffic generation node. The reference platform, as presented in this guide, for both of these nodes is a 2U server with dual Intel® Xeon® Gold 6252N processors and two 100G Intel® Ethernet Network Adapter dual port 810 network interface cards (NICs). The vCMTS dataplane node may also optionally include Intel® QuickAssist Technology (QAT) 8970 100Gbps or 8960 50Gbps PCIe cards.

Note that the Intel® 10G quad port X710-DA4 or 25G Intel® dual port the Intel® 100G dual port NICs may optionally be used instead of 100G NICs. And if PCI slots are available on the platform, multiple NICs per CPU may be used.

Note also that the system described in this document should serve as a sample system configuration. Servers based on other Intel® Xeon® scalable processors or Xeon® D processors with different core-counts are also supported. More NICs may be added to the system for a greater amount of I/O as required, and the system can be configured with or without Intel® QuickAssist cards.

The entire system can be deployed under a Kubernetes* orchestrated environment or optionally under a bare-metal Linux* environment.

In the Kubernetes* deployment, multiple Docker* containers host DPDK-based DOCSIS MAC upstream and downstream dataplane processing for individual cable service-groups (SG’s) on the vCMTS dataplane node. On the vCMTS traffic-generation node Docker* containers host DPDK Pktgen-based traffic generation instances which simulate DOCSIS traffic into corresponding vCMTS dataplane instances. Under a bare-metal Linux* deployment application instances are run as processes on the Linux* host OS.

Each vCMTS dataplane instance, described as a POD in a Kubernetes* deployment, represents a service-group and has separate containers/applications for upstream and downstream DOCSIS MAC dataplane processing. DOCSIS control-plane is simulated through a JSON configuration file containing subscriber cable-modem information. Upstream scheduling is simulated by synthetically generated cable-modem DOCSIS stream segments.
Telemetry functions run in Docker* containers as a Daemonset (a singleton POD) under Kubernetes* or as systemd services in a bare-metal deployment. A comprehensive set of vCMTS dataplane statistics and platform KPI's are gathered by the open-source collectd* daemon and stored in an InfluxDB* time-series database. A Grafana* dashboard is provided for visualization of these metrics based on InfluxDB* queries.

The sample platform configuration shown in Figure 1 below supports vCMTS dataplane processing for a configurable number of service-groups on a Kubernetes orchestrated platform. Various mappings of cores and VF's to service-group Pod's are supported as described in section 2.8 “Configure vCMTS dataplane”.

![Figure 1 Intel vCMTS Reference Dataplane – Kubernetes Orchestrated Platform](image)
Kubernetes* and Docker* may be disabled so that the vCMTS dataplane and telemetry processes run on bare-metal as shown in the Figure 2 below. In this case, process management is performed by the vcmts-\textit{pm} tool as shown below. Such an environment may be preferable for dataplane performance analysis and benchmarking.
1.1.1 vCMTS Reference Dataplane Pipeline

The core component of the Intel® vCMTS reference dataplane release package is a reference implementation of a DOCSIS MAC dataplane, also known as the vCMTS dataplane.

Intel has developed a DOCSIS MAC dataplane compliant with DOCSIS 3.1 specifications (notably MULPI, DEPI, UEPI and SEC specifications) and based on the DPDK packet-processing framework. The key purpose of this development is to provide a tool for characterization of vCMTS dataplane packet-processing performance and power-consumption on Intel® Xeon® platforms.

The vCMTS upstream and downstream packet processing pipelines implemented by Intel® are shown in Figure 3 below. Both upstream and downstream dataplanes are implemented as a two-stage pipeline of upper-mac and lower-mac processing. The DPDK API used for each significant DOCSIS MAC dataplane function is also shown below.

Figure 3  Intel vCMTS Reference Dataplane – Packet-processing Pipeline

Each service-group as shown in a Kubernetes orchestrated platform in Figure 1 or a Bare-metal platform in Figure 2 contains an instantiation of the above upstream and downstream pipelines. Each service-group handles all subscriber traffic for a specific service-group which covers a group of cable subscribers in a particular geographical area.
1.1.2 vCMTS Reference Dataplane NFV Stack

The Intel® vCMTS reference dataplane runs within an industry standard Linux Container based NFV stack as shown in Figure 4 below. Optionally, it can also be run using a bare-metal Linux* stack which removes the Docker*, Kubernetes* and ONOS* components.

In a Kubernetes* deployment vCMTS upstream and downstream dataplane processing for individual cable service-groups run in Docker* containers on Ubuntu OS, allowing them to be instantiated and scaled independently. Likewise, in a bare-metal deployment individual cable service-groups are instantiated as standalone processes.

When deploying in a Kubernetes* environment the entire system including applications, telemetry, power-management and infrastructure management is orchestrated by Kubernetes*, with Intel-developed plugins being used for resource management functions such as CPU core management and assignment of SR-IOV interfaces for NICs and QAT devices. The Kubernetes* deployment also utilizes the included python tool to abstract and simplify the configuration of components before handing over the management to Kubernetes*.

As mentioned in a previous section, a bare-metal environment is also supported. For this deployment option the entire system including applications, telemetry and infrastructure management is managed by a python tool, `vcmts-pm` which has been provided to abstract and simplify the configuration and running of software components on the vCMTS reference dataplane system. The `vcmts-pm` tool will be described in more detail in later sections.
1.1.3 CPU Core Management

CPU core management differs between the Kubernetes* and bare-metal deployments. In the bare-metal Linux* deployment the vcmts-pm tool is exclusively responsible for the core allocation for both upstream and downstream dataplane applications. The vcmts-pm tool supports a number of different core allocation options depending on the specific requirements of the service-groups being configured on the system.

The vcmts-pm tool is not used for vCMTS dataplane core allocation for a Kubernetes* deployment. CPU core allocation is managed by CMK (CPU Manager for Kubernetes) in this case. CMK is an open-source component developed by Intel. The CMK manages two types of core-pool, a shared core-pool for containers that share physical cores (such as for upstream dataplane processing) and an exclusive core-pool for containers which require exclusive use of a physical core (such as for downstream dataplane processing).

A number of different CPU core layouts are supported for the Intel Reference vCMTS platform, which are described in Table 5 vCMTS dataplane configurable service-group options. Figure 5 below shows an example of a CPU core layout on a 24-core dual-processor CPU. In this example CPU cores are allocated separately for upstream and downstream dataplane instances for each service-group. A downstream dataplane processing instance is split across two sibling hyperthreads of a core while two upstream dataplane processing instances run on sibling hyperthreads of a core.

Note that under Kubernetes, CMK (CPU Manager for Kubernetes) allocates CPU cores for downstream processing from the exclusive core-pool, and cores for upstream processing from the shared core-pool.

Figure 5 Intel vCMTS Dataplane Reference Platform – Layout for 24 Core Dual Processor

NOTE: Upstream Scheduler and Control-Plane processing is not currently supported for the Intel Reference vCMTS platform. CPU cores are reserved for such components not running on the Intel Reference vCMTS platform.
1.1.4 Power Management Features

The Intel® vCMTS reference dataplane offers the capability to make use of the latest Intel® Power Management technology. These features enable both power saving and improved performance through the use of Intel® Speed Select Technology and DPDK Power Management features. Three power management features are supported: Time-of-day power management, Branch-monitoring and Intel® Speed Select Technology Base Frequency. These power management features may be enabled using the vcmts-pmt tool menu and are described in more detail in the sections that follow.

1.1.4.1 Time of Day Power Management

By using this power management method power-profiles may be configured for automated power management of a Kubernetes-orchestrated vCMTS system. A power profile may be configured for each vCMTS service-group POD which is sent to the Power-Manager at startup, as shown in the architecture diagram below.

Figure 6 Intel vCMTS Dataplane Reference Platform — Power Management Architecture
These power-profiles determine how dataplane CPU cores are managed by the DPDK Power-Manager. For quiet hours during the day when network traffic is low, individual dataplane CPU cores associated with a service-group may be set to a power-state with a lower frequency as determined from its power-profile. The quiet and busy hours can be configured using the provided ‘vcmts-pm’ tool.

1.1.4.2 Branch Monitor Power Management
This power management method uses branch ratio counters to determine core busy-ness. When this method is selected, a list of vCMTS dataplane cores will be monitored.

The algorithm pays close attention to the ratio between branch hits and branch misses. A tightly polling PMD thread will have a very low branch ratio, indicates that a low rate of packets is being processed and the core frequency is thus scaled down to the minimum allowed value.

When the packet receive rate increases, the code path will alter, causing the branch ratio to increase. When the ratio goes above the ratio threshold, the core frequency will be scaled up to the maximum allowed value.

Through this method idle and busy periods can be identified and power management applied to enable savings without negatively impacting dataplane performance.

1.1.4.3 Speed Select Technology Base Frequency (SST-BF)
Intel® Speed Select Technology is an umbrella term for a collection of features that provide more granular control over CPU performance. The Intel® Speed Select Technology Base Frequency (SST-BF) feature allows specific cores to run at a higher base frequency (i.e. P1) by reducing the base frequencies of other cores, as shown below.

![Figure 7 Intel Speed Select Technology - Base Frequency Feature](image)

The number of low/high priority cores and core-frequencies depends on the CPU SKU. e.g. for 32C Ice Lake N SKU there may be 18 high-priority cores which could be allocated for downstream traffic processing.
The vCMTS reference system supports the prioritization of certain service-groups to use a higher base frequency than others, thus improving their performance.

This feature is only available on compatible platforms. For more details see here: https://www.intel.com/content/www/us/en/architecture-and-technology/speed-select-technology-article.html.
1.2 Network Interface Configuration

The network interface ports of the vCMTS dataplane and traffic-generation nodes shown in Figure 1 should be interconnected by optical fiber cables via an Ethernet switch.

Traffic should be routed between correlating Pktgen and vCMTS dataplane instances by MAC learning in the switch as MAC addresses of vCMTS dataplane and Pktgen ports are based on service-group ID's. Switch configuration is not covered in this document.

If an Ethernet Switch is not available, the NIC ports of the vCMTS dataplane and traffic-generation nodes may be connected directly by optical fiber cables. However, in this case care must be taken to connect the physical NIC ports of correlating Pktgen and vCMTS dataplane application instances. A helper script is provided in the release package to assist with this and instructions are included in the later sections of this install guide as environment settings must be configured before doing this.

When vCMTS dataplane traffic is started as described in section 2.10.4 an ARP request is sent from each traffic-generator instance to establish a link with its correlating vCMTS dataplane instance.

⚠️ Please note that NICs must be installed in appropriate CPU-affinitized PCI slots for balanced I/O (which for the sample configuration in Figure 1 means two NICs per CPU socket).

The NIC layout for the system can be checked by running the command shown below.

Note that for the example below only the 25G XXV710 NICs are used for vCMTS dataplane traffic.

Device ports with their most significant address bit unset below (18:00.0, 18:00.1, 1a:00.0, 1a:00.1) are affinitized to CPU socket 0, while those with their most significant address bit set (86:00.0, 86:00.1, b7:00.0, b7:00.1) are affinitized to CPU socket 1. This system configuration is an example of balanced I/O.

```
$ lspci | grep Ethernet

18:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)  
18:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)  
1a:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)  
1a:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)  
3d:00.0 Ethernet controller: Intel Corporation Ethernet Connection X722 for 10GBASE-T (rev 09)  
3d:00.1 Ethernet controller: Intel Corporation Ethernet Connection X722 for 10GBASE-T (rev 09)  
81:00.0 Ethernet controller: Intel Corporation 82572EI Gigabit Ethernet Controller (Copper) (rev 06)  
86:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)  
86:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)  
b7:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)  
b7:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
```
1.3 Memory Module Configuration

It is very important to ensure that DRAM modules are installed correctly in the server so that all memory channels are utilized.

