### 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 v18.10.2</td>
<td>Feb 28th 2019</td>
</tr>
</tbody>
</table>
Introduction

Related Information

This install guide relates to v18.10.2 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 DEATH MAY 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.

© 2019 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 ........................................................................................................................................ 8
1.1 System overview ............................................................................................................................. 8
  1.1.1 vCMTS Reference Dataplane Pipeline ......................................................................................... 10
  1.1.2 vCMTS Reference Dataplane NFV Stack ...................................................................................... 11
  1.1.3 CPU Core Management ................................................................................................................ 12
1.2 Network Interface Configuration ..................................................................................................... 15
1.3 Memory Module Configuration ....................................................................................................... 16
1.4 System Configuration ..................................................................................................................... 17
1.5 Release Package Contents .............................................................................................................. 18

2 Installation Guide .............................................................................................................................. 21
2.1 vCMTS dataplane server preparation ............................................................................................... 21
  2.1.1 Configure system BIOS settings .................................................................................................. 21
  2.1.2 Check Fortville NIC Firmware and Driver versions ...................................................................... 21
  2.1.3 Check disk space settings .......................................................................................................... 22
  2.1.4 Check OS and Linux kernel versions .......................................................................................... 23
  2.1.5 Disable automatic Linux package updates .................................................................................. 24
  2.1.6 Configure Linux GRUB settings ................................................................................................. 24
  2.1.7 Install public key for remote access to vCMTS traffic-generator server ...................................... 26
  2.1.8 Load the vCMTS reference dataplane package ......................................................................... 26
  2.1.9 Configure proxy servers ............................................................................................................ 27
  2.1.10 Install Linux packages .............................................................................................................. 28
  2.1.11 Configure Docker .................................................................................................................... 28
  2.1.12 Install the vCMTS dataplane control tool ................................................................................ 29

2.2 vCMTS traffic-generator server preparation .................................................................................... 31
  2.2.1 Configure system BIOS settings ................................................................................................ 31
  2.2.2 Check Fortville NIC Firmware and Driver versions ...................................................................... 31
  2.2.3 Check disk space settings .......................................................................................................... 32
  2.2.4 Check OS and Linux kernel versions .......................................................................................... 33
  2.2.5 Disable automatic Linux package updates .................................................................................. 34
  2.2.6 Configure Linux GRUB settings ................................................................................................. 34
  2.2.7 Install public key for remote access to vCMTS dataplane server ................................................ 36
  2.2.8 Load the vCMTS reference dataplane package ......................................................................... 36
  2.2.9 Configure proxy servers ............................................................................................................ 37
  2.2.10 Install Linux packages .............................................................................................................. 38
  2.2.11 Configure Docker .................................................................................................................... 38
  2.2.12 Install vCMTS dataplane configuration and control tools .......................................................... 39

2.3 Configure vCMTS platform settings ................................................................................................. 41
  2.3.1 vCMTS dataplane platform configuration .................................................................................. 41
  2.3.2 vCMTS traffic-generator platform configuration ....................................................................... 42
Introduction

2.4 Install Kubernetes
   2.4.1 Install Kubernetes master and node on the vCMTS traffic-generator server ..... 44
   2.4.2 Install Kubernetes node on the vCMTS dataplane server 
   2.4.3 Verify Kubernetes installation 
2.5 Install vCMTS dataplane node software
   2.5.1 Install QAT drivers 
   2.5.2 Install DPDK 
   2.5.3 Install CMK 
   2.5.4 Build Kubernetes infrastructure components on the vCMTS dataplane server 
   2.5.5 Build telemetry components on the vCMTS dataplane server 
   2.5.6 Build vCMTS dataplane application container 
2.6 Install vCMTS traffic-generator node software
   2.6.1 Install DPDK 
   2.6.2 Install Helm 
   2.6.3 Build Kubernetes infrastructure components on the traffic-generator server 
   2.6.4 Build Pktgen application container 
2.7 Configure vCMTS dataplane
   2.7.1 Configure vCMTS dataplane service-group options 
   2.7.2 Configure vCMTS dataplane power profiles 
2.8 Run vCMTS dataplane and traffic-generator software
   2.8.1 Start infrastructure and application instances 
   2.8.2 How to stop infrastructure and application instances 
   2.8.3 How to restart the statistics daemonset 
2.9 System verification
   2.9.1 Check Kubernetes dashboard 
   2.9.2 Query service-group options 
   2.9.3 Start dataplane traffic 
   2.9.4 Check the Grafana dashboard 
2.10 System reboot 
3 Upgrade Procedure 
   3.1 vCMTS traffic-generator server preparation
   3.1.1 Update Linux GRUB settings 
   3.1.2 Stop all Kubernetes-orchestrated components 
   3.1.3 Backup current installation on vCMTS traffic-generation server 
   3.1.4 Load the new vCMTS reference dataplane package 
   3.1.5 Restore proxy server settings 
   3.1.6 Upgrade vCMTS dataplane configuration and control tools 
   3.2 vCMTS dataplane server preparation
   3.2.1 Update Linux GRUB settings 
   3.2.2 Backup current installation on vCMTS dataplane server 
   3.2.3 Load the new vCMTS reference dataplane package 
   3.2.4 Restore proxy server settings
3.2.5 Upgrade vCMTS dataplane control tool ........................................... 79
3.3 Re-configure vCMTS platform settings .................................................. 81
  3.3.1 vCMTS dataplane platform configuration .......................................... 81
  3.3.2 vCMTS traffic-generator platform configuration ................................. 82
3.4 Apply security updates for Kubernetes .................................................. 84
  3.4.1 Kubernetes updates on vCMTS traffic-generation server ....................... 84
  3.4.2 Kubernetes updates on vCMTS dataplane server .................................. 84
3.5 Upgrade vCMTS traffic-generator node software ..................................... 86
  3.5.1 Upgrade DPDK ...................................................................................... 86
  3.5.2 Re-build Kubernetes infrastructure components on the traffic-generator server 86
  3.5.3 Re-build Pktgen application container .................................................. 87
3.6 Upgrade vCMTS dataplane node software ............................................... 88
  3.6.1 Upgrade DPDK ...................................................................................... 88
  3.6.2 Re-build Kubernetes infrastructure components on the vCMTS dataplane server 89
  3.6.3 Re-build telemetry components on the vCMTS dataplane server ............ 89
  3.6.4 Re-build vCMTS dataplane application container .................................. 90
3.8 Run vCMTS dataplane and traffic-generation software ................................ 91
4 Appendix ....................................................................................................... 92
  4.1 Sample platform configurations ............................................................... 92
    4.1.1 vCMTS common platform configuration ............................................ 92
    4.1.2 vCMTS dataplane platform configuration .......................................... 93
    4.1.3 vCMTS traffic-generator platform configuration ................................. 95
  4.2 Sample Kubernetes infrastructure configuration file .................................. 96
  4.3 Sample Helm chart for vCMTS POD’s ..................................................... 97
    4.3.1 vCMTS dataplane helm chart for 16 service-groups ............................ 97
    4.3.2 vCMTS pktgen helm chart for 16 service-groups ............................... 99
  4.4 Sample CMK configuration file ................................................................... 101
  4.5 Upgrading Linux kernel and Linux packages .......................................... 103
    4.5.1 Upgrade Linux on vCMTS traffic-generator server ............................ 103
    4.5.2 Upgrade Linux on vCMTS dataplane server ...................................... 104
List of Figures

Figure 1  Intel vCMTS Reference Dataplane – Sample Platform for 16 service-groups at max-load 9
Figure 2  Intel vCMTS reference dataplane - packet-processing pipeline ....................... 10
Figure 3  Intel vCMTS reference dataplane – NFV stack ........................................ 11
Figure 4  Intel vCMTS Dataplane Reference Platform – core layout for 16 service-groups at max-load 12
Figure 5  Intel vCMTS Dataplane Reference Platform – core layout for 24 service-groups at max-load 13
Figure 6  Intel vCMTS Dataplane Reference Platform – core layout for 32 service-groups at low-speed 14
Figure 7  Intel vCMTS Dataplane Reference Platform – Power Management Architecture 61

List of Tables

Table 1  System BIOS settings – vCMTS Dataplane server ........................................ 21
Table 2  Linux GRUB settings – vCMTS Dataplane server ........................................... 25
Table 3  System BIOS settings – vCMTS Traffic-generator server ................................. 31
Table 4  Linux GRUB settings – vCMTS Traffic-generator server .................................. 34
Table 5  vCMTS dataplane configurable service-group options .................................... 54
Table 6  vCMTS dataplane fixed service-group options ................................................. 55
Table 7  Service-group bandwidth guide ........................................................................ 56
Table 8  Service-group configuration tool - commandline options ............................... 57
1 Introduction

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

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 6148 processors and four 25G Intel® Ethernet Network Adapter XXV710-DA2 dual port network interface cards (NIC's). The vCMTS dataplane node described in this document also includes an Intel® QuickAssist Technology (QAT) 8970 PCIe card.

Note that the Intel® 10G quad port X710-DA4 NIC's may optionally be used instead of 25G NIC's. And if enough PCI slots are available on the platform, up to 3 NIC's 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 with different core-counts are also supported. More NIC's 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.

On the vCMTS dataplane node, multiple Docker* containers host DPDK-based DOCSIS MAC upstream and downstream dataplane processing for individual cable service-groups (SG's). On the vCMTS traffic-generation node Docker* containers host DPDK Pktgen-based traffic generation instances which simulate DOCSIS traffic into corresponding vCMTS dataplane instances.

The entire system is orchestrated by Kubernetes*, under which each vCMTS dataplane POD represents a service-group and has separate containers 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*. 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 supports vCMTS dataplane processing for 16 service-groups. Such a system can be used to measure maximum traffic load per service-group, with up to 6 OFDM channels (at 1.89 Gbps B/W each) allocated per service-group.
NOTE: The above shows a system configuration with dual-port 25G NIC's. In case of 10G quad-port NIC's, there would be 4 PF's per NIC, and 2 VF's per PF.

In case of a low-speed configuration with 1 OFDM channel per service-group, twice the number of service-groups could be configured. In this case there would be 8 VF's per PF for 25G NIC's, and 4 VF's per PF for 10G NIC's.

It is assumed that upstream traffic rates are 10% of downstream traffic rates.

In the above sample platform there are 20 cores per CPU and each CPU core contains two hyperthreads. For max-load configuration, an entire physical CPU core is allocated for downstream data processing of a single service-group. For upstream data processing, a single physical core handles upstream traffic for two service-groups.

For a low-speed configuration (with 1 OFDM channel per service-group), downstream data processing for two service-groups run on the same core, and a single physical core handles upstream traffic for four service-groups.