For example, for the Intel® Xeon® Gold 6252N scalable processor there are six memory channels per CPU socket. In this case a minimum of 12 DRAM modules are required to utilize all memory channels. Furthermore, modules should be installed in the correct color-coded slots for optimum memory channel utilization. In general it is recommended to use high-speed memory modules if possible.

DRAM module layout on the system can be checked by running the command below.

```bash
ishw -class memory
```

Correct memory-channel utilization can also be verified using the pcm-memory tool which is provided by Intel. Below is an example of a correctly configured system which is using all 12 memory channels.

```
/opt/intel/pcm/pcm-memory.x
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>--- Socket 0 --</td>
<td>-- Socket 1 --</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>--- Memory Channel Monitoring --</td>
<td>-- Memory Channel Monitoring --</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>--- Mem Ch 0: Reads (MB/s): 2088.30 --</td>
<td>-- Mem Ch 0: Reads (MB/s): 2355.60 --</td>
</tr>
<tr>
<td>--- Writes (MB/s): 1753.91 --</td>
<td>-- Writes (MB/s): 2017.74 --</td>
</tr>
<tr>
<td>--- Mem Ch 1: Reads (MB/s): 2108.08 --</td>
<td>-- Mem Ch 1: Reads (MB/s): 2399.79 --</td>
</tr>
<tr>
<td>--- Writes (MB/s): 1710.44 --</td>
<td>-- Writes (MB/s): 2055.59 --</td>
</tr>
<tr>
<td>--- Mem Ch 2: Reads (MB/s): 2123.32 --</td>
<td>-- Mem Ch 2: Reads (MB/s): 2366.44 --</td>
</tr>
<tr>
<td>--- Writes (MB/s): 1707.17 --</td>
<td>-- Writes (MB/s): 2031.86 --</td>
</tr>
<tr>
<td>--- Mem Ch 3: Reads (MB/s): 1687.27 --</td>
<td>-- Mem Ch 3: Reads (MB/s): 2220.50 --</td>
</tr>
<tr>
<td>--- Writes (MB/s): 1620.93 --</td>
<td>-- Writes (MB/s): 2010.21 --</td>
</tr>
<tr>
<td>--- Mem Ch 4: Reads (MB/s): 1755.19 --</td>
<td>-- Mem Ch 4: Reads (MB/s): 2274.56 --</td>
</tr>
<tr>
<td>--- Writes (MB/s): 1659.92 --</td>
<td>-- Writes (MB/s): 1971.64 --</td>
</tr>
<tr>
<td>--- Mem Ch 5: Reads (MB/s): 1690.55 --</td>
<td>-- Mem Ch 5: Reads (MB/s): 2316.71 --</td>
</tr>
<tr>
<td>--- NODE 0 Mem Read (MB/s): 11452.71 --</td>
<td>-- NODE 0 Mem Read (MB/s): 13933.60 --</td>
</tr>
<tr>
<td>--- NODE 0 Mem Write(MB/s): 10113.44 --</td>
<td>-- NODE 0 Mem Write(MB/s): 12088.41 --</td>
</tr>
<tr>
<td>--- NODE 0 P. Write (T/s): 18804 --</td>
<td>-- NODE 0 P. Write (T/s): 18866 --</td>
</tr>
<tr>
<td>--- NODE 0 Memory (MB/s): 21566.15 --</td>
<td>-- NODE 0 Memory (MB/s): 26022.00 --</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>--- System Read Throughput (MB/s): 25386.31 --</td>
<td></td>
</tr>
<tr>
<td>--- System Write Throughput (MB/s): 22201.84 --</td>
<td></td>
</tr>
<tr>
<td>--- System Memory Throughput (MB/s): 47588.15 --</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--</td>
</tr>
</tbody>
</table>
```
1.4 System Configuration

The following is a sample system configuration for Intel vCMTS reference dataplane v20.10.0.

<table>
<thead>
<tr>
<th>vCMTS Dataplane Node</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Hard Drive</td>
</tr>
<tr>
<td>Network Interface Card</td>
</tr>
<tr>
<td>Crypto Acceleration Card</td>
</tr>
<tr>
<td><strong>Software</strong></td>
</tr>
<tr>
<td>Host OS</td>
</tr>
<tr>
<td>DPDK</td>
</tr>
<tr>
<td>vCMTS</td>
</tr>
<tr>
<td>Linux Container</td>
</tr>
<tr>
<td>Container Orchestrator</td>
</tr>
<tr>
<td>Statistics</td>
</tr>
<tr>
<td>vCMTS Traffic Generator Node</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Hard Drive</td>
</tr>
<tr>
<td>Network Interface Card</td>
</tr>
<tr>
<td><strong>Software</strong></td>
</tr>
<tr>
<td>Host OS</td>
</tr>
<tr>
<td>DPDK</td>
</tr>
<tr>
<td>Linux Container</td>
</tr>
<tr>
<td>Traffic Generator</td>
</tr>
</tbody>
</table>
1.5 Release Package Contents

The directory tree of the Intel vCMTS Reference Dataplane Release package is shown below.
The following is a description of the main directories of the release package, and their contents.

"vcmts" top-level directory:
Contains the following release documentation:
- README: release package details
- RELEASE_NOTES: release notes relevant to the package
- LICENSE: license file relevant to the package

"kubernetes" subdirectory:
Contains files to build and install Kubernetes infrastructure required for the Intel(R) vCMTS reference dataplane system, in the following sub-directories:
- cmk: contains files required for CMK (CPU Manager for Kubernetes)
- dashboard: contains files required for Kubernetes WebUI dashboard
- docker: contains files to build docker images for Kubernetes infrastructure components
- helm: contains files required for Helm, which is used for configuration of vCMTS dataplane and Pktgen Kubernetes POD's
- install: contains files required for Kubernetes installation

"pktgen" subdirectory:
Contains files to build the traffic simulation components of the vCMTS reference dataplane platform, in the following sub-directories:
- docker: contains files to build the docker image for the vCMTS traffic generator (based on DPDK Pktgen application)
- config: contains PKT files with commands performed at Pktgen start-up such as setting of src/dest IP addresses and ARP handling

"src" subdirectory:
Contains top-level Makefile to build the vCMTS dataplane application and directory structure for C source-code modules, in the following sub-directories:
- vcmtsd: contains C source-code modules, template config files and files to build the docker image for the vCMTS dataplane application, and the following sub-directories:
  - config: contains vCMTS dataplane application configuration files
  - stats: contains C source-code modules for statistics collection
  - docker: contains files to build the docker image for the vCMTS dataplane application

  docsis-ddp-fwd: contains C source-code modules and Pktgen PKT file for the DDP sample application
patches: contains DPDK and Pktgen patch files in the following sub-directories:

dpdk: contains patches for DPDK libraries which are required for the vCMTS dataplane application

pktgen: contains patches for DPDK Pktgen which are required for vCMTS traffic simulation

"telemetry" subdirectory:
Contains files to build the telemetry components of the vCMTS reference dataplane platform, in the following sub-directories:

collectd: contains configuration files for Collectd

grafana: contains configuration files for Grafana dashboard

influxdb: contains configuration files for influxdb

docker: contains files to build docker images for Collectd, InfluxDB and Grafana.

"tools" subdirectory:
Contains Python tools which may be used to configure and manage the vCMTS reference dataplane platform, in the following sub-directories:

vcmts-env: contains shell scripts to configure the vCMTS dataplane and traffic generation environments

vcmts-cli: contains Python source modules for a command-line tool to display status and statistics for vCMTS dataplane application instances.

vcmts-pm: contains Python source modules for a platform management tool which performs the following functions:

- platform configuration based on hardware detection
- configuration of vCMTS dataplane application instances
- start/stop/re-start vCMTS dataplane application instances
- start/stop Pktgen application instances
- start/stop/rate control of Pktgen-generated traffic
- vCMTS dataplane throughput and latency measurement

"traffic-profiles" subdirectory:
Contains a zip package with control-plane configuration files for a number of vCMTS service-group scenarios and correlating PCAP traffic files for upstream and downstream traffic simulation. The zip-file included in the release package contains pcaps based on an iMix packet-size distribution.
Note that a separate zip package with fixed-size packet pcaps may be downloaded separately from the Intel 01.org site at the link below:
https://01.org/access-network-dataplanes/downloads
2 Installation Guide

2.1 vCMTS dataplane server preparation
A number of steps are required to prepare the vCMTS dataplane server for software installation.

⚠️ Please note: you must log in as root user to the vCMTS dataplane server to perform these steps.

2.1.1 Configure system BIOS settings
The following System BIOS settings are required on the vCMTS dataplane server.

<table>
<thead>
<tr>
<th>BIOS Setup Menu</th>
<th>BIOS Option</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced-&gt;Processor Configuration</td>
<td>Intel(R) Hyper-Threading Tech</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Integrated IO Configuration</td>
<td>Intel(R) VT for Directed I/O</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance</td>
<td>CPU Power and Performance profile</td>
<td>Balanced Performance</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPUP State Control</td>
<td>Intel(R) Turbo Boost Technology</td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td>NOTE: enable if SST-BF required</td>
<td></td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPUP State Control</td>
<td>Energy Efficient Turbo</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPUP State Control</td>
<td>Enhanced Intel Speed Select</td>
<td>Enabled</td>
</tr>
<tr>
<td></td>
<td>Technology</td>
<td></td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPUC State Control</td>
<td>C1E</td>
<td>Enabled</td>
</tr>
</tbody>
</table>
2.1.2 Check NIC Firmware and Driver versions

2.1.2.1 Intel 710 25G or 10G NIC's
If using Intel 710 25G or 10G NIC's, check the i40e base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

e.g. for network interface `ens785f1`

```
ethtool -i ens785f1
driver: i40e
version: 2.10.19.30
firmware-version: 8.10 0x80009424 1.1747.0
expansion-rom-version:
bus-info: 0000:18:00.1
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```

The driver version should be 2.10.19 or later.
The firmware version should be 8.10 or later.

If need to update, the required 710 NIC driver and firmware can be downloaded at the links below.

710 NIC Driver

710 NIC Firmware

2.1.2.2 Intel 810 100G NIC's
If using Intel 810 100G NIC's, check the ICE base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

e.g. for network interface `ens801f0`
The latest driver version available at the time of this vCMTS reference dataplane release is: 1.1.4
The compatible firmware version for this driver version is: 1.02

If need to update, the required 810 NIC driver and firmware can be downloaded at the links below.

810 NIC Driver

810 NIC Firmware

2.1.3  Check disk space settings

At least 400GB of disk-space should be available on the vCMTS dataplane server for the installation of the vCMTS reference dataplane software and infrastructure components.

See typical disk info below for a fully installed vCMTS dataplane server.

```
```
df -h
Filesystem Size Used Avail Use% Mounted on
udev 11G 0 11G 0% /dev
tmpfs 9.4G 875M 8.5G 10% /run
/dev/sdb1 440G 188G 229G 46% /
tmpfs 47G 0 47G 0% /dev/shm
tmpfs 5.0M 0 5.0M 0% /run/lock
tmpfs 47G 0 47G 0% /sys/fs/cgroup
```

It is recommended to disable swap space as described below.
First check if swap space is enabled, by running the following command.

```
blkid
/dev/sda1: UUID="08070332-90b8-45a5-891d-e76167ee876d" TYPE="ext4" PARTUUID="ee39ea61-01"
/dev/sda5: UUID="34c0a658-225b-4b28-abc8-8ac33c30b819" TYPE="swap" PARTUUID="ee39ea61-05"
```

If there is an entry such as TYPE="swap" in the above, this needs to be disabled by running the following command.

```
swapoff -a
```

Furthermore, swap space should be permanently disabled by commenting out any swap entry in the /etc/fstab file.

```
sed -i.bk '/ swap / s/^\(.*\)$/#1/g' /etc/fstab
```

On reboot, the disabling of swap space can be verified by running the following command, which should display the output below.

```
free -h
```

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th>used</th>
<th>free</th>
<th>shared</th>
<th>buff/cache</th>
<th>available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem:</td>
<td>62G</td>
<td>57G</td>
<td>239M</td>
<td>2.4M</td>
<td>4.7G</td>
<td>4.4G</td>
</tr>
<tr>
<td>Swap:</td>
<td>0B</td>
<td>0B</td>
<td>0B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.1.4 Check OS and Linux kernel versions

The recommended OS distribution is Ubuntu 20.04 LTS, and recommended Linux kernel version is 5.4.x or later.

Run the following command to check the OS distribution version

```
lsb_release -a
```

The following output is expected.

```
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 20.04.1 LTS
Release: 20.04
Codename: bionic
```

Run the following command to check the kernel version of the system:
The following type of output is expected:

```
5.4.0-47-generic
```

If a kernel version older than 5.4.x is displayed, the required kernel can be installed by running the following command:

```
apt-get update
```

### 2.1.5 Disable automatic Linux package updates

First, create the automatic upgrades system file by running the following command.

```
dpkg-reconfigure --priority-low unattended-upgrades
```

Select "Yes" to create the file.

Next, disable automatic package upgrades by editing the system file `/etc/apt/apt.conf.d/20auto-upgrades`, which was created above, as follows:

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Automatic package updates must also be disabled by editing the system file, `/etc/apt/apt.conf.d/10periodic`, as follows:

```
APT::Periodic::Update-Package-Lists "0";
```

### 2.1.6 Change file size limit settings

Add the following lines to the system file `/etc/security/limits.conf`

```
root    soft   FSIZE    unlimited
root    hard    FSIZE    unlimited
root    soft    NOFILE   unlimited
root    hard    NOFILE   unlimited
```
Changes will be applied upon next login to a bash shell.

This change is required as it has been known to cause deployment issues on some systems.

### 2.1.7 Configure Linux GRUB settings

The following Linux GRUB settings are required on the vCMTS dataplane server.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default_hugepagesz=1G</td>
<td>Huge-page memory size and number of pages reserved for DPDK applications</td>
</tr>
<tr>
<td>hugepagesz=1G</td>
<td></td>
</tr>
<tr>
<td>hugepages=72</td>
<td></td>
</tr>
<tr>
<td>NOTE: 2M hugepages also supported</td>
<td></td>
</tr>
<tr>
<td>intel_iommu=on</td>
<td>Enable SR-IOV</td>
</tr>
<tr>
<td>iommu=pt</td>
<td></td>
</tr>
<tr>
<td>isolcpus=2-23,26-47,50-71,74-95</td>
<td>Isolate vCMTS dataplane cores from the Linux kernel task scheduler</td>
</tr>
<tr>
<td>nr_cpus=96</td>
<td>Total number of logical cores on the system (aka hyper-threads)</td>
</tr>
</tbody>
</table>

The sample isolcpus and nr_cpus settings shown here are for a 24-core dual-processor CPU package. If not using a 24-core dual-processor CPU, the nr_cpus and isolcpus settings need to be adapted to the core-layout of the CPU package being used, as shown in the instructions that follow below. Note also that for the isolcpus setting, all except the first 2 cores on each CPU in the package must be isolated from the Linux kernel task scheduler.

The sample isolcpus and nr_cpus settings shown here are for a 24-core dual-processor CPU package. If not using a 24-core dual-processor CPU, the nr_cpus and isolcpus settings need to be adapted to the core-layout of the CPU package being used, as shown in the instructions that follow below. Note also that for the isolcpus setting, all except the first 2 cores on each CPU in the package must be isolated from the Linux kernel task scheduler.

Based on the above table for a 24-core dual-processor CPU package, the Linux kernel GRUB file /etc/default/grub should be edited as follows to set the appropriate GRUB_CMDLINE_LINUX options.

```bash
GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=72 intel_iommu=on iommu=pt isolcpus=2-23,26-47,50-71,74-95 nr_cpus=96"
```

⚠️ It is very important to verify the core layout of the CPU package being used on the target system and to adapt isolcpus and nr_cpus GRUB settings based on this.

The core layout of the CPU package can be verified by running the `lscpu` command as shown below.

The output shown here is for a 24-core dual-processor CPU package.

```bash
lscpu | grep "CPU(s):"
CPU(s):  96
NUMA node0 CPU(s):  0-23,48-71
NUMA node1 CPU(s):  24-47,72-96
```
Once editing of the Linux kernel GRUB file is complete, run the commands below to compile the GRUB configuration and reboot the server for updated settings to take effect:

```
update-grub2
reboot
```

### 2.1.8 Install public key for remote access to vCMTS traffic-generator server

A public key must be installed to allow remote access from vCMTS dataplane server to the traffic generator server. Replace the entry below with the actual vCMTS traffic-generator server hostname.

```
ssh-keygen -b 4096 -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@trafficgen-hostname
```

Select default options at each prompt.

### 2.1.9 Load the vCMTS reference dataplane package

Next, download the vCMTS reference dataplane package from 01.org to the vCMTS dataplane server and extract into the root directory of the installation, which is assumed to be /opt below.

```
cd /opt
wget https://01.org/sites/default/files/downloads/intel-vcmtsd-v20-10-0.tar.gz
tar -zxvf intel-vcmtsd-v20-10-0.tar.gz
cd vcmts
ls -lR
```

If downloading directly is unsuccessful, visit the Intel Access Network Dataplanes 01.org site at [https://01.org/access-network-dataplanes](https://01.org/access-network-dataplanes) and get the package from the Downloads section.

All files from the vCMTS reference dataplane package files as described in section 1.5, "Release Package Contents" should have been extracted to the /opt/vcmts directory.

Set the MYHOME and VCMTS_HOST environment variables as follows (assuming the vCMTS reference dataplane package will be installed in the /opt directory)
These should also be added as environment settings to the root bashrc file ~/.bashrc (again assuming that the vCMTS reference dataplane release package has been installed into /opt).

```
export VCMTS_HOST=y
export MYHOME=/opt
```

### 2.1.10 Configure proxy servers

⚠️ Please note: proxy configuration is different for each installation environment, so care should be taken to understand the required proxy settings for a particular environment as misconfiguration may greatly disrupt this installation.

The settings below need to be configured if the vCMTS dataplane server is behind a proxy. Note that proxy settings must also be applied as part of Docker installation steps.

Firstly, configure HTTP proxy servers and the no_proxy setting in the `$MYHOME/vcmts/tools/vcmts-env/proxy.sh` file.

The example entries below should be replaced with actual proxy address and port number and the actual hostname of the vCMTS dataplane server.

```
export http_proxy=http://myproxy.example.com:8080
export https_proxy=http://myproxy.example.com:8080
export no_proxy=localhost,127.0.0.1,vcmtsd-hostname,trafficgen-hostname
```

For Linux package installations, HTTP proxy server entries must also be added to the /etc/apt/apt.conf file.

```
Acquire::http::Proxy "http://myproxy.example.com:8080/";
Acquire::https::Proxy "http://myproxy.example.com:8080/";
```
2.1.11 Install Linux packages

A number of Linux packages must be installed on the vCMTS dataplane server which are required for the vCMTS reference dataplane runtime environment.

Run the environment function as shown below to install Linux packages required for the vCMTS dataplane server.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
install_base_ubuntu_pkgs
```

A power management utilities script is also required on the vCMTS dataplane server. Run the environment function as shown to download and install it.

```
install_power_mgmt_utilities
```
2.2 vCMTS traffic-generator server preparation

A number of steps are required to prepare the vCMTS traffic-generator server for software installation.

⚠️ Please note: you must log in as root user to the vCMTS traffic-generator server to perform these steps.

2.2.1 Configure system BIOS settings

The following System BIOS settings are required on the vCMTS traffic-generator server.

<table>
<thead>
<tr>
<th>BIOS Setup Menu</th>
<th>BIOS Option</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced-&gt;Processor Configuration</td>
<td>Intel(R) Hyper-Threading Tech</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Integrated IO Configuration</td>
<td>Intel(R) VT for Directed I/O</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance</td>
<td>CPU Power and Performance profile</td>
<td>Balanced Performance</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance---&gt;CPUP State Control</td>
<td>Intel(R) Turbo Boost Technology</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance---&gt;CPUP State Control</td>
<td>Energy Efficient Turbo</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance---&gt;CPUP State Control</td>
<td>Enhanced Intel Speed Select Technology</td>
<td>Enabled</td>
</tr>
</tbody>
</table>
2.2.2 Check NIC Firmware and Driver versions

2.2.2.1 Intel 710 25G or 10G NIC's
If using Intel 710 25G or 10G NIC's, check the i40e base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

```bash
ethtool -i ens785f1
```

```
driver: i40e
version: 2.10.19.30
firmware-version: 8.10 0x80009424 1.1747.0
```

The driver version should be 2.10.19 or later.
The firmware version should be 8.10 or later.

If need to update, the required 710 NIC driver and firmware can be downloaded at the links below.

**710 NIC Driver**

**710 NIC Firmware**

2.2.2.2 Intel 810 100G NIC's
If using Intel 810 100G NIC's, check the ICE base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

```bash
e.g. for network interface ens801f0
```

```bash
ethtool -i ens801f0
```

```
driver: ICE
version: 17.10.19.30
firmware-version: 8.10.19.0x80009424 1.1747.0
```

expansion-rom-version:
bus-info: 0000:18:00.1
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
The latest driver version available at the time of this vCMTS reference dataplane release is: 1.2.1
The compatible firmware version for this driver version is: 1.02

If need to update, the required 810 NIC driver and firmware can be downloaded at the links below.

**810 NIC Driver**

**810 NIC Firmware**

### 2.2.3 Check disk space settings

At least 400GB of disk-space should be available on the vCMTS traffic-generator server for the installation of the vCMTS reference dataplane software and related components.

See typical disk info below for a fully installed vCMTS traffic-generator server.

```
df -h

Filesystem            Size  Used  Avail %Mounted on
udev                   12G    0    12G  0% /dev
tmpfs                  9.5G   1.1G  8.5G  11% /run
/dev/sdb1              330G  150G  164G  48% /
tmpfs                  48G     0    48G  0% /dev/shm
tmpfs                  5.0M    0    5.0M  0% /run/lock
tmpfs                  48G    0    48G  0% /sys/fs/cgroup
```
It is recommended to disable swap space as described below.

First check if swap space enabled, by running the `blkid` command.

**e.g.**

```
blkid
/dev/sda1: UUID="08070332-90b8-45a5-891d-e76167ee876d" TYPE="ext4" PARTUUID="ee39ea61-01"
/dev/sda5: UUID="34c0a658-225b-4b28-abcb-8ac33c30b819" TYPE="swap" PARTUUID="ee39ea61-05"
```

If there is an entry such as `TYPE="swap"` in the above, this needs to be disabled by running the following command.