See section 1.1.3 for CPU core allocation details.
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 2 below. Both upstream and downstream 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 2 Intel vCMTS reference dataplane - packet-processing pipeline](image)

Each of the Kubernetes POD's shown in the reference system in Figure 1 contains an instantiation of the above upstream and downstream pipelines in separate Linux containers. Each POD thus 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 3 below.

vCMTS upstream and downstream dataplane processing for individual cable service-groups run in Docker\footnote{Docker} containers on Ubuntu OS, allowing them to be instantiated and scaled independently.

The entire system including applications, telemetry, power-management and infrastructure management is orchestrated by Kubernetes\footnote{Kubernetes}, with Intel-developed plugins being used for resource management functions such as CPU core management and assignment of SR-IOV interfaces for NIC’s and QAT devices.

**Figure 3 Intel vCMTS reference dataplane – NFV stack**
### 1.1.3 CPU Core Management

CPU core allocation is managed by CMK (CPU Manager for Kubernetes) which is an open-source component provided 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 at max-load).

In the sample configuration shown below, CPU cores are allocated separately for upstream and downstream dataplane for each service-group. CPU cores for processing of downstream traffic are allocated from the exclusive core-pool, and cores for processing of upstream traffic are allocated from the shared core-pool, with 2 upstream dataplane instances per core. The detailed CPU core layout for such a 16 service-group system is shown in Figure 4 below.

#### Figure 4 Intel vCMTS Dataplane Reference Platform – core layout for 16 service-groups at max-load

<table>
<thead>
<tr>
<th>CPU 0</th>
<th>CPU 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>79</td>
</tr>
</tbody>
</table>

**Legend:**

- Infrastructure
- Telemetry
- Upstream Dataplane (2 instances per core)
- Downstream Dataplane (1 instance per core)
- Unused - reserved for Control-plane, Management-plane, HA redundancy

**NOTE:** For the above CPU core configuration unused cores are reserved for compute resources that would be used by DOCSIS MAC components not running on Intel's reference vCMTS platform such as Upstream scheduler, Control-plane, MAC management and standby dataplane instances for High-Availability.
If there are enough PCIe slots available on the system, two additional NIC’s may be added for more dataplane I/O and the unused cores in Figure 4 may be used to handle this extra dataplane traffic. In this case up to 24 service-groups may be deployed to handle max traffic load per service-group. The detailed CPU core layout for such a 24 service-group system is shown in Figure 5 below.

**Figure 5  Intel vCMTS Dataplane Reference Platform – core layout for 24 service-groups at max-load**

![CPU Core Layout Diagram](Image)

**NOTE:** For the above CPU core configuration, all of the available 40 cores on the system are allocated so that there are no cores reserved for DOCSIS MAC components not running on Intel's reference vCMTS platform such as Upstream scheduler, Control-plane, MAC management and HA.

If using a 22 or 28 core CPU package, some cores may be reserved for such components.
For initial vCMTS deployments there may be no more than a single OFDM channel available per service-group, so that an entire CPU core may not be necessary to handle downstream traffic for a service-group. In this case it is desirable for 2 such service-groups running at low traffic speed to share a core for downstream traffic processing.

In such a low-speed configuration, CPU cores for upstream and downstream dataplane traffic processing are now both allocated from the shared core-pool, with 2 downstream instances per core and 4 upstream instances per core. The detailed CPU core layout for such a 32 service-group system is shown in Figure 6 below.

**Figure 6  Intel vCMTS Dataplane Reference Platform – core layout for 32 service-groups at low-speed**

<table>
<thead>
<tr>
<th>CPU 0</th>
<th>CPU 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>29</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>31</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
</tr>
<tr>
<td>13</td>
<td>33</td>
</tr>
<tr>
<td>14</td>
<td>34</td>
</tr>
<tr>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>16</td>
<td>36</td>
</tr>
<tr>
<td>17</td>
<td>37</td>
</tr>
<tr>
<td>18</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>21</td>
<td>41</td>
</tr>
<tr>
<td>22</td>
<td>42</td>
</tr>
<tr>
<td>23</td>
<td>43</td>
</tr>
<tr>
<td>24</td>
<td>44</td>
</tr>
<tr>
<td>25</td>
<td>45</td>
</tr>
<tr>
<td>26</td>
<td>46</td>
</tr>
<tr>
<td>27</td>
<td>47</td>
</tr>
<tr>
<td>28</td>
<td>48</td>
</tr>
<tr>
<td>29</td>
<td>49</td>
</tr>
<tr>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>31</td>
<td>51</td>
</tr>
<tr>
<td>32</td>
<td>52</td>
</tr>
<tr>
<td>33</td>
<td>53</td>
</tr>
<tr>
<td>34</td>
<td>54</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
<tr>
<td>36</td>
<td>56</td>
</tr>
<tr>
<td>37</td>
<td>57</td>
</tr>
<tr>
<td>38</td>
<td>58</td>
</tr>
<tr>
<td>39</td>
<td>59</td>
</tr>
</tbody>
</table>

**NOTE:** For the above CPU core configuration unused cores are reserved for DOCSIS MAC components not running on Intel’s reference vCMTS platform such as Upstream scheduler, Control-plane, MAC management and HA.

If there are enough PCIe slots available for two additional NIC’s, all of the available 40 cores on the system may be allocated so that there are no cores reserved for such components. In this case, it is possible to deploy up to 48 low-speed service-groups.
1.2 Network Interface Configuration

The network interface ports of the vCMTS dataplane and traffic-generation nodes shown in Figure 1 should be inter-connected 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.

When vCMTS dataplane traffic is started as described in section 2.9.3 an ARP request is sent from each traffic-generator instance to establish a link with its correlating vCMTS dataplane instance.

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

```bash
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 6148 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.

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

```
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 1 Mem Read (MB/s): 13933.60</td>
</tr>
<tr>
<td>NODE 0 Mem Write (MB/s): 10113.44</td>
<td>NODE 1 Mem Write (MB/s): 12088.41</td>
</tr>
<tr>
<td>NODE 0 P. Write (T/s): 18804</td>
<td>NODE 1 P. Write (T/s): 18866</td>
</tr>
<tr>
<td>NODE 0 Memory (MB/s): 21566.15</td>
<td>NODE 1 Memory (MB/s): 26022.00</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>

```

Intel vCMTS Reference Dataplane v18.10.2
# 1.4 System Configuration

The following is a sample system configuration for Intel vCMTS reference dataplane v18.10.2.

**vCMTS Node**

<table>
<thead>
<tr>
<th><strong>Hardware</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel® Xeon® Gold 6148 Dual Processor, 2.4 GHz, 20 Cores</td>
</tr>
<tr>
<td>Memory</td>
<td>12 x 8GB DDR4</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>Intel® SSD DC S3520 Series (480G)</td>
</tr>
<tr>
<td>Network Interface Card</td>
<td>4 x Intel® Ethernet Network Adapter XXV710-DA2 25GbE or 4 x Intel® Ethernet Network Adapter X710-DA4 10GbE</td>
</tr>
<tr>
<td>Crypto Acceleration Card</td>
<td>Intel® QuickAssist Adapter 8970 Card (50Gbps)</td>
</tr>
</tbody>
</table>

**Software**

| **Host OS**                        | Ubuntu* 18.04, Linux* Kernel v4.15.0 |
| **DPDK**                           | DPK v18.11, intel-ipsec-mb v0.51       |
| **vCMTS**                          | Intel vCMTS Reference Dataplane v18.10.2|
| **Linux Container**                | Docker* v18.06.1                       |
| **Container Orchestrator**         | Kubernetes* v1.13, CMK v1.3.0          |
| **Statistics**                     | Collectd* v5.8.0 (+)                   |

**vCMTS Traffic Generator Node**

<table>
<thead>
<tr>
<th><strong>Hardware</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel® Xeon® Gold 6148 Dual Processor, 2.4 GHz, 20 Cores</td>
</tr>
<tr>
<td>Memory</td>
<td>12 x 8GB DDR4</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>Intel® SSD DC S3520 Series (480G)</td>
</tr>
<tr>
<td>Network Interface Card</td>
<td>4 x Intel® Ethernet Network Adapter XXV710-DA2 25GbE or 4 x Intel® Ethernet Network Adapter X710-DA4 10GbE</td>
</tr>
</tbody>
</table>

**Software**

| **Host OS**                        | Ubuntu* 18.04, Linux* Kernel v4.15.0 |
| **DPDK**                           | DPK v18.11                           |
| **Linux Container**                | Docker* v18.06.1                     |
| **Traffic Generator**              | DPDK Pktgen v3.5.9                   |
1.5 Release Package Contents

The directory tree of the Intel vCMTS Reference Dataplane Release package is shown below.

```
|-- vcmts
 |  |-- kubernetes
 |  |  |-- cmk
 |  |  |-- dashboard
 |  |  |-- docker
 |  |  |  |-- docker-image-cloud-init
 |  |  |  |-- docker-image-pktgen-init
 |  |  |  |-- docker-image-power-mgr
 |  |  |  `-- docker-image-sriov-dp
 |  |  |-- helm
 |  |  |  |-- pktgen
 |  |  |  |  |-- resources
 |  |  |  |  |-- templates
 |  |  |  |  `-- vcmtsd
 |  |  |  `-- resources
 |  |  |     `-- templates
 |  |  `-- install
 |  |     |-- config
 |  |     `-- services
 |`-- pktgen
 |  |-- docker
 |  |  `-- docker-image-pktgen
 |  |     `-- config
 |  `-- patches
|-- stats
 |  |-- collectd
 |     `-- docker
 |         |-- docker-image-collectd
 |         `-- docker-image-grafana
 |     `-- dashboards
|-- tools
 |  |-- setup
 |  |-- vcmtsd-config
 |     `-- vcmtsd-ctl
|-- traffic-profiles
 |`-- vcmtsd
 |  |-- cycle-capture
 |  |-- docker
 |     `-- docker-image-vcmtsd
 |     `-- config
 |     `-- patches
 |     `-- stats
```
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
- ReleaseNotes.txt: release notes relevant to the package

"vcmtsd" subdirectory:
Contains C source-code modules for the vCMTS dataplane application, and the following sub-directories:

- stats: contains C source-code modules for statistics collection
- cycle-capture: contains C source code modules for CPU cycle measurement
- docker: contains files to build the docker image for the vCMTS dataplane application
- patches: contains vCMTS dataplane patch file for DPDK scheduler library

"tools" subdirectory:
Contains tools to configure the vCMTS reference dataplane and traffic generation environments, in the following sub-directories:

- setup: contains scripts to configure the vCMTS dataplane and traffic generation environments
- vcmtsd-config: contains python source modules for a tool to configure vCMTS dataplane and traffic-generator platform environments settings and vCMTS service-group options.
- vcmtsd-ctl: contains python source modules for a tool to query vCMTS service-group configuration and to control traffic simulation

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

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

"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)
- patches: contains patches for DPDK Pktgen which are required for vCMTS traffic simulation
"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.

"kubernetes" subdirectory:
Contains files to build and install Kubernetes infrastructure required for the Intel(R) vCMTS reference dataplane system, in the following sub-directories:
  install : contains files required for Kubernetes installation
  cmk : contains files required for CMK (CPU Manager for Kubernetes)
  helm : contains files required for Helm, which is used for configuration of vCMTS dataplane and Pktgen Kubernetes POD’s
  dashboard : contains files required for Kubernetes WebUI dashboard
  docker : contains files to build docker images for Kubernetes infrastructure components
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 1  System BIOS settings – 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;CPU</td>
<td>Intel(R) Turbo Boost Technology</td>
<td>Disabled</td>
</tr>
<tr>
<td>P State Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU</td>
<td>Energy Efficient Turbo</td>
<td>Disabled</td>
</tr>
<tr>
<td>P State Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enhanced Intel SpeedStep Technology</td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>C State Control</td>
<td>C1E</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

2.1.2 Check Fortville NIC Firmware and Driver versions

Check the Fortville NIC i40e base driver and firmware versions by running `ethtool` as follows for a dataplane network interface.

e.g. for network interface `enp26s0f2`
The driver version should be 2.4.6 or later.
The firmware version should be 6.01 or later.

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

**Fortville NIC Driver**

**Fortville 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 % Mounted on
udev        11G    0   11G  0% /dev
tmpfs       9.4G  875M  8.5G 10% /run
/dev/adb1   440G 188G  229G 46% /
```
It is recommended to disable swap space as described below.

First check if swap space is enabled, by running the following command.

```bash
blkid
/dev/sda1: UUID="08070332-90b8-45a5-891d-e76167ee876d" TYPE="ext4" PARTUUID="ee39ea61-01"
/dev/sda5: UUID="34c0a658-225b-4b28-8ab9-8ac333c30b819" 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
```

```
total       used       free      shared  buff/cache   available
Mem:        62G        57G        239M       2.4M        4.7G        4.4G
Swap:        0B        0B 0B
```

### 2.1.4  Check OS and Linux kernel versions

The recommended OS distribution is Ubuntu 18.04 LTS, and recommended Linux kernel version is 4.15.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 18.04.1 LTS
Release: 18.04
Codename: bionic
```
The following type of output is expected:

```
4.15.0-39-generic
```

If a kernel version older than 4.15.x is displayed, the required kernel can be installed by running the following:

```
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";
```

**NOTE:** The recommended steps for manually upgrading to the latest versions of Linux kernel and Linux packages which are available for the LTS distro being used can be found in the appendices, section 4.5.2.

### 2.1.6 Configure Linux GRUB settings

The following Linux GRUB settings are required on the vCMTS dataplane server.
Table 2  Linux GRUB settings – vCMTS Dataplane server

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>default_hugepagesz=1G hugepagesz=1G hugepages=72</code></td>
<td>Huge-page memory size and number of pages reserved for DPDK applications</td>
</tr>
<tr>
<td><code>intel_pstate=disable</code></td>
<td>Disable hardware control of P-states</td>
</tr>
<tr>
<td><code>intel_iommu=on iommu=pt</code></td>
<td>Enable SR-IOV</td>
</tr>
<tr>
<td><code>isolcpus=2-19,22-39,42-59,62-79</code></td>
<td>Isolate vCMTS dataplane cores from the Linux kernel task scheduler</td>
</tr>
<tr>
<td><code>nr_cpus=80</code></td>
<td>Total number of logical cores on the system (aka hyperthreads)</td>
</tr>
</tbody>
</table>

**NOTE:** The sample `isolcpus` and `nr_cpus` settings shown here are for a 20-core dual-processor CPU package. If not using a 20-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 20-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_hugepagesz=1G hugepagesz=1G hugepages=72 intel_pstate=disable intel_iommu=on iommu=pt isolcpus=2-19,22-39,42-59,62-79 nr_cpus=80"
```

⚠️ 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 20-core dual-processor CPU package.

```
lscpu | grep "CPU(s):"
CPU(s):     80
NUMA node0 CPU(s): 0-19,40-59
NUMA node1 CPU(s): 20-39,60-79
```

Below is the output for a 22-core dual-processor CPU package.

In this case the `isolcpus` and `nr_cpus` settings would be: `isolcpus=2-21,24-43,46-65,68-87 nr_cpus=88`

```
lscpu | grep "CPU(s):"
CPU(s):     88
NUMA node0 CPU(s): 0-21,44-65
NUMA node1 CPU(s): 22-43,66-87
```
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-grub
geruot
```

### 2.1.7 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.

### 2.1.8 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-v18-10-2.tar.gz
tar -zxvf intel-vcmtsd-v18-10-2.tar.gz
cd vcmts
ls -lr
```

**NOTE:** 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)

```
export VCMTSD_HOST=y
export MYHOME=/opt
```
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 VCMTSD_HOST=y
export MYHOME=/opt
```

### 2.1.9 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/setup/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
```

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/";
```

### 2.1.10 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.
2.1.11 Configure Docker

The community edition of Docker should already have been installed in the previous section, Install Linux packages. 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.

```bash
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"
```

TCP port 2375 needs to be enabled to allow control of dockerized applications e.g. by the vcmtsd control tool (see section 2.2.12). Add TCP port 2375 to the ExecStart entry in the file `/lib/systemd/system/docker.service` as shown below.

```
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H fd://
```

⚠️ Please note: enabling the above tcp port should only be done in a secure environment, such as a private lab network.

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.

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

Now start Docker by running the following commands.

```bash
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.

```bash
systemctl restart docker
```

### 2.1.12 Install the vCMTS dataplane control tool

The following two tools are provided with the vCMTS reference dataplane release package.

- **vcmtsd-config**:
  - configure vCMTS dataplane platform environments settings
  - configure vCMTS traffic-generator platform environments settings
  - configure vCMTS service-group options

- **vcmtsd-ctl**:
  - arp the links between traffic-generator and vCMTS dataplane instances
  - control traffic simulation
  - query vCMTS service-group options

The vcmtsd configuration tool should only be run from the Kubernetes master (which also acts as the vCMTS traffic-generator server). See section 2.2.12.

However, the vcmtsd control tool may be run from the vCMTS dataplane server, and the following commands are required for installation (assuming vCMTS dataplane package installed in `/opt` directory).
The vCMTS dataplane control tool is now installed.
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 3  System BIOS settings – 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;CPU P State Control</td>
<td>Intel(R) Turbo Boost Technology</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Energy Efficient Turbo</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Enhanced Intel SpeedStep Technology</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

2.2.2 Check Fortville NIC Firmware and Driver versions

Check the Fortville NIC i40e base driver and firmware versions by running the following ethtool for a dataplane network interface.

e.g.

```
ethtool -l enp26s0f2
driver: i40e
version: 2.4.10
firmware-version: 6.01 0x80003484 1.1747.0
expansion-rom-version:
bus-info: 0000:1a:00.2
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```
The driver version should be 2.4.6 or later.
The firmware version should be 6.01 or later.

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

**Fortville NIC Driver**

**Fortville 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.

```bash
df -h
Filesystem Size Used Avail Use% 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.**

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

```bash
swapoff -a
```

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

```bash
free -h
```

<table>
<thead>
<tr>
<th>Mem:</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 4.15.x or later, and recommended OS distribution is Ubuntu 18.04 LTS.

Run the following command to check the OS distribution version:

```bash
lsb_release -a
```

```
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.1 LTS
Release:        18.04
Codename:       bionic
```

Run the following command to check the kernel version of the system:

```bash
uname -r
```

The following type of output is expected:

```
4.15.0-39-generic
```

If a kernel version older than 4.15.x is displayed, the required kernel can be installed by running the following:

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

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

**NOTE:** The recommended steps for manually upgrading to the latest versions of Linux kernel and Linux packages which are available for the LTS distro being used can be found in the appendices, section 4.5.1.

2.2.6 Configure Linux GRUB settings

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

**Table 4** Linux GRUB settings – vCMTS Traffic-generator server

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default_hugepagesz=1G hugepagesz=1G hugepagesize=72</td>
<td>Huge-page memory size and number of pages reserved for DPDK applications</td>
</tr>
<tr>
<td>intel_iommu=on iommu=pt</td>
<td>Enable SR-IOV</td>
</tr>
</tbody>
</table>
| isolcpus=2-19,22-39,42-59,62-79 | Isolate vCMTS dataplane cores from the Linux kernel task scheduler  
**NOTE:** based on Dual 20-core CPU |
| nr_cpus=80 | Total number of logical cores on the system (aka hyperthreads) |
NOTE: The sample `isolcpus` and `nr_cpus` settings shown here are for a **20-core dual-processor CPU package**. If not using a 20-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 **20-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_pstate=disable intel_iommu=on iommu=pt isolcpus=2-19,22-39,42-59,62-79 nr_cpus=80"
```

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 **20-core** dual-processor CPU package.

```
lscpu | grep "CPU(s):"
CPU(s): 80
NUMA node0 CPU(s): 0-19,40-59
NUMA node1 CPU(s): 20-39,60-79
```

Below is the output for a **22-core** dual-processor CPU package.

In this case the `isolcpus` and `nr_cpus` settings would be: `isolcpus=2-21,24-43,46-65,68-87 nr_cpus=88`

```
lscpu | grep "CPU(s):"
CPU(s): 88
NUMA node0 CPU(s): 0-21,44-65
NUMA node1 CPU(s): 22-43,66-87
```

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-grub
reboot
```
2.2.7 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.

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

Select default options at each prompt.

2.2.8 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.

```
cd /opt
wget https://01.org/sites/default/files/downloads/intel-vcmtsd-v18-10-2.tar.gz
tar -zxvf intel-vcmtsd-v18-10-2.tar.gz
cd vcmts
ls -lR
```

**NOTE:** 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.

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.

```
cd /opt/vcmts/traffic-profiles
```

**NOTE:** 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)
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`).

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

### 2.2.9 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/vcmts/tools/setup/proxy.sh` file (assuming installation in `/opt`).

The example entries below should be replaced with actual proxy address and port number and the actual hostname of the vCMTS traffic generator 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,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/";
```
2.2.10 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/setup/env.sh
install_base_ubuntu_pkgs
```

2.2.11 Configure Docker

The community edition of Docker should already have been installed in the previous section, Install Linux packages. 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"
```

TCP port 2375 needs to be enabled to allow control of dockerized applications e.g. by the vcmtsd control tool (see section 2.2.12). Add TCP port 2375 to the ExecStart entry in the file `/lib/systemd/system/docker.service` as shown below.

```
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H fd://
```

Please note: enabling the above tcp port should only be done in a secure environment, such as a private lab network.

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.

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

Now start Docker by running the following commands.

```bash
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.

```bash
systemctl restart docker
```

### 2.2.12 Install vCMTS dataplane configuration and control tools

The following two tools are provided with the vCMTS reference dataplane release package.