```
swapoff -a
```

Furthermore, swap space should be permanently disabled by commenting out any swap entry in the `/etc/fstab` file.

```
sed -i.bk '/ swap / s/^\(.*\)/#\1/g' /etc/fstab
```

On reboot, the disabling of swap space can be verified by running the following command, which should display the output below.

```
free -h
```

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th>used</th>
<th>free</th>
<th>shared</th>
<th>buff/cache</th>
<th>available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem:</td>
<td>62G</td>
<td>57G</td>
<td>239M</td>
<td>2.4M</td>
<td>4.7G</td>
<td>4.4G</td>
</tr>
<tr>
<td>Swap:</td>
<td>0B</td>
<td>0B</td>
<td>0B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.2.4 Check OS and Linux kernel versions

The recommended Linux kernel version is 5.4.x or later, and recommended OS distribution is Ubuntu 20.04 LTS.

Run the following command to check the OS distribution version

```
lsb_release -a
```

No LSB modules are available.

- Distributor ID: Ubuntu
- Description: Ubuntu 20.04.1 LTS
- Release: 20.04
- Codename: bionic
Run the following command to check the kernel version of the system:

```
uname -r
```

The following type of output is expected:

```
5.4.0-47-generic
```

If a kernel version older than 5.4.x is displayed, the required kernel can be installed by running the following:

```
apt-get update
```

### 2.2.5 Disable automatic Linux package updates

First, create the automatic upgrades system file by running the following command:

```
dpkg-reconfigure --priority-low unattended-upgrades
```

Select "Yes" to create the file.

Next, disable automatic package upgrades by editing the system file `/etc/apt/apt.conf.d/20auto-upgrades`, which was created above, as follows:

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Automatic package upgrades must also be disabled by editing the system file, `/etc/apt/apt.conf.d/10periodic`, as follows:

```
APT::Periodic::Update-Package-Lists "0";
```

### 2.2.6 Change file size limit settings

Add the following lines to the system file `/etc/security/limits.conf`
Changes will be applied upon next login to a bash shell.

This change is required as it has been known to cause deployment issues on some systems.

### 2.2.7 Configure Linux GRUB settings

The following Linux GRUB settings are required on the vCMTS traffic-generator server.

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Linux GRUB settings – vCMTS Traffic-generator server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>default_hugepagesizez=1G hugepagesizez=1G hugepages=72</td>
<td>Huge-page memory size and number of pages reserved for DPDK applications</td>
</tr>
<tr>
<td>NOTE: 2M hugepages also supported</td>
<td></td>
</tr>
<tr>
<td>intel_iommu=on iommu=pt</td>
<td>Enable SR-IOV</td>
</tr>
<tr>
<td>isolcpus=2-23,26-47,50-71,74-95</td>
<td>Isolate vCMTS dataplane cores from the Linux kernel task scheduler</td>
</tr>
<tr>
<td>NOTE: based on Dual 24-core CPU</td>
<td></td>
</tr>
<tr>
<td>nr_cpus=96</td>
<td>Total number of logical cores on the system (aka hyper-threads)</td>
</tr>
</tbody>
</table>

The sample isolcpus and nr_cpus settings shown here are for a **24-core dual-processor CPU package**. If not using a 24-core dual-processor CPU, the nr_cpus and isolcpus settings need to be adapted to the core-layout of the CPU package being used, as shown in the instructions that follow below. Note also that for the isolcpus setting, all except the first 2 cores on each CPU in the package must be isolated from the Linux kernel task scheduler.

Based on the above table for a **24-core dual-processor CPU package**, the Linux kernel GRUB file /etc/default/grub should be edited as follows to set the appropriate `GRUB_CMDLINE_LINUX` options.

```
GRUB_CMDLINE_LINUX="default_hugepagesizez=1G hugepagesizez=1G hugepages=72 intel_iommu=on iommu=pt isolcpus=2-23,26-47,50-71,74-95 nr_cpus=96"
```

⚠️ It is very important to verify the core layout of the CPU package being used on the target system and to adapt `isolcpus` and `nr_cpus` GRUB settings based on this.

The core layout of the CPU package can be verified by running the `lscpu` command as shown below. The output shown here is for a **24-core** dual-processor CPU package.
Once editing of the Linux kernel GRUB file is complete, run the commands below to compile the GRUB configuration and reboot the server for updated settings to take effect:

```bash
update-grub2
geru
```

### 2.2.8 Install public key for remote access to vCMTS dataplane server

A public key must be installed to allow remote access from the vCMTS traffic-generator server to the dataplane server. Replace the entry below with the actual vCMTS dataplane server hostname.

```bash
ssh-keygen -b 4096 -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@hostname
```

Select default options at each prompt.

### 2.2.9 Load the vCMTS reference dataplane package

Next, download the vCMTS reference dataplane package from 01.org to the vCMTS traffic-generator server from 01.org and extract into the root directory of the installation, assumed as /opt below.

Note that the same package is used for vCMTS traffic-generator server installation as was used for the vCMTS dataplane server.

```bash
cd /opt
wget https://01.org/sites/default/files/downloads/intel-vcmtsd-v20-10-0.tar.gz
tar -zxvf intel-vcmtsd-v20-10-0.tar.gz
cd vcmts
ls -lR
```

If downloading directly is unsuccessful, visit the Intel Access Network Dataplanes 01.org site at https://01.org/access-network-dataplanes and get the package from the Downloads section.
All files from the vCMTS reference dataplane package files as described in section 1.5, "Release Package Contents" should have been extracted to the /opt/vcms directory.

The vCMTS reference dataplane release package contains iMix style traffic-profiles by default. For RFC 2544 style benchmarking of a range of fixed packet sizes, an additional traffic-profile package may be downloaded as follows.

```bash
cd /opt/vcms/traffic-profiles
wget https://01.org/sites/default/files/downloads/intel-vcmtds-fixedsz-tp-20.10.0.tar.bz2
```

The fixed-sz traffic profile package from the previous vCMTS reference dataplane release version is used for this release version.

Set the MYHOME and PKTGEN_HOST environment variables as follows (assuming the vCMTS reference dataplane package is installed in the /opt directory)

```bash
export PKTGEN_HOST=y
export MYHOME=/opt
```

These must also be added as environment settings to the root bashrc file ~/.bashrc (again assuming that the vCMTS reference dataplane release package has been installed into /opt).

```bash
export PKTGEN_HOST=y
export MYHOME=/opt
```

### 2.2.10 Configure proxy servers

⚠️ Please note: proxy servers are different for every installation environment, so care should be taken to understand the required proxy settings for a particular environment as misconfiguration may greatly disrupt this installation.

The settings below need to be configured if the vCMTS traffic generator server is behind a proxy.

Note that proxy settings must also be applied as part of Docker and CMK installation steps.

Firstly, configure HTTP proxy servers and the no_proxy setting in the $MYHOME/vcms/tools/vcms-env/proxy.sh file.

The example entries below should be replaced with actual proxy address and port number and the actual hostname of the vCMTS traffic generator server.
For Linux package installations, HTTP proxy server entries must also be added to the /etc/apt/apt.conf file.

```
export http_proxy=http://myproxy.example.com:8080
export https_proxy=http://myproxy.example.com:8080
export no_proxy=localhost,127.0.0.1,trafficgen-hostname,vcmtsd-hostname
```

2.2.11 Install Linux packages

A number of Linux packages must be installed on the vCMTS traffic-generator server which are required for the vCMTS reference dataplane runtime environment.

Run the environment function as shown below to install Linux packages required for the vCMTS traffic-generator server (assuming installation in /opt).

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
install_base_ubuntu_pkgs
```

2.2.12 Install the vCMTS platform management tool

The vcmts-pm tool is provided with the vCMTS reference dataplane release package. The vcmts-pm tool simplifies the configuration, running and management of vCMTS applications on the reference platform whether using a Kubernetes* or Bare-metal Linux* environment. It provides the following functionality for both Bare-metal and Kubernetes* environments:

- configure vCMTS dataplane platform environment settings
- configure vCMTS traffic-generator platform environment settings
- configure vCMTS service-group options
- start/stop vCMTS infrastructure components
- start/stop vCMTS and traffic-generator instances
- control traffic simulation
- measure vCMTS dataplane throughput and latency capability
- query vCMTS platform and service-group configurations
The `vcmts-pm` tool should only be run from the Kubernetes controller node (which also acts as the vCMTS traffic-generator server). This applies to both Kubernetes* and Bare-metal Linux* environments.

```
    cd $VCMTS_ROOT/tools/vcmts-pm
    sudo apt-get install python3-pip
    pip3 install virtualenv
    python3.6 -m virtualenv env
    source env/bin/activate
    pip3 install -e .
    deactivate
```

The `vcmts-pm` tool is now installed.
2.3 Configure vCMTS platform settings

Platform configurations must be generated for the vCMTS dataplane and traffic-generator servers in order for the Intel vCMTS reference dataplane runtime environment to operate correctly.

⚠️ Please note: both platform configurations must be generated from the vCMTS traffic-generation server (also Kubernetes controller node) and you must log in as root user to perform the steps described below.

Follow the instructions in the next two sections to configure the runtime environments for vCMTS dataplane and traffic-generator platforms, using the vcmts-pm tool provided in the vCMTS reference dataplane release package.

2.3.1 vCMTS dataplane platform configuration

Firstly, the vCMTS dataplane platform environment configuration is performed.

While the tool is run on the vCMTS traffic-generation server (also Kubernetes controller node), the actual platform details are gathered remotely from the vCMTS dataplane platform.

The following platform details need to be gathered for the vCMTS dataplane server runtime environment.

- Fully qualified domain-name of the vCMTS Dataplane server
- CPU information - number of CPU's and cores per CPU (isolated and not isolated)
- NIC information - NIC physical functions per NUMA node and their PCI addresses
- QAT information - QAT physical functions per NUMA node and their PCI addresses

Run the vcmts-pm tool provided in the vCMTS reference dataplane release package as follows to specify the settings for the vCMTS dataplane platform configuration.

```
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
vcmts-pm config-platform vcmtsd
deactivate
```

Follow the menu and prompted instructions to generate the vCMTS dataplane platform configuration.

⚠️ Please note: you must specify the fully qualified domain-name for server addresses e.g, myhost.example.com, as hostname alone is not sufficient.
2.3.2 vCMTS traffic-generator platform configuration

Next the vCMTS traffic-generator platform environment configuration is performed.

The following platform details need to be gathered for the vCMTS traffic-generator (also known as Pktgen) server runtime environment.

- Fully qualified domain-name of the vCMTS traffic-generation server
- CPU information - number of CPU’s, cores per CPU (isolated and not isolated) and Pktgen application instance mappings
- NIC information - NIC physical functions per NUMA node and their PCI addresses

Note that CMK (CPU Manager for Kubernetes) is not used on the traffic-generator platform so the core mappings to Pktgen application instances are assigned by the vcmts-pm tool in this case.

Run the vcmts-pm tool provided in the vCMTS reference dataplane release package as follows to specify the settings for the vCMTS traffic-generator (pktgen) platform configuration.

```
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
vcmts-pm config-platform pktgen
deactivate
```

Follow the menu and prompted instructions to generate the vCMTS traffic-generator (Pktgen) platform configuration.

⚠️ Please note: you must specify the fully qualified domain-name for server addresses e.g. myhost.example.com, as hostname only is not sufficient.
2.4 Install Kubernetes

vCMTS dataplane and pktgen (traffic-generator) application instances can be orchestrated using Kubernetes*.

Note that this section should only be completed if a Kubernetes* orchestrated environment is required. For a Linux* bare-metal installation, skip this entire section.

Some Kubernetes add-on components are later installed for CPU core management (CMK) and management of QuickAssist resources (QAT device plugin). Helm is also later installed for configuration of vCMTS dataplane and pktgen POD's.

As shown in Figure 1, the Kubernetes controller node installation must be performed on the vCMTS traffic-generator server. Kubernetes Node installation must be performed on both the vCMTS dataplane and traffic-generator servers.

The following sections cover the steps for Kubernetes installation on the vCMTS reference dataplane system. Note that the installation is performed through environment functions and scripts which are provided in the vCMTS reference dataplane package.

⚠️ Please note: you must log in as root user to the respective vCMTS traffic-generator server (also Kubernetes controller node) and vCMTS dataplane node to perform these steps.

2.4.1 Install and configure docker on the vCMTS dataplane server

To install the community edition of Docker run the following commands on the vCMTS dataplane server.

```
cd $MYHOME/vcmts
export VCMTS_HOST=y
source tools/vcmts-env/env.sh
install_docker
```

Some further steps are required to configure Docker to run behind a proxy server and to configure a DNS server address.

The following steps are required to configure Docker to run behind an HTTP or HTTPS proxy server.

Create a systemd directory for the docker service.
Create a file called `/etc/systemd/system/docker.service.d/http-proxy.conf` and replace the entries below with the actual proxy address and port number which is applicable to the system.

```
[Service]
Environment="HTTP_PROXY=http://myproxy.example.com:8080/" "HTTPS_PROXY=http://myproxy.example.com:8080/"
"NO_PROXY=localhost,127.0.0.1"
```

If the installation environment uses a domain name server, Docker may need to perform domain name resolution when installing packages during container builds.

By default, Docker uses Google's domain name server, 8.8.8.8, however if a local domain name server exists, this should be specified as described below.

Create a file called `/etc/docker/daemon.json` with the content shown below, specifying the local DNS server IP address which is applicable to the installation environment.

```
{
    "dns": ["<local-dns-server-ip-address>", "8.8.8.8"]
}
```

Now start Docker by running the following commands.

```
systemctl daemon-reload
systemctl start docker
systemctl show --property=Environment docker
```

If it is necessary to restart Docker at any point, the following command should be used.

```
systemctl restart docker
```

### 2.4.2 Install and Configure Docker on the vCMTS traffic-generator server

To install the community edition of Docker run the following commands on the vCMTS traffic-generator server.
Some further steps are required to configure Docker to run behind a proxy server and to configure a DNS server address.

The following steps are required for Docker to run behind an HTTP or HTTPS proxy server.

Create a systemd directory for the docker service.

```
mkdir -p /etc/systemd/system/docker.service.d/
```

Create a file called `/etc/systemd/system/docker.service.d/http-proxy.conf` and replace the entries below with the actual proxy address and port number which is applicable to the system.

```
[Service]
Environment="HTTP_PROXY=http://myproxy.example.com:8080/" "HTTPS_PROXY=http://myproxy.example.com:8080/" "NO_PROXY=localhost,127.0.0.1"
```

If the installation environment uses a domain name server, Docker may need to perform domain name resolution when installing packages during container builds.

By default, Docker uses Google’s domain name server, 8.8.8.8, however if a local domain name server exists, this should be specified as described below.

Create a file called `/etc/docker/daemon.json` with the content shown below, specifying the local DNS server IP address which is applicable to the installation environment.

```
{
   "dns": ["<local-dns-server-ip-address>", "8.8.8.8"]
}
```

Now start Docker by running the following commands.

```
systemctl daemon-reload
systemctl start docker
systemctl show --property=Environment docker
```
2.4.3 Install Kubernetes controller node on the vCMTS traffic-generator server

The vCMTS traffic-generator server acts as Kubernetes controller node.

To install Kubernetes controller node on vCMTS traffic-generator server, log in to it as root user.

After logging in, first set the vcmts reference dataplane environment by running the following command.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

To install Kubernetes controller node, run the environment function provided in the release package, as follows, and follow instructions when prompted.

```bash
install_kubernetes_controller_node
```

⚠️ Note that the Kubernetes installation takes a few minutes and should NOT be interrupted.

Once this is complete, Kubernetes controller node is installed and running on the vCMTS traffic-generator server.

2.4.4 Install Kubernetes node on the vCMTS dataplane server

The vCMTS dataplane server acts as a Kubernetes node.

To install Kubernetes node on vCMTS dataplane server, log in to it as root user.

After logging in, first set the vcmts reference dataplane environment by running the following command.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

To install Kubernetes node, run the environment function provided in the release package, as follows, and follow instructions when prompted.
Note that the Kubernetes installation takes a few minutes and should NOT be interrupted.

Kubernetes Node is now installed and running on the vCMTS dataplane server.

### 2.4.5 Verify Kubernetes installation

A successful Kubernetes installation may be verified by running the following command on the Kubernetes controller node (traffic-generator server).

```
kubectl get nodes
```

The output of this command should indicate both nodes, traffic-generator (pktgen) and vCMTS dataplane as ‘Ready’.

Each Kubernetes node must be labelled as a pktgen or vcmts node.

Label the nodes by running the commands below on the Kubernetes controller node.

It is important that the node name used for `pktgen_node` and `vcmtspktgen` is the name returned from the ‘`kubectl get nodes`’ command above.

```
kubectl label nodes pktgen_node vcmtspktgen=true
kubectl label nodes vcmtspktgen=true
```

For a more detailed verification of the installation, run the `check_kubernetes` environment function on the Kubernetes controller node and node.

On the Kubernetes controller node, output such as shown below is expected.
source $MYHOME/vcmts/tools/vcmts-env/env.sh

check_kubernetes

• etcd.service - Etcd Server
  Loaded: loaded (/lib/systemd/system/etcd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-09-30 15:49:09 IST; 1 weeks 5 days ago

• kube-apiserver.service - Kubernetes API Server
  Loaded: loaded (/lib/systemd/system/kube-apiserver.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-09-30 15:49:13 IST; 1 weeks 5 days ago

• kube-scheduler.service - Kubernetes Scheduler Plugin
  Loaded: loaded (/lib/systemd/system/kube-scheduler.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-09-30 15:49:13 IST; 1 weeks 5 days ago

• kube-controller-manager.service - Kubernetes Controller Manager
  Loaded: loaded (/lib/systemd/system/kube-controller-manager.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-09-30 15:49:13 IST; 1 weeks 5 days ago
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Drop-in: /etc/systemd/system/docker.service.d
    └─http-proxy.conf
      Active: active (running) since Wed 2020-09-30 15:49:15 IST; 1 weeks 5 days ago

• kubelet.service - Kubernetes Kubelet Server
  Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-09-30 15:57:59 IST; 1 weeks 5 days ago

• flanneld.service - Flanneld overlay address etcd agent
  Loaded: loaded (/lib/systemd/system/flanneld.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-09-30 15:49:15 IST; 1 weeks 5 days ago

• kube-proxy.service - Kubernetes Kube-Proxy Server
  Loaded: loaded (/lib/systemd/system/kube-proxy.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-09-30 15:49:15 IST; 1 weeks 5 days ago

On the Kubernetes vcmtsd node, output such as shown below is expected.
If a service has failed it is recommended to attempt to restart the service as follows:

systemctl status <service>

systemctl stop <service>

systemctl start <service>
2.5 Connect NIC Interfaces

As stated previously in section 1.2, if a switch is not used in the setup, the network interfaces of the vCMTS application and vCMTS traffic-generator servers may need to be directly connected to allow simulation traffic to flow.

In this case, the corresponding links must be connected correctly to one another.

The following steps ensure that the correct connections are made. This section can optionally be ignored if a switch is configured between vCMTS dataplane and traffic-generator servers.

1. Run the following commands on the vCMTS dataplane server then go to step 2.
   ```
   source $MYHOME/vcmts/tools/vcmts-env/env.sh
cable_pf_helper
   ```

2. Run the following commands on the vCMTS traffic-generator server then go to step 3.
   ```
   source $MYHOME/vcmts/tools/vcmts-env/env.sh
cable_pf_helper
   ```

3. There will be console output from both vCMTS dataplane and traffic-generator servers at this point. A NIC interface on both the vCMTS dataplane and traffic-generator server will also be flashing. Using optical fiber cables, connect the flashing interface on the vCMTS dataplane server to the flashing interface on the vCMTS traffic-generator server.

4. Press Enter key on the console on the vCMTS dataplane server once.
   ```
   This function helps to correctly cable vCMTS and PKTGEN nodes
   Run this function in parallel on both vCMTS and PKTGEN nodes
   Wire the flashing ports on each node together
   Then press enter on both systems and continue
   press enter once link 1 is cabled:
   ```

5. Press Enter key on the console on the vCMTS traffic-generator server once.
   ```
   This function helps to correctly cable vCMTS and PKTGEN nodes
   Run this function in parallel on both vCMTS and PKTGEN nodes
   Wire the flashing ports on each node together
   Then press enter on both systems and continue
   press enter once link 1 is cabled:
   ```

6. Go back to step 3 until all interfaces have been connected and the script exits.
2.6 Install vCMTS dataplane node software

The following sections describe installation of software components required on the vCMTS dataplane node. These steps should be performed on the vCMTS dataplane server after logging in as root user.

After logging in to the vCMTS dataplane server, first set the vCMTS reference dataplane environment by running the following command.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

2.6.1 Generate openssl certificates

A cli application is provided as part of the release package that communicates with vCMTS dataplane applications over TLS connection allowing for detailed statistics to be queried. Openssl certificates are required to facilitate this connection. Generate them with the following commands.

```
cd $MYHOME
generate_openssl_certs
```

⚠️ Note that it is important that this command completes successfully, as any error here may later cause vCMTS datalane application initialization failure.

2.6.2 Install QAT drivers

Install QAT drivers on the vCMTS dataplane node host OS by running the following command.

```
cd $MYHOME
install_qat_drivers
```

The PCI addresses of the QAT cards on the system can be checked by running the command shown below.

```
lspci -d:37c8
```

```
8a:00.0 Co-processor: Intel Corporation Device 37c8 (rev 04)
8c:00.0 Co-processor: Intel Corporation Device 37c8 (rev 04)
8e:00.0 Co-processor: Intel Corporation Device 37c8 (rev 04)
```
2.6.3 Install Intel® IPSEC MB Library

Install Intel® IPSEC MB Library on the vCMTS dataplane node host OS by running the following command.

```
   cd $MYHOME
   build_baremetal_ipsec_mb
```

2.6.4 Install DPDK

Install DPDK on the vCMTS dataplane node host OS by running the following command.

```
   cd $MYHOME
   build_baremetal_dpdk
   build_baremetal_collectd
```

2.6.5 Install Linux* bare-metal specific software

This sub-section only needs to be completed by users installing a Linux* Bare-metal environment. Section 2.6.6 describes the software installation of Kubernetes* specific software.