- **vcmtsd-config** :
  - configure vCMTS dataplane platform environments settings
  - configure vCMTS traffic-generator platform environments settings
  - configure vCMTS service-group options
- **vcmtsd-ctl** :
  - arp links between traffic-generator and vCMTS dataplane instances,
  - control traffic simulation
  - query vCMTS service-group options

Both of these tools should be run from the vCMTS traffic-generator server, which also acts as the Kubernetes master, and the following steps are required for installation.
First, install the Python Virtual environment, which is required to run these tools.

```
pip install virtualenv
```

Next, install the vcmtsd-config tool by running the following commands.

```
cd $MYHOME/vcmts/tools/vcmtsd-config
python -m virtualenv env
source env/bin/activate
pip install -e .
deactivate
```

Next install the vcmtsd-ctl tool by running the following commands.

```
cd $MYHOME/vcmts/tools/vcmtsd-ctl
python -m virtualenv env
source env/bin/activate
pip install -e .
deactivate
```

Both tools are now installed.

⚠️ Please note: the vcmtsd configuration tool should only be run from the Kubernetes master (vCMTS traffic-generator server).
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 server) 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 configuration 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.

NOTE: While the configuration tool is run on the vCMTS traffic-generation server (also Kubernetes master), 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 `vcmtsd-config` 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/vcmtsd-config
source env/bin/activate
vcmtsd-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.
Once completed, it can be verified that settings for the vCMTS dataplane platform runtime environment have been configured correctly in the files below on the vCMTS dataplane server:

- the fully qualified domain-name of the vCMTS dataplane server should be specified in the common platform configuration file:
  
  $MYHOME/vcmts/tools/setup/common-host-config.sh
  
  (see Appendix section 4.1.1 for example)

- the vCMTS dataplane server PCI address settings should be specified in the vCMTS dataplane host configuration file:
  
  $MYHOME/vcmts/tools/setup/vcmtsd-host-config.sh
  
  (see Appendix section 4.1.2 for example)

**NOTE:** The vcmtsd-host-config.sh file is not installed onto the Kubernetes master (also traffic-generator/pktgen server) at this point. This will happen later in the installation process when the "vcmtsd-config service-groups" command is run (see section 2.7.1).

### 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:** CMK (CPU Manager for Kubernetes) is not used on the traffic-generator platform so that core mappings to Pktgen application instances are assigned by the configuration tool in this case

Run the vcmtsd-config 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/vcmtsd-config
source env/bin/activate
vcmtsd-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.
Once completed, it can be verified that settings for the vCMTS traffic-generator platform runtime environment have been configured correctly in the files below on the vCMTS traffic-generator server:

- the vCMTS traffic-generator fully qualified domain name should be specified in the common configuration file:
  
  ```bash
  $MYHOME/vcmts/tools/setup/common-host-config.sh:
  ```

  (see Appendix section 4.1.1 for example)

- the vCMTS traffic-generator PCI address settings and core-mappings should be specified in the pktgen host configuration file:
  ```bash
  $MYHOME/vcmts/tools/setup/pktgen-host-config.sh
  ```

  (see Appendix section 4.1.3 for example)
2.4 Install Kubernetes

vCMTS dataplane and pktgen (traffic-generator) application instances are orchestrated using Kubernetes. 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 Master 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 master) and vCMTS dataplane node to perform these steps.

2.4.1 Install Kubernetes master and node on the vCMTS traffic-generator server

The vCMTS traffic-generator server acts as both Kubernetes master and node.

To install Kubernetes master and 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.

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

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

```
install_kubernetes_master
```

⚠️ Note that the Kubernetes installation takes a long time and should NOT be interrupted.

Once this is complete, Kubernetes Master and Node is installed and running on the vCMTS traffic-generator server.
2.4.2 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:

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

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

```
install_kubernetes_node
```

⚠️ Note that the Kubernetes installation takes a long time and should NOT be interrupted.

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

2.4.3 Verify Kubernetes installation

A successful Kubernetes installation may be verified by running the following command on the Kubernetes master (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 master.

It is important that the node name used for pktgen_node and vcmtspktgen_node is the name returned from the 'kubectl get nodes' command above.

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

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

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

check_kubernetes

• etcd.service - Etcd Server
  Loaded: loaded (/lib/systemd/system/etcd.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:02:11 IST; 2h 13min ago

• kube-apiserver.service - Kubernetes API Server
  Loaded: loaded (/lib/systemd/system/kube-apiserver.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:02:26 IST; 2h 13min ago

• kube-scheduler.service - Kubernetes Scheduler Plugin
  Loaded: loaded (/lib/systemd/system/kube-scheduler.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:02:36 IST; 2h 12min 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 Tue 2018-09-18 10:02:37 IST; 2h 12min 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 Tue 2018-09-18 10:02:41 IST; 2h 12min ago

• kubelet.service - Kubernetes Kubelet Server
  Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:02:41 IST; 2h 12min ago

• flanneld.service - Flanneld overlay address etcd agent
  Loaded: loaded (/lib/systemd/system/flanneld.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:02:41 IST; 2h 12min ago

• kube-proxy.service - Kubernetes Kube-Proxy Server
  Loaded: loaded (/lib/systemd/system/kube-proxy.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:02:41 IST; 2h 12min ago
On the Kubernetes vcmtsd node, output such as shown below is expected.

```bash
export VCMTSD_HOST=y
export MYHOME=/opt
source $MYHOME/vcmts/tools/setup/env.sh

check_kubernetes

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 Tue 2018-09-18 10:14:31 IST; 1h 46min ago

• kubelet.service - Kubernetes Kubelet Server
  Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:14:31 IST; 1h 46min ago

• flanneld.service - Flanneld overlay address etcd agent
  Loaded: loaded (/lib/systemd/system/flanneld.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:14:30 IST; 1h 46min ago

• kube-proxy.service - Kubernetes Kube-Proxy Server
  Loaded: loaded (/lib/systemd/system/kube-proxy.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2018-09-18 10:14:07 IST; 1h 47min ago
```

If a service has failed it is recommended to attempt to restart the service as follows:

```bash
systemctl status <service>

systemctl stop <service>

systemctl start <service>
```
2.5 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.

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

### 2.5.1 Install QAT drivers

Install QAT drivers on the vCMTS dataplane node host OS by running the following command and follow the prompt accepting default options.

```bash
cd $MYHOME
install_qat_drivers
```

Select option '2 Install Acceleration' and select defaults when prompted.

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

```bash
lspci | grep 37c8
```

```plaintext
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)
```

### 2.5.2 Install DPDK

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

```bash
cd $MYHOME
install_dpdk
```
2.5.3 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
```

**NOTE:** The version of CMK built here contains some updates to CMK v1.3.0, which are required for the Intel vCMTS reference dataplane platform.

The CMK docker image should have been built and tagged with v1.3.0.
To verify, run the following command to display docker images on the system.

```
docker images
```

This should display ‘cmk:v1.3.0’ for CMK.

2.5.4 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
build_docker_power_mgr
build_docker_sriov_dp
build_docker_qat
```

2.5.5 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.
NOTE: 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.

### 2.5.6 Build vCMTS dataplane application container

Two types of vCMTS dataplane docker images are created and tagged as follows
- `feat`: maximum statistics including cycle count captures
- `perf`: minimum statistics, with no cycle count captures

One of the two modes may be selected as a service-group option (see section 2.7.1 - Configure vCMTS dataplane service-group options).

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

```
build_docker_vcmtd_feat
```

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

```
build_docker_vcmtd_perf
```
2.6 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 master).

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

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

2.6.1 Install DPDK

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

```
cd $MYHOME
install_dpdk
```

2.6.2 Install Helm

Helm is used by the Kubernetes master 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 master managing all application containers on the system.

```
cd $MYHOME
wget https://storage.googleapis.com/kubernetes-helm/helm-v2.10.0-rc.1-linux-amd64.tar.gz
tar -zxvf helm-v2.10.0-rc.1-linux-amd64.tar.gz
cp linux-amd64/helm /usr/bin/.
```

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
```
2.6.3 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_pktgen_init
build_docker_sriov_dp
```

2.6.4 Build Pktgen application container

Run the following command to build the DPDK Pktgen docker image.

```
build_docker_pktgen
```
2.7 Configure vCMTS dataplane

Each vCMTS dataplane (Kubernetes) POD 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>Service-group mode</td>
<td>Service-group mode - one of two settings described below. Max-load: exclusive core for downstream, shared core for upstream (2 x SG’s) Low-speed: shared cores for downstream (2 x SG’s) and upstream (4 x SG’s) <strong>Default: Max-load</strong></td>
</tr>
<tr>
<td>Number of Service-Groups</td>
<td>Number of service-groups. <strong>Default: Max based on mode, and number of CPU cores and NW ports</strong></td>
</tr>
<tr>
<td>Number of Subscribers</td>
<td>100, 300, 500 or 1000 subscribers per service-group <strong>Default: 300</strong></td>
</tr>
<tr>
<td>Channel Configuration</td>
<td>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 <strong>Default: 4xOFDM</strong></td>
</tr>
<tr>
<td>Percentage of DES CM’s</td>
<td>0,5 or 10 percent of cable-modems using DES encryption <strong>NOTE: 5 and 10% only applicable for 1 x OFDM channel configuration</strong> <strong>Default: 0% DES</strong></td>
</tr>
<tr>
<td>Downstream CRC re-generation</td>
<td>Enable or Disable CRC re-generation for downstream DOCSIS frames <strong>Default: Enable CRC</strong></td>
</tr>
<tr>
<td>Downstream Crypto Offload</td>
<td>Enable or Disable QuickAssist offload for downstream encryption <strong>Default: Disable QAT</strong></td>
</tr>
<tr>
<td>Traffic type</td>
<td>Select iMix or fixed-sized packets for cable traffic. <strong>iMix1:</strong> Upstream 65% : 84B, 18% : 256B, 17% : 1280B Downstream 15% : 84B, 10% : 256B, 75% : 1280B</td>
</tr>
</tbody>
</table>
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></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 NIC’s 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 dataplane configuration tool provided in the vCMTS dataplane release package, as described in the next section.
2.7.1 Configure vCMTS dataplane service-group options

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

After logging in to the Kubernetes master, set the environment for running the configuration tool.

```bash
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtsd-config
source env/bin/activate
```

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

```bash
vcmtsd-config service-groups
   deactivate
```

Follow the prompts to select service-group settings and generate Helm charts for the Kubernetes cluster. 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 `vcmtsd` configuration tool, as described below.

For example, to configure 16 service-groups at max-load with 4OFDM channels, 300 subscribers, no DES 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:

```bash
vcmtsd-config service-groups -u -m max-load -p packed -n 16 -c 4ofdm -s 300 -d 0 -v enabled -q "8,9,10,11" -t imix2 deactivate -z "1"
```

A full description of `vcmtsd-config` command-line arguments for service-group configurations provided below.