2.6.5.1 Build telemetry components on the vCMTS dataplane server

The following telemetry components are used by the vCMTS reference dataplane system.

- **Collectd**: used to collect vcmits dataplane and platform statistics
- **InfluxDB**: used to store vcmits dataplane and platform statistics
- **Grafana**: used to visualize vcmits dataplane and platform statistics

Run the following commands to build bare-metal installations for the telemetry components listed above.

```
   build_baremetal_collectd
   build_baremetal_influxdb
   build_baremetal_grafana
```

The collectd build includes download and build of the **intel_pmu** and **intel_rdt collectd plugins**.
A sample collectd.conf.template is provided which loads all of the plugins required on the vCMTS dataplane node. It will be updated with platform specific information at a later stage by the vcmts-pm tool. Also, vCMTS dataplane statistic types are installed in the collectd types DB based on the vcmts.types.db file provided in the vCMTS reference dataplane release package.

NOTE: The intel_pmu and intel_rdt collectd plugins are disabled (i.e. not loaded) by default as these may impact dataplane performance at very high traffic-rates. If these need to be enabled in order to display platform and memory bandwidth metrics, the collectd.conf file may be manually edited to load these plugins. If this is desired run the following commands on both the vcmtsd and pktgen servers.

```
cd $MYHOME/vcmts/telemetry/collectd
cp collectd.conf.template_no_pmu collectd.conf.template_no_pmu.orig
cp collectd.conf.template collectd.conf.template_no_pmu
```

### 2.6.5.2 Build vCMTS dataplane application
Build the vCMTS dataplane application by running the following command.

```
build_baremetal_vcmtsd
```

### 2.6.6 Install Kubernetes specific software
This sub-section only needs to be completed by users installing for a Kubernetes* orchestrated environment. Section 2.6.5 describes the software installation of Linux* Bare-metal specific software.

#### 2.6.6.1 Install CMK
CMK (Core Manager for Kubernetes) is a key component which manages allocation of cores to POD's under Kubernetes control. This is installed on the vCMTS dataplane node only.

On the vCMTS dataplane node, run the following commands to download and build CMK.

```
cd $MYHOME
build_docker_cmk
```

The version of CMK built here contains some updates to CMK v1.4.1, which are required for the Intel vCMTS reference dataplane platform.

The CMK docker image should have been built and tagged with v1.4.1.
To verify, run the following command to display docker images on the system.

```
docker images
```

This should display 'cmk:v1.4.1' for CMK.

### 2.6.6.2 Build Kubernetes infrastructure components on the vCMTS dataplane server

Run the following commands to build docker container images for the Kubernetes infrastructure components required on the vCMTS dataplane server for platform initialization, power-management, and QAT and NIC resource management.

```
build_docker_cloud_init --no-cache
build_docker_power_mgr --no-cache
build_docker_sriov_dp --no-cache
build_docker_qat --no-cache
```

### 2.6.6.3 Build telemetry components on the vCMTS dataplane server

The following telemetry components are used by the vCMTS reference dataplane system.

- **Collectd**: used to collect vcmts dataplane and platform statistics
- **InfluxDB**: used to store vcmts dataplane and platform statistics
- **Grafana**: used to visualize vcmts dataplane and platform statistics

Run the following commands to build docker container images for the telemetry components listed above.

```
build_docker_collectd --no-cache
build_docker_influxdb
build_docker_grafana --no-cache
```

The collectd docker build includes download and build of the `intel_pmu` and `intel_rdt` collectd plugins.

A sample `collectd.conf` is provided which loads all of the plugins required on the vCMTS dataplane node. Also, vCMTS dataplane statistic types are installed in the collectd types DB based on the `vcmts.types.db` file provided in the vCMTS reference dataplane release package.

**NOTE:** The `intel_pmu` and `intel_rdt` collectd plugins are disabled (i.e. not loaded) by default as these may impact dataplane performance at very high traffic-rates. If these need to be enabled in order to display platform and memory bandwidth metrics, the `collectd.conf` file may be manually edited to load these plugins. If this is desired run the flowing commands on both the vcmts and pktgen servers.
2.6.6.4 Build vCMTS dataplane application container

Build the vCMTS dataplane docker image by running the following command.

```bash
cd $MYHOME/vcmts/telemetry/collectd
cpy collectd.conf.template_no_pmu collectd.conf.template_no_pmu.orig
ncpy collectd.conf.template collectd.conf.template_no_pmu
build_docker_vcmtsd
```
2.7 Install vCMTS traffic-generator node software

The following sections describe installation of software components required on the vCMTS traffic-generator node. These steps should be performed after logging in as root user to the vCMTS traffic-generator server (which also acts as the Kubernetes controller node).

After logging in to the vCMTS traffic-generator node, set the vCMTS reference dataplane environment by running the following command.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

2.7.1 Install DPDK

Install DPDK on the vCMTS traffic-generator node host OS by running the following commands.

```
cd $MYHOME
build_baremetal_dpdk
```

2.7.2 Install Linux* baremetal specific software

This sub-section only needs to be completed by users installing a Linux* bare-metal environment. Section 2.7.2 describes the software installation of Kubernetes* specific software.

2.7.2.1 Build Pktgen application

Run the following command to build the DPDK Pktgen application.

```
build_baremetal_pktgen
```

2.7.3 Install Kubernetes specific software

This sub-section only needs to be completed by users installing for a Kubernetes* orchestrated environment. Section 2.7.2 describes the software installation of Linux* bare-metal specific software.
2.7.3.1 Install Helm
Helm is used by the Kubernetes controller node to configure service-group information for vCMTS dataplane and traffic-generator instances.

Run the following commands to install Helm on the vCMTS traffic-generator node, which also acts as the Kubernetes controller node managing all application containers on the system.

```
cd $MYHOME
install_helm
```

If the installation is being done behind a proxy an empty repository file must be created for Helm, as follows:

```
mkdir -p /root/.helm/repository
touch /root/.helm/repository/repositories.yaml
```

2.7.3.2 Build Kubernetes infrastructure components on the traffic-generator server
Run the following commands to build docker container images for the Kubernetes infrastructure components required on the vCMTS traffic-generator server for platform initialization and NIC resource management.

```
build_docker_cloud_init
build_docker_sriov_dp --no-cache
```

2.7.3.3 Build Pktgen application container
Run the following command to build the DPDK Pktgen docker image.

```
build_docker_pktgen
```
## 2.8 Configure vCMTS dataplane

Each vCMTS dataplane instance (POD in Kubernetes* environment) handles upstream and downstream traffic for a single service-group, which includes a number of cable subscribers, typically hundreds.

The following service-group options may be configured.

### Table 5  vCMTS dataplane configurable service-group options

<table>
<thead>
<tr>
<th>Configurable Service-Group Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment Type</td>
<td>Environment Type - one of two settings described below. baremetal: all components run on Linux* Host OS kubernetes: all components orchestrated by Kubernetes* and run in Docker* containers <strong>Default</strong>: baremetal</td>
</tr>
<tr>
<td>Power Management</td>
<td>The following Power Mgmt options are offered: Branch-monitor: branch hit/miss ratio used to determine business of cores and set frequency Time-of-day: a time based policy is used to set frequency of cores Disabled: no power mgmt. applied <strong>Default</strong>: disabled</td>
</tr>
<tr>
<td>Power busy hours</td>
<td>List of hours for busy periods of time-of-day based power policy Only applies when time-of-day based power mgmt. has been selected</td>
</tr>
<tr>
<td>SST-BF Configuration</td>
<td>SST-BF Configuration: This option allows the user to configure the prioritizaton of Service Groups through use of SST-BF capabilities when present. Options: DS only prioritization, US+DS prioritization, none <strong>Default</strong>: none</td>
</tr>
<tr>
<td>SST-BF-SGs</td>
<td>Number of Service Groups to use High Priority SST-BF Cores, default is max-possible</td>
</tr>
<tr>
<td>Number of Service-Groups</td>
<td>Number of service-groups. <strong>Default</strong>: Max based on core configuration, and number of CPU cores and NW ports</td>
</tr>
<tr>
<td>Number of Subscribers</td>
<td>100, 300, 500 or 1000 subscribers per service-group <strong>Default</strong>: 300</td>
</tr>
<tr>
<td>Core configuration</td>
<td>Core configuration of service-groups 2us1t_1ds2t: 2 US per core (single threaded), 1 DS per core (dual threaded) (DEFAULT) 2us1t_1ds1t: 2 US per core (single threaded), 1 DS per core (single threaded) 1us1t_1ds2t: 1 US per core (single threaded), 1 DS per core (dual threaded)</td>
</tr>
<tr>
<td>Channel Configuration</td>
<td>1us1t_1ds1t: 1 US per core (single threaded), 1 DS per core (single threaded) 1us1t_1ds2t2c: 1 US per core (single threaded), 1 DS on 2 cores (dual threaded) 2us1t_2ds1t: 2 US per core (single threaded), 2 DS per core (single threaded) 1usds1t: Single threaded US+DS, 1 per core 2usds1t: Single threaded US+DS, 2 per core Default: 2us1t_1ds2t</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| CM Crypto Configurations | One of the following channel configurations:  
1 x OFDM, 32 x SC-QAM  
2 x OFDM, 32 x SC-QAM  
4 x OFDM, 0 x SC-QAM  
6 x OFDM, 0 x SC-QAM Default: 6xOFDM |
| AES Key Size | AES key size (applies to all service groups using AES crypto type)  
128, 256 Default: 128 |
| CPU Cycle Count Stats Capture | Enable or Disable CPU Cycle count statistics capture per service group  
NOTE: Enabling CPU Cycle count statistics has an impact on dataplane performance Default: Disable Cycle count statistics capture |
| Application Stats Capture | Enable or Disable detailed application statistics capture per service group  
NOTE: Enabling detailed application statistics has an impact on dataplane performance Default: Disable detailed application statistics capture |
| Latency Stats Capture | Enable or Disable Latency statistics capture per service group Default: Disable Latency statistics capture |
| Downstream CRC re-generation | Enable or Disable CRC re-generation for downstream DOCSIS frames  
NOTE: CRC is disabled for upstream traffic Default: Enable CRC |
| Downstream Crypto Offload | Enable or Disable QuickAssist offload for downstream encryption Default: Disable QAT |
| Traffic type | Select iMix or fixed-sized packets for cable traffic.  
iMix1:  
Upstream 65%: 84B, 18%: 256B, 17%: 1280B |
The following settings are applied to all service-groups and are not configurable.