### Table 8 Service-group configuration tool - commandline options

<table>
<thead>
<tr>
<th>param</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Service-group mode.</td>
</tr>
<tr>
<td>values: max-load</td>
<td>exclusive dataplane cores for downstream</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>num-service-groups</td>
<td>Number of service-groups.</td>
</tr>
<tr>
<td></td>
<td>Integer value.</td>
</tr>
<tr>
<td></td>
<td>NOTE: service-groups are evenly divided between CPU’s, number of service-groups per CPU is limited by the number of NIC PF’s and available isolated CPU cores.</td>
</tr>
<tr>
<td></td>
<td>default: maximum service-groups based on mode, number of NIC PF’s and number of isolated CPU cores</td>
</tr>
<tr>
<td>cpu-socket-mode</td>
<td>Order of service group assignment to CPU sockets</td>
</tr>
<tr>
<td></td>
<td>values:</td>
</tr>
<tr>
<td></td>
<td>packed: service groups assigned to first CPU socket until full, then assigned to second</td>
</tr>
<tr>
<td></td>
<td>spread: service groups assigned to alternating CPU sockets</td>
</tr>
<tr>
<td></td>
<td>default: packed</td>
</tr>
<tr>
<td>channel-config</td>
<td>Channel configuration</td>
</tr>
<tr>
<td></td>
<td>values:</td>
</tr>
<tr>
<td></td>
<td>1ofdm: 1 OFDM channels, 32 SQ-QAM</td>
</tr>
<tr>
<td></td>
<td>2ofdm: 2 OFDM channels, 32 SQ-QAM</td>
</tr>
<tr>
<td></td>
<td>4ofdm: 4 OFDM channels, no SQ-QAM</td>
</tr>
<tr>
<td></td>
<td>6ofdm: 6 OFDM channels, no SQ-QAM</td>
</tr>
<tr>
<td></td>
<td>default: 4ofdm</td>
</tr>
<tr>
<td>numsubs</td>
<td>Number of subscribers (cable-modems)</td>
</tr>
<tr>
<td></td>
<td>values: 100, 300, 500 or 1000</td>
</tr>
<tr>
<td></td>
<td>default: 300</td>
</tr>
<tr>
<td>percent-des-subscribers</td>
<td>Percentage of DES crypto subscribers (cable-modems)</td>
</tr>
<tr>
<td></td>
<td>NOTE: only applies for the 1 OFDM channel configuration</td>
</tr>
<tr>
<td></td>
<td>values: 0, 5, or 10</td>
</tr>
<tr>
<td></td>
<td>default: 0</td>
</tr>
<tr>
<td>crc</td>
<td>CRC enabled/disabled</td>
</tr>
<tr>
<td></td>
<td>values:</td>
</tr>
<tr>
<td></td>
<td>enabled</td>
</tr>
<tr>
<td></td>
<td>disabled</td>
</tr>
<tr>
<td></td>
<td>default: enabled</td>
</tr>
<tr>
<td>qat-sgs</td>
<td>Comma separated list of service-group ID integer values with QAT offload</td>
</tr>
<tr>
<td></td>
<td>NOTE: comma separated list must be enclosed in quotes.</td>
</tr>
<tr>
<td></td>
<td>default: none</td>
</tr>
<tr>
<td>feature-level-stats-sgs</td>
<td>Comma separated list of service-group ID integer values with feature-level stats</td>
</tr>
</tbody>
</table>
NOTE: comma separated list must be enclosed in quotes.
default: none

| latency-stats-sgs | Comma separated list of service-group ID integer values with latency stats enabled
| NOTE: comma separated list must be enclosed in quotes.
default: none |

| traffic-type | Traffic-type values: imix1, imix2, 64B, 256B, 512B, 640B, 768B, 1024B, 1280B or 1536B
default: imix2 |

Once service-group configuration has been completed, it can be verified that service-group settings have been correctly applied for the files below on the Kubernetes master (also the Pktgen server):

- Selected service-group settings should have been applied to the vCMTS dataplane helm chart in the yaml file below:
  ```bash
  $MYHOME/vcmts/kubernetes/helm/vcmts/values.yaml
  ```
  (See Appendix section 4.3.1 for an example)

- Selected service-group settings should have been applied to vCMTS pktgen helm chart in the yaml file below:
  ```bash
  $MYHOME/vcmts/kubernetes/helm/pktgen/values.yaml
  ```
  (See Appendix section 4.3.2 for an example)

- Power management settings for the vCMTS dataplane server should be specified in the Helm file below for vCMTS dataplane infrastructure:
  ```bash
  $MYHOME/vcmts/kubernetes/helm/vcmts-infra/values.yaml
  ```
  (see Appendix section 4.1 for example)

- The fully qualified domain-name of the vCMTS dataplane server and the vCMTS dataplane core-pool configuration based on the selected number of service-groups should have been applied to the CMK cluster initialization yaml file below:
  ```bash
  $MYHOME/vcmts/kubernetes/cmk/cmk-cluster-init-pod.yaml
  ```
  (see Appendix section 4.4 for an example)

The following platform environment configuration files on the Kubernetes master should also be updated with final settings after running the service-group configuration command.

- the fully qualified domain-name of the vCMTS dataplane and traffic-generator servers should be specified in the common platform configuration file:
  ```bash
  $MYHOME/vcmts/tools/setup/common-host-config.sh
  ```
  (see Appendix section 4.1.1 for example)

- The vCMTS dataplane server PCI address settings should be specified in the vCMTS dataplane host configuration file:
  ```bash
  $MYHOME/vcmts/tools/setup/vcmtsd-host-config.sh
  ```
  (see Appendix section 4.1.2 for example)
the vCMTS traffic-generator PCI address settings and core-mappings should be specified in the pktgen host configuration file:

$MYHOME/vcmts/tools/setup/pktgen-host-config.sh

(see Appendix section 4.1.3 for example)
2.7.2 Configure vCMTS dataplane power profiles

Power-profiles may be configured for automated power management of the vCMTS system.

A power profile may be configured for each vCMTS service-group POD which is sent to the Power-Manager Daemonset (a singleton POD) at POD startup, as shown in the architecture diagram below.

Figure 7 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.

Power profile settings for the vCMTS reference dataplane system may be configured in the file, $MYHOME/vcmts/kubernetes/helm/vcmtsd/resources/power_policy.cfg as shown below.

The default configuration shown below specifies no quiet hours but may be updated as required.

Power profiles are set at POD initialization by Kubernetes, so vCMTS POD's must be restarted to effect a change (see section 2.8).
Note that the power-profile file shown above controls the power profiles for all service-groups on the system. This is provided to illustrate the concept of automated power-management. In order to manage a separate power profile per service-group, some modification is required to the vCMTS reference dataplane configuration tool and Helm chart schema.

The Power-Consumption Grafana dashboard below illustrates the benefit of automated power management by reducing power consumption of the system during quiet periods of the day when traffic is low.
2.8 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 master (also the traffic-generator/pktgen server).

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

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

### 2.8.1 Start infrastructure and application instances

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

```
start_cmk
start_tiller
start_init
start_dns
start_dash
start_vcmtsd
start_pktgen
```

Verify that all components started correctly by running the following Kubernetes command.

```
kubectl get all -a
```

The Kubernetes log function can be used to diagnose vcmtsd application issues. 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
```
If any vcmtsd or pktgen instances failed to start, this can be resolved by re-starting a specific vcmtsd or pktgen POD by running the command below.

```
restart_vcmtsd_pod <pod-number>
restart_pktgen_pod <pod-number>
```

### 2.8.2 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.

```
stop_pktgen
stop_vcmtsd
stop_dash
stop_dns
stop_init
stop_tiller
stop_cmk
```

### 2.8.3 How to restart the statistics daemonset
Note that to restart the statistics daemonset, possibly for some configuration change to collectd or InfluxDB, all vCMTS dataplane and pktgen instances must be restarted as shown below.

```
stop_pktgen
stop_vcmtsd
restart_stats_daemonset
start_vcmtsd
start_pktgen
```
2.9 System verification

The following sections describe how to verify that the system is running correctly.

2.9.1 Check Kubernetes dashboard

Kubernetes infrastructure and application POD status may be checked on the Kubernetes WebUI dashboard.

Firstly, set the vcmts reference dataplane environment by running the following command:

```
source $MYHOME/vcmts/tools/setup/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:** If accessing from a web-browser on a remote client an SSH tunnel to the WebUI dashboard service port on the Kubernetes master may be required.

Once signed in, the Kubernetes WebUI dashboard should be displayed as shown below.

All vcmtsd and pktgen pods should be displayed as running.
2.9.2 Query service-group options

The vCMTS control 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 POD’s on the system.

These steps should be performed as root user on the vCMTS dataplane server.

After logging in to the vCMTS dataplane server, set the environment for running the control tool.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtsd-ctl
source env/bin/activate
```

Then, run the `vcmtsd-ctl` tool provided in the vCMTS dataplane release package as follows to query configuration settings for all service-group.
Configuration settings for each vCMTS service-group should be displayed.

**NOTE:** It may take up to 1 minute to gather information for all service-groups.

### 2.9.3 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 `vcmtsd-ctl` tool provided in the vCMTS dataplane release package.

These steps should be performed as root user on the **Kubernetes master** (also the traffic-generator/pktgen server).

After logging in to the Kubernetes master, set the environment for running the vcmtsd control tool.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtsd-ctl
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.

```
vcmtsd-ctl pktgen-rate -i 0-15 us -r 2
vcmtsd-ctl pktgen-rate -i 0-15 ds -r 20
vcmtsd-ctl pktgen-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.

```
vcmtsd-ctl pktgen-rate -i 0-15 us -r 1
vcmtsd-ctl pktgen-rate -i 0-15 ds -r 10
```
To stop traffic, run the following command.

```
vcmtsd-ctl pktgen-stop -i 0-15 usds
```

**NOTE:** If using 10G NIC’s the pktgen-rate values above are percentages of 10G line-rate instead of 25G

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

```
vcmtsd-ctl measure -i 0 ds -m rx
```

### 2.9.4 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/setup/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.

**NOTE:** 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.
Once signed in, select the required vCMTS dataplane dashboard by clicking on Dashboards Home. e.g. the System Summary dashboard is shown below.
2.9.4.1 How to modify Grafana platform metrics dashboard in case of non-default CPU core count

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.