### Table 6  vCMTS dataplane fixed service-group options

<table>
<thead>
<tr>
<th>Fixed Settings</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscriber Lookup</td>
<td>4 IP addresses per subscriber</td>
</tr>
<tr>
<td>DOCSIS Filtering</td>
<td>16 filters per cable-modem</td>
</tr>
<tr>
<td></td>
<td>10% matched (permit rule), 90% unmatched (default action - permit)</td>
</tr>
<tr>
<td>DOCSIS Classification</td>
<td>16 IPv4 classifiers per cable-modem</td>
</tr>
<tr>
<td></td>
<td>10% matched, 90% unmatched (default service-flow queue)</td>
</tr>
<tr>
<td>Downstream Service-Flow Scheduling</td>
<td>8 service-flow queues per cable-modem (4 active)</td>
</tr>
<tr>
<td>Downstream Channel Scheduling</td>
<td>42.24 Mbps Bandwidth per SQ-QAM channel</td>
</tr>
<tr>
<td></td>
<td>1.89 Gbps Bandwidth per OFDM channel</td>
</tr>
<tr>
<td></td>
<td>99% profile density ratio for OFDM channels</td>
</tr>
<tr>
<td>Downstream Channel Bonding Groups</td>
<td>1xOFDM, 32xSC-QAM</td>
</tr>
<tr>
<td></td>
<td>BG1: Channel 0-23 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG2: Channel 0-15, 24-31 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG3: Channel 0-7, 16-31 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG4: Channel 8-31 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>2xOFDM, 32xSC-QAM</td>
</tr>
<tr>
<td></td>
<td>BG1: Channel 0-23 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG2: Channel 0-15, 24-31 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG3: Channel 0-7, 16-31 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG4: Channel 8-31 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>4xOFDM</td>
</tr>
<tr>
<td></td>
<td>BG1: Channel 0,1 (OFDM)</td>
</tr>
</tbody>
</table>
The maximum amount of downstream traffic that needs to be handled for a service group is determined by its number of OFDM and SC-QAM channels as shown in the table below. It is assumed that upstream traffic is 10% of downstream traffic.

### Table 7  Service-group bandwidth guide

<table>
<thead>
<tr>
<th>Number OFDM Channels (1.89 Gbps)</th>
<th>Additional SC-QAM Channels (42.24 Mbps)</th>
<th>Total Downstream Bandwidth per SG * (Gbps)</th>
<th>Max Total BW (US = 10% DS) (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>3.24</td>
<td>3.56</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>5.13</td>
<td>5.64</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>7.56</td>
<td>8.32</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>11.34</td>
<td>12.47</td>
</tr>
</tbody>
</table>

* Actual downstream channel bandwidth is reduced by DOCSIS MAC and Phy overhead

**NOTE:** Key points to consider when selecting service-group options.
- DES cable-modems are only supported for the 1 OFDM channel configuration
- total available QAT crypto bandwidth should be taken into account when selecting number of service-groups for QAT crypto offload option (noting that QAT offload is only used for downstream traffic)
- when using 10G NICs with 6xOFDM channel configuration, the Ethernet port bandwidth limit may be reached before downstream channel bandwidth limit

The required service-group settings may be configured on the system using the `vcmts-pm` tool provided in the vCMTS dataplane release package, as described in the next section.

### 2.8.1 Configure vCMTS dataplane service-group options

Follow the steps below to specify service-group options for vCMTS dataplane instances on the system.
These steps should be performed as root user on the Kubernetes controller node (also the traffic-generator/pktgen server).

After logging in to the Kubernetes controller node, set the environment for running the vcmts-pm tool.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
```

Then, run the vcmts-pm tool provided in the vCMTS dataplane release package as follows to specify the required service-group options for the vCMTS dataplane environment.

```
vmts-pm config-service-groups
```

Follow the prompts to select service-group settings and generate Helm charts for the Kubernetes cluster or configuration files for baremetal deployment. The CMK cluster-init.yaml file for core-pool configuration is also updated based on the number of service-groups selected.

Service-group options may also be selected by passing command-line arguments to the vcmts-pm tool, as described below.

For example, to configure a Kubernetes environment with 16 service-groups using the 2us1t_1ds2t core configuration with 4 OFDM channels, 300 subscribers, all AES-128 CM's, CRC enabled, SG's 8 to 11 with QAT offload, SG 1 with Latency stats and iMix-2 traffic, run the following command:

```
vmts-pm config-service-groups -u -i kubernetes --core-config 2us1t_1ds2t -n 16 -c 4ofdm -s 300 -d aes -k 128 -v enabled -q "8,9,10,11" -z "1" -t imix2
```

A full description of vcmts-pm command-line arguments for service-group configurations can be examined by running the following command.

```
vmts-pm config-service-groups --help
```
2.9 Run vCMTS dataplane and traffic-generator software

The following sections describe how to start, stop and re-start the vCMTS reference dataplane and traffic-generator software, and also how to verify that the software is running correctly.

These steps should be performed after logging in as root user to the Kubernetes controller node (also the traffic-generator/pktgen server).

Note that these steps remain the same regardless of whether a Kubernetes* or Bare-metal Linux* environment has been selected. All commands use the vcmts-pm tool which detects the environment type and runs the appropriate command.

After logging in to the Kubernetes controller node, set the vcmts reference dataplane environment by running the following command.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
```

2.9.1 Start infrastructure and application instances

Run the following environment functions on the Kubernetes controller node to start vCMTS dataplane and Pktgen application instances as well as the supporting infrastructure and telemetry components.

```
vcmts-pm infra-start
vcmts-pm vcmtsd-start
vcmts-pm pktgen-start
```

In a Kubernetes* environment the output of ‘vcmts-pm infra-start’ looks as follows.

```
# vcmts-pm infra-start
Do you want to start CMK ? [y/N]: y
Do you want to start Kubernetes Dashboard ? [y/N]: y
Do you want to start Kubernetes DNS ? [y/N]: y
Do you want to start init ? [y/N]: y
```

Conversely on a baremetal Linux* environment it looks like this.
Verify that all components started correctly by running the following commands.

```
# vcmts-pm infra-start

Do you need to re-configure pktgen ports (required after vcmts-pm config-service-groups) ? [y/N]: y
Do you need to re-configure vcmtsd ports (required after vcmts-pm config-service-groups) ? [y/N]: y
```

```
vcmts-pm infra-status
vcmts-pm vcmtsd-status
vcmts-pm pktgen-status
```

The Kubernetes log function can be used to diagnose vcmtsd application issues in a Kubernetes* environment. e.g. the following commands may be run to check logs of upstream and downstream containers for vcmtsd pod number 0 (i.e. service-group ID 0)

```
kubectl logs pod/vcmtsd-0 vcmtsd-0-ds
kubectl logs pod/vcmtsd-0 vcmtsd-0-us
```

### 2.9.2 How to stop infrastructure and application instances

To stop vCMTS dataplane and Pktgen application instances as well as supporting infrastructure and telemetry components, run the following commands.

```
vcd ts-pm pktgen-stop
vcmts-pm vcmtsd-stop
vcmts-pm infra-stop
```
2.10 System verification
The following sections describe how to verify that the system is running correctly.

2.10.1 Check Kubernetes dashboard
Kubernetes infrastructure and application POD status may be checked on the Kubernetes WebUI dashboard. This is only applicable in a Kubernetes* deployment.
Firstly, set the vcmts reference dataplane environment by running the following command.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

Next, run the following command to display the Kubernetes WebUI dashboard URI and login details.

```
show_kubernetes_dash
```

Open a web-browser at the URI displayed and use the token to sign in to the Kubernetes WebUI dashboard as shown below.
Note that if accessing from a web-browser on a remote client an SSH tunnel to the WebUI dashboard service port on the Kubernetes controller node may be required.

Figure 8  Kubernetes WebUI Login Screen

![Kubernetes WebUI Login Screen](image)

Once signed in, the Kubernetes WebUI dashboard should be displayed as shown below.
All vcmtsd and pktgen pods should be displayed as running.
2.10.2 Attach to application instances

The `vcmts-pm` tool may be used to generate instructions on how to directly attach to specific vCMTS dataplane or pktgen applications. Run the following commands for instructions on how to attach directly to an application instance running in either Kubernetes* or baremetal Linux* environment.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
vcmts-pm pktgen-attach -i 0 us
vcmts-pm vcmtsd-attach -i 0 us
```

**NOTE:** after attaching to a process issuing Ctrl+C command will terminate the application.
2.10.3 Query service-group options
The vcmts-pm tool may be used as shown below to check that the correct service-group options have been applied. Follow the steps below to query service-group options for vCMTS dataplane instances on the system. After logging in to the vCMTS traffic-generator server, set the environment for running the vcmts-pm tool.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
```

Then, run the vcmts-pm tool provided in the vCMTS dataplane release package as follows to query configuration settings for all service-group.

```
vcmts-pm query-platform
vcmts-pm query-service-groups -i 0 usds -q all
```

Configuration settings for each vCMTS service-group should be displayed. It may take up to 1 minute to gather information for all service-groups.

2.10.4 Start dataplane traffic
This section covers how to generate simulated cable traffic into the vCMTS dataplane server. Upstream and downstream DOCSIS dataplane traffic may be simulated by running the vcmts-pm tool provided in the vCMTS dataplane release package. These steps should be performed as root user on the Kubernetes controller node (also the traffic-generator/pktgen server).

After logging in to the Kubernetes controller node, set the environment for running the vcmts-pm tool.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
```

Next, start traffic. Note that because dataplane NW interface VF’s are dynamically allocated to POD’s by the Kubernetes SR-IOV device plugin, an ARP request-response handshake is required between correlating Pktgen and vCMTS application instances for correct routing of traffic. This is implicitly performed before starting to send traffic when the Pktgen start command is sent below.
For example, to start traffic at 5Gbps (20% of 25G line-rate) for downstream and 0.5Gbps (2% of 25G line-rate) for upstream for 16 vCMTS dataplane instances (0 to 15), run the following commands:

```
vcmts-pm traffic-rate -i 0-15 us -r 2
vcmts-pm traffic-rate -i 0-15 ds -r 20
vcmts-pm traffic-start -i 0-15 usds
```

To reduce the traffic-rate to 2.5Gbps (10% of 25G line-rate) for downstream and 0.25Gbps (1% of 25G line-rate) for upstream for 16 vCMTS dataplane instances (0 to 15), run the following commands.

```
vcmts-pm traffic-rate -i 0-15 us -r 1
vcmts-pm traffic-rate -i 0-15 ds -r 10
```

To stop traffic, run the following command.

```
vcmts-pm traffic-stop -i 0-15 usds
```

If using 10G NICs the pktgen-rate values above are percentages of 10G line-rate instead of 25G and if using 100G NICs the pktgen-rate values above are percentages of 100G line-rate.

An RFC 2544 throughput measurement test may also be run. For example, to measure downstream throughput for service-group 0, run the following command.

```
vcmts-pm traffic-measure -i 0 ds -m rx
```

### 2.10.5 Check the Grafana dashboard

vCMTS dataplane and platform metrics may be checked on a Grafana dashboard. First, log in to the vCMTS dataplane node as root user.

After logging in, first set the vcmts reference dataplane environment by running the following command.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

Next, run the following command to display the Grafana dashboard details.