Then, update as shown below for each individual graph to ensure that a query is present for each logical core (i.e. hyperthread) on the system.
Once the required updates have been made, save the dashboard by clicking the "Save" icon and selecting "Save JSON to file".
2.10 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 master (also the traffic-generator/pktgen server), after logging in as root user:

```
source $MYHOME/vcmts/tools/setup/env.sh
start_cmk
start_tiller
start_init
start_dns
start_dash
start_vcmtsd
start_pktgen
```

Verify that all components started correctly by running the following Kubernetes command:

```
kubectl get all -a
```

The Kubernetes log function can be used to diagnose vcmtsd application issues, 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
```

Perform system verification as described in previous section.
3 Upgrade Procedure

This section describes the steps required to upgrade to vCMTS reference dataplane v18.10.2, if the previous versions v18.10.0 or v18.10.1 have already been installed.

⚠️ It is assumed that vCMTS reference dataplane v18.10.0 or v18.10.1 is already installed.

3.1 vCMTS traffic-generator server preparation

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

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

It is assumed that the MYHOME and PKTGEN_HOST environment variable settings have been added to the root bashrc file ~/.bashrc during the original installation of the vCMTS reference dataplane.

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

3.1.1 Update Linux GRUB settings

Firstly, the isolcpus setting must be updated in the Linux kernel GRUB file /etc/default/grub to isolate only the first 2 cores on each CPU. Previously 3 cores were isolated.

An example of the updated isolcpus setting is shown below for a 20-core dual-processor CPU package.

```
GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=72 intel_pstate=disable
intel_iommu-on iommu-pt isolcpus=2-19,22-39,42-59,62-79 nr_cpus=80"
```

⚠️ 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 20-core dual-processor CPU package.
Below is the output for a 22-core dual-processor CPU package.
In this case the isolcpus and nr_cpus settings would be: isolcpus=2-21,24-43,46-65,68-87 nr_cpus=88

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-grub
reboot
```

After reboot, continue with the next section of the upgrade procedure.

### 3.1.2 Stop all Kubernetes-orchestrated components
After the reboot for the Linux kernel GRUB settings update, all Kubernetes-orchestrated components must be stopped before they can be upgraded. To do this execute the following commands.

```
source $MYHOME/vcmts/tools/setup/env.sh
stop_pktgen
stop_vcmtsd
stop_dash
stop_dns
stop_init
stop_tiller
stop_cmk
```

### 3.1.3 Backup current installation on vCMTS traffic-generation server
Back up the current installation on the vCMTS traffic-generation server in case need to revert to this version.
3.1.4 Load the new vCMTS reference dataplane package

Download the new vCMTS reference dataplane package from 01.org to the vCMTS traffic-generation server from 01.org and extract into the root directory of the installation.

```bash
mv $MYHOME/vcmts $MYHOME/vcmts-v18.10.0
cd $MYHOME
wget https://01.org/sites/default/files/downloads/intel-vcmtsd-v18-10-2.tar.gz
tar -zxvf intel-vcmtsd-18-10-2.tar.gz
cd vcmts
ls -lR
```

All files from the vCMTS reference dataplane package files as described in section 1.5, "Release Package Contents" should have been extracted to the $MYHOME/vcmts 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 $MYHOME/vcmts/traffic-profiles
```

3.1.5 Restore proxy server settings

vCMTS traffic-generation server proxy settings must be restored from the original installation.

```bash
cp -f $MYHOME/vcmts-v18.10.0/tools/setup/proxy.sh $MYHOME/vcmts/tools/setup/proxy.sh
```

It is assumed that all other proxy settings for apt and docker have been correctly completed in the original installation.

3.1.6 Upgrade vCMTS dataplane configuration and control tools

The following two tools are provided with the vCMTS reference dataplane release package.

- `vcmtsd-config`:
  - configure vCMTS dataplane platform environments settings
  - configure vCMTS traffic-generator platform environments settings
  - configure vCMTS service-group options
- `vcmtsd-ctl`:
- arp links between traffic-generator and vCMTS dataplane instances,
- control traffic simulation
- query vCMTS service-group options

Both of these tools should be run from the vCMTS traffic-generator server, which also acts as the Kubernetes master, and the following steps are required for re-installation of latest versions of these tools:

Install the latest version of the vcmtsd-config tool by running the following commands. It is assumed that the Python Virtual environment has already been installed in the original installation.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtsd-config
python -m virtualenv env
source env/bin/activate
pip install -e .
deactivate
```

Next install the latest version of that vcmtsd-ctl tool by running the following commands.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtsd-ctl
python -m virtualenv env
source env/bin/activate
pip install -e .
deactivate
```

Both tools are now installed.

⚠️ Please note: the vcmtsd configuration tool should only be run from the Kubernetes master (vCMTS traffic-generator server).

### 3.2 vCMTS dataplane server preparation

A number of steps are required to prepare the vCMTS dataplane server for software upgrade. Please note: you must log in as root user to the vCMTS dataplane server to perform these steps.
It is assumed that the MYHOME and VCMTSD_HOST environment variable settings have been added to the root bashrc file ~/.bashrc during the original installation of the vCMTS reference dataplane.

```
export VCMTSD_HOST=y
export MYHOME=/opt
```

### 3.2.1 Update Linux GRUB settings

Firstly, the isolcpus setting must be updated in the Linux kernel GRUB file `/etc/default/grub` to isolate only the first 2 cores on each CPU. Previously 3 cores were isolated.

An example of the updated `isolcpus` setting is shown below for 20-core dual-processor CPU.

```
GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=72 intel_pstate=disable
intel_iommu=on iommu=pt isolcpus=2-19,22-39,42-59,62-79 nr_cpus=88"
```

⚠️ 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 20-core dual-processor CPU package.

```
lscpu | grep "CPU(s):"
CPU(s): 80
NUMA node0 CPU(s): 0-19,40-59
NUMA node1 CPU(s): 20-39,60-79
```

Below is the output for a 22-core dual-processor CPU package.

In this case the `isolcpus` and `nr_cpus` settings would be: `isolcpus=2-21,24-43,46-65,68-87 nr_cpus=88`

```
lscpu | grep "CPU(s):"
CPU(s): 88
NUMA node0 CPU(s): 0-21,44-65
NUMA node1 CPU(s): 22-43,66-87
```

Once editing of the 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-grub
reboot
```

After reboot, continue with the next section of the upgrade procedure.
3.2.2 Backup current installation on vCMTS dataplane server

After the reboot for the Linux kernel GRUB settings update, back up the current installation on the vCMTS dataplane server in case need to revert to this version.

```
mv $MYHOME/vcmts $MYHOME/vcmts-v18.10.0
```

3.2.3 Load the new vCMTS reference dataplane package

Download the new vCMTS reference dataplane package from 01.org to the vCMTS dataplane server from 01.org and extract into the root directory of the installation.

```
cd $MYHOME
wget https://01.org/sites/default/files/downloads/intel-vcmtsd-v18-10-2.tar.gz
tar -zxvf intel-vcmtsd-18-10-2.tar.gz
cd vcmts
ls -lR
```

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

3.2.4 Restore proxy server settings

vCMTS dataplane server proxy settings must be restored from the original installation.

```
cp -f $MYHOME/vcmts-v18.10.0/tools/setup/proxy.sh $MYHOME/vcmts/tools/setup/proxy.sh
```

It is assumed that all other proxy settings for apt and docker have been correctly completed in the original installation.

3.2.5 Upgrade vCMTS dataplane control tool

The vcmtsd control tool may also be run from the vCMTS dataplane server, and the following commands are required for installation of the latest version of this tool.

It is assumed that the Python Virtual environment has already been installed in the original installation.
source $MYHOME/vcmts/tools/setup/env.sh

cd $MYHOME/vcmts/tools/vcmtsd-ctl

python -m virtualenv env

source env/bin/activate

pip install -e .

deactivate

The vCMTS dataplane control tool is now installed.
3.3 Re-configure vCMTS platform settings

Platform configurations must be re-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 server) 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 configuration tool provided in the vCMTS reference dataplane release package.

3.3.1 vCMTS dataplane platform configuration

Firstly the vCMTS dataplane platform environment configuration is re-generated.

**NOTE:** While the configuration tool is run on the vCMTS traffic-generation server (also Kubernetes master), 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 `vcmtsd-config` tool provided in the vCMTS dataplane release package as follows to specify the settings for the vCMTS dataplane platform configuration.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtsd-config
source env/bin/activate
vcmtsd-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.
Upgrade Procedure

Once completed, it can be verified that settings for the vCMTS dataplane platform runtime environment have been configured correctly in the files below on the vCMTS dataplane server:

- the fully qualified domain-name of the vCMTS dataplane server should be specified in the common platform configuration file:
  \$MYHOME/vcmts/tools/setup/common-host-config.sh
  (see Appendix section 4.1.1 for example)

- the vCMTS dataplane server PCI address settings should be specified in the vCMTS dataplane host configuration file:
  \$MYHOME/vcmts/tools/setup/vcmtsd-host-config.sh
  (see Appendix section 4.1.2 for example)

**NOTE:** The vcmtsd-host-config.sh file is not installed onto the Kubernetes master (also traffic-generator/pktgen server) at this point. This will happen later in the installation process when the "vcmtsd-config service-groups" command is run (see section 2.7.1).

### 3.3.2 vCMTS traffic-generator platform configuration

Next the vCMTS traffic-generator platform environment configuration is re-generated.

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

Run the vcmtsd-config tool provided in the vCMTS dataplane release package as follows to specify the settings for the vCMTS traffic-generator (Pktgen) platform configuration.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtsd-config
source env/bin/activate
vcmtsd-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.
Once completed, it can be verified that settings for the vCMTS traffic-generator platform runtime environment have been configured correctly in the files below on the vCMTS traffic-generator server:

- the vCMTS traffic-generator fully qualified domain name should be specified in the common configuration file:
  ```bash
  $MYHOME/vcmts/tools/setup/common-host-config.sh
  ```
  (see Appendix section 4.1.1 for example)

- the vCMTS traffic-generator PCI address settings and core-mappings should be specified in the pktgen host configuration file:
  ```bash
  $MYHOME/vcmts/tools/setup/pktgen-host-config.sh
  ```
  (see Appendix section 4.1.3 for example)
3.4 Apply security updates for Kubernetes

Some updates for improved security of Kubernetes must be applied on both the vCMTS traffic-generator and vCMTS dataplane servers.

3.4.1 Kubernetes updates on vCMTS traffic-generation server

These steps should be performed after logging in as root user to the vCMTS traffic-generator server (which also acts as the Kubernetes master).