```
show_grafana_dash
```
Open a web-browser at the displayed URI and use the login credentials to login to the Grafana dashboard as shown below.

If accessing from a web-browser on a remote client an SSH tunnel to the Grafana dashboard service port on the vCMTS dataplane node may be required.

**Figure 10  Grafana Dashboard Login Screen**

Once signed in, it should be verified that the Influxdb data-source has been added to Grafana successfully. For instructions to do this, see Appendix section 3.2.

Once signed in and the data-source has been verified, select the required vCMTS dataplane dashboard by clicking on Dashboards Home.

*e.g.* the System Summary dashboard is shown below.
Figure 11 Intel vCMTS Dataplane Reference Platform – System Summary Dashboard
2.10.5.1 How to modify Grafana platform metrics dashboard for different CPU core counts

The Grafana "Platform Metrics" dashboard has been configured in the release package for a 20-core dual processor platform by default. If the CPU core-count is different, the dashboard will show incorrect platform metrics.

To rectify this, the "Platform Metrics" dashboard queries must be updated as described below.

The queries in each individual graph on the Platform Metrics page must be updated by selecting "Edit" from the drop down menu as shown below.

Figure 12 CPU Utilization Graph on Grafana Platform Metrics Dashboard
Then, update as shown below for each individual graph to ensure that a query is present for each logical core (i.e. hyper-thread) on the system.

**Figure 13  CPU Utilization Query on Grafana Platform Metrics Dashboard**

Once the required updates have been made, save the dashboard by clicking the "Save" icon and selecting "Save JSON to file".
2.10.6 vCMTS CLI Tool

The vCMTS reference dataplane system includes a utility application that provides a command line interface to query status and statistics for each vCMTS service-group which is active on the system. Note that for each vCMTS service-group there may be a vCMTS upstream and downstream process running.

The CLI application should be run from the vCMTS dataplane server. It can be run with the following commands. Type 'help' to view the usage.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd tools/vcmts-cli
vcmts-cli
```

In order to connect to a running vCMTS service group, it needs to be added to the CLI with the following command in the CLI shell.

```
vcmts-cli> add vcmts <service-group-id> <ip> <tls-port-1> <tls-port-2>
```

The service group ID of the required service group should be entered in place of `<service-group-id>`. The IP address of the vCMTS dataplane server should be entered in place of `<ip>`.

The TLS port number of the vCMTS upstream application instance should be entered in place of `<tls-port-1>` and the TLS port number of the vCMTS downstream application instance should be entered in place of `<tls-port-2>`.

The upstream TLS port number may be calculated as follows: 8100 + service-group-id.

The downstream TLS port number may be calculated as follows: 8200 + service-group-id.

Once added to the CLI, vCMTS service-group statistics can be queried through the CLI tool as shown below.

```
vcmets-cli> help
+----------------------------------------------------------+ vCMTS CLI Command List +----------------------------------------------------------+
help                               - Prints help string
quit                               - Exits CLI program
add vcmts <service-group-id> <ip> <tls-port-1> [tls-port-2]  - Save given Service Group information
del <service-group-id>             - Deletes saved Service Group information
list                               - Displays saved Service group information
status <service-group-id>          - Requests status of given Service Group
summary                            - Displays a summary of Service Groups
stats <service-group-id> <stats-type>  - Query stats from running vCMTS instance
```
2.11 System reboot

After reboot of vCMTS dataplane and traffic-generation servers, core components such as Docker and Kubernetes should start automatically.

To restart vCMTS dataplane and traffic-generator software the following commands should be run on the Kubernetes controller node (also the traffic-generator/pktgen server), after logging in as root user:

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
vcmts-pm infra-start
vcmts-pm vcmtsd-start
vcmts-pm pktgen-start
```

Verify that all components started correctly by running the following commands:

```
vcmts-pm infra-status
vcmts-pm vcmtsd-status
vcmts-pm pktgen-status
```

Perform system verification as described in previous section.
3 Appendix

3.1 DOCSIS DDP Forwarding Sample Application

3.1.1 Description

This sample application (docsis-ddp-fwd) demonstrates how the Dynamic Device Profile (DDP) feature of the Intel 800 series NIC can be used to enable Cable-specific NIC pre-processing to steer DOCSIS MAC traffic using the Flow-director NIC feature for upstream traffic and the Receiver Side Scaling (RSS) NIC feature for downstream traffic.

Flow-director traffic-steering rules are based on L2TPv3 session ID which indicates the type of upstream frame according to DOCSIS MAC protocol specifications. The L2TPv3 session ID flow-director feature is enabled in the Intel 800 series NIC through use of the Dynamic Device Profile (DDP) feature of Intel NIC’s.

Incoming packets are steered to Virtual Function (VF) queues based on the criteria below:

1. Upstream L2TPv3 traffic is steered to 1 or 2 VF queues (2 by default) based on the L2TPv3 session ID:
   a) If 2 queues are selected, L2TPv3 packets are steered to separate VF queues, one for OFDMA channel frames and one for SC-QAM channel frames.
   b) If 1 queue is selected all L2TPv3 packets are sent to the same VF queue.
2. Downstream (non-L2TPv3) packets are distributed by RSS based on the packets IP 5-tuple to either 2 or 3 VF queues (3 by default)

Each queue is polled by a dedicated DPDK lcore thread.
NIC VF receive and transmit statistics are collected and printed by the DPDK init lcore thread.

Note that this sample app should be used in conjunction with DPDK Pktgen and custom-generated pcaps are provided to demonstrate the use of NIC traffic steering rules.

Note also that this application will only work with an Intel 800 series 100G NIC Virtual Function with latest NIC firmware and L2TPv3 DDP enabled.
3.1.2 User Guide

See below for instructions to install and run the `docsis-ddp-fwd` sample application and to generate traffic to demonstrate DOCSIS MAC traffic steering by the Intel 800 series 100G NIC.

3.1.2.1 Installing the DOCSIS DDP Fwd sample application

Follow these steps to Install the sample application (on the vCMTS server):

Ensure the Intel 800 series NIC driver and DDP package are loaded.

Download links:


Use ICE Driver version >=1.1.4
Use ICE Comms package version >=1.3.16.0

Untar the ICE driver package, go to the driver src directory and install the ICE driver.

```
tar -xvf ice-1.1.4.tar.gz
cd ice-1.1.4/src/
make -j 8
make install
```

Unzip the ICE Comms package, copy the extracted package to the ICE DDP directory. Remove the previous ICE comms package from the ICE DDP directory and rename the new ICE comms package.

```
unzip 617296_ice_comms-1.3.17.0.zip
cp ice-1.3.17.0.pkg /lib/firmware/updates/intel/ice/ddp/
rm /lib/firmware/updates/intel/ice/ddp/ice.pkg
mv /lib/firmware/updates/intel.ice/ddp/ice_comms-1.3.17.0.pkg /lib/firmware/updates/intel/ice/ddp/ice.pkg
```

If a previous ICE driver is loaded, unload it. Load the new ICE driver and check the ICE driver was loaded correctly.
If a line containing "version: 1.1.4" is present, the ICE driver was loaded correctly.

Check for successful DDP package installation.

```
journalctl -n 200
dmesg | grep ice
```

If a line containing "The DDP package was successfully loaded: ICE COMMS Package version 1.3.17.0" is present, the DDP package was installed correctly.

Set the environment normally used for vCMTS.

```
export VCMTSD_HOST=y
source $VCMTS_ROOT/tools/vcmts-env/env.sh
```

Install base packages, install DPDK (if not already installed) and build the sample application.

```
install_base_ubuntu_pkgs
build_baremetal_dpdk
cd $VCMTS_ROOT/src/docsis-ddp-fwd
make
```

The DOCSIS DDP Fwd application has now been successfully installed on the vCMTS dataplane server.

It is assumed that traffic-generation software has already been installed on the vCMTS traffic-generator server. This can be used to generate traffic for the DDP sample application.
3.1.2.2 Running the DOCSIS DDP Fwd sample application

The following steps describe how to run the sample DOCSIS DDP Fwd sample application.

Select a device from the list of PCI devices on the system and create VF's for it.

```
dpk-devbind.py -s
echo 4 > /sys/class/net/<device-name>/device/sriov_numvfs
```

Bind one of the created VF's to DPDK for use by the sample application.

```
dpk-devbind.py -s
dpk-devbind.py -b igr_uio <vf-name>
```

Launch the sample application using the helper script provided.

```
$VCMTS_ROOT/src/docsis-ddp-fwd/start_docsis_ddp_fwd.sh -l [core_list] -w [pci_address]
```

The `docsis-ddp-fwd` helper script accepts the following command-line arguments:

- `-l [cores]`: Comma separated list of cores to use
  - e.g. `-l 2,3,4,5,6,7`
  - note: number of cores must equal the total number of queues + 1
  - e.g. for 4 queues, 5 cores must be allocated
  - note: defaults to 6 cores (5 queues)

- `-w [pci address]`: PCI address of virtual function to be used
  - e.g. `-w 18:01.1`

- `-r [nb of RSS queues]`: Number of RSS queues for downstream traffic
  - note: option, default 3, max 3

- `-t [nb of flow-director queues]`: Number of Flow-Director queues for upstream traffic
  - note: options, default 2, max 2
Below are additional command-line arguments used by the *docsis-ddp-fwd* application, which are defaulted by the helper script:

```
--: Used to separate DPDK EAL arguments and application arguments

-p [bitmask]: hexadecimal bitmask of VF ports to configure
  note: defaults to 0x1

-n|--ip-addr-network [ip address]: IP address used for ARP requests to sample applications
downstream interface
  note: must be set to 192.168.1.100

-y|--ip-address-rphy [ip address]: IP address used for ARP requests to sample applications
upstream interface
  note: must be set to 192.168.1.200

-c [period]: statistics generation period in seconds
  note: defaults to 1 second

-s [period]: statistics print period in seconds
  note: defaults to 1 seconds
```

Note that the *docsis-ddp-fwd* sample application can also be run without the helper script and use all of the available arguments described above.

### 3.1.2.3 Generating traffic into the DOCSIS DDP Fwd sample application

Once the DOCSIS DDP Fwd application is running, Pktgen is required to send traffic to the sample application.

First, log into the Pktgen server and set the vCMTS Pktgen environment.

Note that it is assumed that the Pktgen environment has already been installed using the vCMTS traffic-generation installation procedure.

```
export PKTGEN_HOST=y
source $VCMTS_ROOT/tools/vcmts-env/env.sh
```
Then, run Pktgen as follows using the provided helper script.

```bash
$VCMTS_ROOT/src/docsis-ddp-fwd/start_docsis_ddp_fwd_pktgen.sh \
   -l [core list] (Only 4 cores must be used and the selected cores must be on the 
   same socket as the PCI device) \
   -w [US pci address] \ 
   -w [DS pci address] \ 
   -p [imix1|imix2] (optional, default is imix2)
```

The full list of Pktgen helper script command-line arguments are as follows:

- `-l [core list]`: Comma separated list of CPU cores to which the Pktgen threads will be affinitized
  
  *note*: only 4 cores in list are supported and the PCI device and selected cores must be on the same socket. e.g. `-l 2,3,4,5`

- `-w [pci address]`: Virtual Function PCI address to use for Pktgen network interface
  
  *note*: two ports must be provided, one for the upstream interface and second for the downstream interface. The PCI device and selected cores must be on the same socket.

- `-p [imix1|imix2]`: PCAP type to use

There are also additional command-line arguments used by the pktgen application which are defaulted by the helper script. Some are listed below.
Note that the Pktgen application can also be run without the helper script and use all of the available arguments described above.

### 3.2 How to Verify/Create Grafana Data-source for vCMTS Telemetry

Once signed in to Grafana, it should be verified that the Influxdb data-source has been added to Grafana successfully. This can be done with the following steps.

From the Grafana Home page select the 'Data Sources' option from the 'Configuration' submenu.
Verify that the Influxdb data-source is present and configured as the default.

If no data-source is present the influxdb data-source must be added manually as follows.
Select the 'Add data source' button.
Select 'Influxdb' as the data-source type and fill out the fields as follows, set user/password to admin/admin.
Finally click the 'save and test' button to create the data-source.