The etcd version used with Kubernetes must be upgraded to protect against the use of insecure SSL ciphers. This can be done by executing the following command on the vCMTS traffic-generator server.

```
source $MYHOME/vcmts/tools/setup/env.sh
stop_kubernetes
upgrade_etcd
```

Kubernetes apiserver and kubelet also allow for the use of insecure SSL ciphers by default. By executing the following commands both apiserver and kubelet can be re-configured to disable use of insecure SSL ciphers.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts
cp kubernetes/install/config/apiserver /etc/kubernetes/apiserver
cp kubernetes/install/config/kubelet /etc/kubernetes/kubelet
sed -i "s/K8S_MASTER/$KUBERNETES_MASTER_IP/" /etc/kubernetes/apiserver
sed -i "s/K8S_NODE/$KUBERNETES_MASTER_IP/" /etc/kubernetes/kubelet
upgrade_kubernetes
start_kubernetes
```

3.4.2 Kubernetes updates on vCMTS dataplane server

These steps should be performed after logging in as root user to the vCMTS dataplane server.

Kubelet is the only Kubernetes component that needs to be reconfigured on the vCMTS dataplane server to prevent the use of insecure SSL ciphers. Run the following commands to reconfigure kubelet.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts
cp kubernetes/install/config/apiserver /etc/kubernetes/apiserver
cp kubernetes/install/config/kubelet /etc/kubernetes/kubelet
sed -i "s/K8S_MASTER/$KUBERNETES_MASTER_IP/" /etc/kubernetes/apiserver
sed -i "s/K8S_NODE/$KUBERNETES_MASTER_IP/" /etc/kubernetes/kubelet
upgrade_kubernetes
start_kubernetes
```
source $MYHOME/vcmts/tools/setup/env.sh

cd $MYHOME/vcmts

stop_kubernetes

cp kubernetes/install/config/kubelet /etc/kubernetes/kubelet

sed -i "s/K8S_NODE/$KUBERNETES_NODE_IP/" /etc/kubernetes/kubelet

upgrade_kubernetes

start_kubernetes
3.5 Upgrade vCMTS traffic-generator node software

The following sections describe the upgrading 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 master).

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

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

### 3.5.1 Upgrade DPDK

Upgrade DPDK to the latest version on the vCMTS traffic generator node host OS by running the following commands.

Back up previous version of DPDK.

```
mv $MYHOME/dpdk $MYHOME/dpdk-v18.08
```

```
cd $MYHOME
install_dpdk
```

### 3.5.2 Re-build Kubernetes infrastructure components on the traffic-generator server

Run the following commands to re-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_pktgen_init --no-cache
build_docker_sriov_dp --no-cache
```
3.5.3 Re-build Pktgen application container

Run the following command to re-build the DPDK Pktgen docker image.

```
build_docker_pktgen --no-cache
```
3.6 Upgrade vCMTS dataplane node software

The following sections describe the upgrading 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/setup/env.sh
```

3.6.1 Upgrade DPDK

Upgrade DPDK to the latest version on the vCMTS dataplane node host OS by running the following commands.

Back up previous version of DPDK.

```
mv $MYHOME/dpdk $MYHOME/dpdk-v18.08
```

Install latest version of DPDK.

```
cd $MYHOME
install_dpdk
```
3.6.2 Re-build Kubernetes infrastructure components on the vCMTS dataplane server

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

Firstly, run the following commands to rebuild CMK (CPU Manager for Kubernetes).

```
cd $MYHOME
build_docker_cmk
```

**NOTE:** The version of CMK built here contains some updates to CMK v1.3.0, which are required for the Intel vCMTS reference dataplane platform.

Next, run the following commands to rebuild the remaining Kubernetes infrastructure components.

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

3.6.3 Re-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 re-build docker container images for the telemetry components listed above.

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

**NOTE:** 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.

### 3.6.4 Re-build vCMTS dataplane application container

Two types of vCMTS dataplane docker images are created and tagged as follows:

- `feat`: maximum statistics including cycle count captures
- `perf`: minimum statistics, with no cycle count captures

One of the two modes may be selected as a service-group option (see section 0- 3.7 Re-configure vCMTS dataplane service-group options).

Re-build the vCMTS dataplane feat docker image by running the following command.

```
build_docker_vcmtd_feat --no-cache
```

Re-build the vCMTS dataplane perf docker image by running the following command.

```
build_docker_vcmtd_perf
```

### 3.7 Re-configure vCMTS dataplane service-group options

Follow the steps below to re-configure service-group options for vCMTS dataplane POD's on the system.

These steps should be performed as root user on the Kubernetes master (also the traffic-generator/pktgen server).

After logging in to the Kubernetes master, set the environment for running the configuration tool.

```
source $MYHOME/vcmts/tools/setup/env.sh
cd $MYHOME/vcmts/tools/vcmtd-config
source env/bin/activate
```

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

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

### 3.8 Run vCMTS dataplane and traffic-generation software

To run vCMTS reference dataplane and traffic-generation software, please refer to section 2.8 of the installation procedure.

To verify that the system is running correctly, please refer to section 2.9 of the installation procedure.
4 Appendix

4.1 Sample platform configurations

See sample platform configuration files below for vCMTS dataplane and traffic-generator nodes.

4.1.1 vCMTS common platform configuration

A sample version of the vCMTS common platform configuration file, $MYHOME/vcmts/tools/setup/common-host-config.sh as generated by the vcmtsd configuration tool is shown below.

This file contains common settings required on both the vCMTS dataplane and traffic-generator nodes.

```bash
#!/usr/bin/env bash

# COMMON HOST CONFIGURATION

export VCMTSD_NODE_HOSTNAME="myvcmtsd-node.example.com"
export PKTGEN_NODE_HOSTNAME="mypktgen-node.example.com"

export VCMTS_ENVIRONMENT="kubernetes"
export KUBERNETES_MASTER_IP=$PKTGEN_NODE_HOSTNAME
export KUBERNETES_NODE_IP=$VCMTSD_NODE_HOSTNAME

# Root directory where packages are installed
export MYHOME="/opt"

export VCMTS_ROOT="$MYHOME/vcmts"
export KUBERNETES_ROOT="$VCMTS_ROOT/kubernetes"
export DPDK_ROOT_RELEASE="$MYHOME/dpdk"
export AESNI_MULTI_BUFFER_LIB_PATH="$MYHOME/intel-ipsec-mb"
export HUGEPAGES_ROOT="/dev/hugepages"
```
4.1.2 vCMTS dataplane platform configuration

A sample version of the vCMTS dataplane platform configuration file, 
$MYHOME/vcmts/tools/setup/vcmtsd-host-config.sh as generated by the vcmtsd configuration 
tool is shown below for a platform with 4 x Dual port 25G NIC's and a single Intel QuickAssist Technology PCIe card.

#!/usr/bin/env bash

###############################################################################
# VCMTS HOST CONFIGURATION
###############################################################################

export DOCKERFILE_VCMTSROOT=$VCMTS_ROOT/vcmtsd/docker/docker-image-vcmtsd
export LD_LIBRARY_PATH=$COLLECTD_ROOT/lib

# Number of NW VF’s per PF, 16 SG’s -> 4
um_vf_per_pf=4
export num_qat_vf_per_pf=16
export max_sg_per_cpu_socket=8

# NIC PFs and VFs
# 2 x NIC’s per CPU socket, 2 x PF’s per NIC, 4 x VF’s per PF
declare -a ports_pf=('18:00.0' '18:00.1' '1a:00.0' '1a:00.1' '86:00.0' '86:00.1' 'b7:00.0' 'b7:00.1')
declare -A ports_vf=('18:00.0'='18:02.0 18:02.1 18:02.2 18:02.3 18:02.4 18:02.5 18:02.6 18:02.7'
                    '18:00.1'='18:0a.0 18:0a.1 18:0a.2 18:0a.3 18:0a.4 18:0a.5 18:0a.6 18:0a.7'
                    '1a:00.0'='1a:02.0 1a:02.1 1a:02.2 1a:02.3 1a:02.4 1a:02.5 1a:02.6 1a:02.7'
                    '1a:00.1'='1a:0a.0 1a:0a.1 1a:0a.2 1a:0a.3 1a:0a.4 1a:0a.5 1a:0a.6 1a:0a.7'
                    '86:00.0'='86:02.0 86:02.1 86:02.2 86:02.3 86:02.4 86:02.5 86:02.6 86:02.7'
                    '86:00.1'='86:0a.0 86:0a.1 86:0a.2 86:0a.3 86:0a.4 86:0a.5 86:0a.6 86:0a.7'
                    'b7:00.0'='b7:02.0 b7:02.1 b7:02.2 b7:02.3 b7:02.4 b7:02.5 b7:02.6 b7:02.7'
                    'b7:00.1'='b7:0a.0 b7:0a.1 b7:0a.2 b7:0a.3 b7:0a.4 b7:0a.5 b7:0a.6 b7:0a.7'

# QAT PFs and VFs
# 1 x QAT card, 3 x PF’s per QAT card, NOTE: VF’s managed by QAT device plugin
declare -a qat_ports_pf=('8a:00.0' '8c:00.0' '8e:00.0' 'b1:00.0' 'b3:00.0' 'b5:00.0')
declare -A qat_ports_vf=('8a:00.0'='8a:01.0 8a:01.1 8a:01.2 8a:01.3 8a:01.4 8a:01.5 8a:01.6 8a:01.7 8a:02.0 8a:02.1 8a:02.2 8a:02.3 8a:02.4 8a:02.5 8a:02.6 8a:02.7'
                    '8c:00.0'='8c:01.0 8c:01.1 8c:01.2 8c:01.3 8c:01.4 8c:01.5 8c:01.6 8c:01.7 8c:02.0 8c:02.1 8c:02.2 8c:02.3 8c:02.4 8c:02.5 8c:02.6 8c:02.7'
                    '8e:00.0'='8e:01.0 8e:01.1 8e:01.2 8e:01.3 8e:01.4 8e:01.5 8e:01.6 8e:01.7 8e:02.0 8e:02.1 8e:02.2 8e:02.3 8e:02.4 8e:02.5 8e:02.6 8e:02.7'
                    'b1:00.0'='b1:01.0 b1:01.1 b1:01.2 b1:01.3 b1:01.4 b1:01.5 b1:01.6 b1:01.7 b1:02.0 b1:02.1 b1:02.2 b1:02.3 b1:02.4 b1:02.5 b1:02.6 b1:02.7'
                    'b3:00.0'='b3:01.0 b3:01.1 b3:01.2 b3:01.3 b3:01.4 b3:01.5 b3:01.6 b3:01.7 b3:02.0 b3:02.1 b3:02.2 b3:02.3 b3:02.4 b3:02.5 b3:02.6 b3:02.7
Appendix

{"b5:00.0"}="b5:01.0 b5:01.1 b5:01.2 b5:01.3 b5:01.4 b5:01.5 b5:01.6 b5:01.7 b5:02.0 b5:02.1 b5:02.2 b5:02.3 b5:02.4 b5:02.5 b5:02.6 b5:02.7"
}
4.1.3 vCMTS traffic-generator platform configuration

A sample version of the vCMTS traffic-generator platform configuration file, 
$MYHOME/vcmts/tools/setup/pktgen-host-config.sh as generated by the vcmtsd configuration tool is shown below for a platform with 4 x Dual port 25G NIC’s.

**NOTE:** A core mapping is required in the traffic-generator platform environment file as it does not use CMK.

```bash
#!/usr/bin/env bash

# Core list to use for pktgen
export PKTGEN_CORE_LIST="0, 1, 18, 19, 36, 37, 54, 55, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71"

# Number of VF to create and use per PF
export num_vf_per_pf=4
export max_sg_per_cpu_socket=8

# PFs and VFs to instantiate
declare -a ports_pf=("05:00.0" "05:00.1" "07:00.0" "07:00.1" "81:00.0" "81:00.1" "8a:00.0" "8a:00.1")

declare -A ports_vf= 
("05:00.0")="05:02.0 05:02.1 05:02.2 05:02.3 05:02.4 05:02.5 05:02.6 05:02.7"
("05:00.1")="05:0a.0 05:0a.1 05:0a.2 05:0a.3 05:0a.4 05:0a.5 05:0a.6 05:0a.7"
("07:00.0")="07:02.0 07:02.1 07:02.2 07:02.3 07:02.4 07:02.5 07:02.6 07:02.7"
("07:00.1")="07:0a.0 07:0a.1 07:0a.2 07:0a.3 07:0a.4 07:0a.5 07:0a.6 07:0a.7"
("81:00.0")="81:02.0 81:02.1 81:02.2 81:02.3 81:02.4 81:02.5 81:02.6 81:02.7"
("81:00.1")="81:0a.0 81:0a.1 81:0a.2 81:0a.3 81:0a.4 81:0a.5 81:0a.6 81:0a.7"
("8a:00.0")="8a:02.0 8a:02.1 8a:02.2 8a:02.3 8a:02.4 8a:02.5 8a:02.6 8a:02.7"
("8a:00.1")="8a:0a.0 8a:0a.1 8a:0a.2 8a:0a.3 8a:0a.4 8a:0a.5 8a:0a.6 8a:0a.7"

# Mapping of Pktgen ports to cores
declare -A pktgen_port_to_core_map= 
("0 ct")="1" # HT0 of core 1 used for Pktgen command shell and timers for CPU socket 0
("0 crx")="41" # HT1 of core 1 used for traffic Rx for CPU socket 0
("1 ct")="21" # HT0 of core 19 used for Pktgen command shell and timers for CPU socket 1
("1 crx")="61" # HT1 of core 19 used for traffic Rx for CPU socket 1

("0 us")="3" # Pktgen port for SG 0 US uses core 3 (CPU socket 0)
("0 ds")="4" # Pktgen port for SG 0 DS uses core 4
("1 us")="5" # Pktgen port for SG 1 US uses core 5
("1 ds")="6" # Pktgen port for SG 1 DS uses core 6
("2 us")="7" # etc.
("2 ds")="8"
```
4.2 Sample Kubernetes infrastructure configuration file

A sample version of the Kubernetes vCMTS infrastructure configuration file, 
$MYHOME/vcmts/kubernetes/helm/vcmts-infra/values.yaml is shown below.

The configuration tool sets the power_mgmt setting based on the option selected for the vCMTS dataplane platform specification.

```yaml
vars:
  collectd_img: vcmts-collectd:v18.10.2
  influxdb_img: influxdb:1.6
  grafana_img: vcmts-grafana:v18.10.2
  cloud_init_img: vcmts-cloud-init:v18.10.2
  cloud_pktgen_init_img: vcmts-cloud-pktgen-init:v18.10.2
  power_mgr_img: vcmts-power-mgr:v18.10.2
  sriov_dev_plugin_img: sriov-device-plugin:v18.10.2
  qat_dev_plugin_img: qat-device-plugin:v18.10.2
  # power_mgmt: pm_off | pm_on
  power_mgmt: pm_on
```
4.3 Sample Helm chart for vCMTS POD's

See sample Helm yaml files below for vCMTS dataplane and traffic-generator nodes.

4.3.1 vCMTS dataplane helm chart for 16 service-groups

A sample version of the vCMTS dataplane Helm chart file, $MYHOME/vcmts/kubernetes/helm/vcmtsd/values.yaml as generated by the vcmtsd configuration tool is shown below for a 16 service-group configuration with 25G Dual port NIC's.

For the vcmtsd dataplane helm chart there is an entry per vcmtsd service-group. See service-group options at the top of the file.

```
serviceAccount: cmk-serviceaccount
topology:
  vcmts_replicas: 16
  vcmts_pods:
    # service_group_config options
    # num_ofdm: 1|2|4|6
    # num_subs: 100|300|500|1000
    # cm_crypto: aes|5pc-des|10pc-des
    # crc: crc_on|crc_off
    # qat: qat_off|qat_on
    # qat_dev: <pci-bus-number> - QAT device bus number if QAT enabled
    # power_mgmt: pm_off|pm_on
    # cpu_socket_id: 0|1
    # ds_core_type: exclusive|shared
    # ds_core_pool_index: integer value - nth core in pool in case of ds core pool type set to shared
    # vcmtsd_image: vcmts-d:feat|perf - feat:max stats, perf:min stats
    # latency: lat_on|lat_off - enable/disable latency stats collection

- service_group_config:
  sg_id: 0
  num_ofdm: 4
  num_subs: 300
  cm_crypto: aes
  crc: crc_on
  qat: qat_off
  power_mgmt: pm_on
  cpu_socket_id: 0
  ds_core_type: exclusive
  ds_core_pool_index: 0
  vcmtsd_image: vcmts-d:feat
  latency: lat_off

- service_group_config:
  sg_id: 1
  num_ofdm: 4
  num_subs: 300
```
cm_crypto: aes
crc: crc_on
qat: qat_off
power_mgmt: pm_on
cpu_socket_id: 0
ds_core_type: exclusive
ds_core_pool_index: 0
vcmtsd_image: vcmts-d:perf
latency: lat_off

<entries for sg_id 2 to 13>

- service_group_config:
  sg_id: 14
  num_ofdm: 4
  num_subs: 300
  cm_crypto: aes
crc: crc_on
qat: qat_off
power_mgmt: pm_on
cpu_socket_id: 1
ds_core_type: exclusive
ds_core_pool_index: 0
vcmtsd_image: vcmts-d:feat
latency: lat_off

- service_group_config:
  sg_id: 15
  num_ofdm: 4
  num_subs: 300
  cm_crypto: aes
crc: crc_on
qat: qat_off
power_mgmt: pm_on
cpu_socket_id: 1
ds_core_type: exclusive
ds_core_pool_index: 0
vcmtsd_image: vcmts-d:perf
latency: lat_off
4.3.2 vCMTS pktgen helm chart for 16 service-groups

A sample version of the vCMTS Pktgen Helm chart file, $MYHOME/vcmts/kubernetes/helm/pktgen/values.yaml as generated by the vcmtsd configuration tool is shown below for a 16 service-group configuration with 25G Dual port NIC’s.

In the pktgen helm chart, there is an entry per pktgen port. See port options at the top of the file.

```yaml
images:
vcmts_pktgen: vcmts-pktgen:v18.10.2
serviceAccount: cmk-serviceaccount
topology:
  pktgen_replicas: 16
  pktgen_pods:
  - pktgen_id: 0
    cpu_socket_id: 0|1
    cpu_socket_id: 0
    ports:
    # port options
    # traffic_type: imix1|imix2|64b|256b|512b|640b|768b|1024b|1280b|1536b
    # num_ofdm: 1|2|4|6
    # num_subs: 100|300|500|1000
    - port_0:
      traffic_type: imix2
      num_ofdm: 4
      num_subs: 300
    - port_1:
      traffic_type: imix2
      num_ofdm: 4
      num_subs: 300
  - pktgen_id: 1
    cpu_socket_id: 0
    ports:
    - port_0:
      traffic_type: imix2
      num_ofdm: 4
      num_subs: 300
    - port_1:
      traffic_type: imix2
      num_ofdm: 4
      num_subs: 300
  ...<entries for pktgen_id 2 to 14>
  ...
  - pktgen_id: 15
    cpu_socket_id: 1
    ports:
```
- port_0:
  traffic_type: imix2
  num_ofdm: 4
  num_subs: 300
- port_1:
  traffic_type: imix2
  num_ofdm: 4
  num_subs: 300
4.4 Sample CMK configuration file

A sample version of the CMK cluster-init yaml file, $MYHOME/vcmts/kubernetes/cmk/cmk-cluster-init.yaml is shown below for 16 service-group configuration option.

In this case, based on the number of service-groups selected as 16, the configuration tool sets num-exclusive-cores to 16 as the number of cores reserved for downstream traffic and num-shared-cores to 8 as the number of shared cores for upstream traffic (two upstream dataplane instances per core).

The host should also be set to the vCMTS dataplane server address.

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: cmk-cluster-init-pod
    name: cmk-cluster-init-pod
spec:
  serviceAccountName: cmk-serviceaccount
  nodeSelector:
    vcmts: "true"
  containers:
    - args:
      - " /cmk/cmk.py cluster-init --host-list=myvcmtsd-node.example.com --saname=cmk-serviceaccount --cmk-img=cmk:v1.3.0 --num-exclusive-cores=16 --exclusive-mode=spread --num-shared-cores=8 --shared-mode=spread"
      command:
        - "/bin/bash"
        - "-c"
      image: cmk:v1.3.0
      name: cmk-cluster-init-pod
  restartPolicy: Never
```

Note that for a single CPU system that the shared-mode parameter should be set to "packed" instead of "spread".
4.5 Upgrading Linux kernel and Linux packages

It may be necessary to upgrade to the latest versions of Linux kernel and Linux packages which are available for the Ubuntu LTS distro being used. For example, this may be required for security patches.

The steps below describe how to perform such an upgrade on the vCMTS traffic-generator and dataplane servers, and how to ensure that the Intel vCMTS Reference dataplane system still functions as expected.

4.5.1 Upgrade Linux on vCMTS traffic-generator server

After logging in to the vCMTS traffic-generator server (as root-user), the Linux upgrade should be performed manually as described below.

First, check the what updates are going to be applied.

```
sudo apt update
```

Next, apply the upgrade.

```
sudo apt upgrade
```

Following the Linux upgrade, it is recommended that software components on the vCMTS traffic-generator server are re-built as described below.

First, set the vCMTS reference dataplane environment by running the following command.

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

Rebuild DPDK as follows:

```
cd $MYHOME
install_dpdk
```

Re-build all Docker images by running the commands shown below.

⚠️ The “no-cache” option must be used to ensure that docker images are fully re-built.
4.5.2 Upgrade Linux on vCMTS dataplane server

After logging in to the vCMTS dataplane server (as root-user), the Linux upgrade should be performed manually as described below.

First, check the what updates are going to be applied.

```bash
sudo apt update
```

Next, apply the upgrade.

```bash
sudo apt upgrade
```

Following the upgrade, it is recommended that software components on the vCMTS dataplane server are rebuilt as described below.

First, set the vCMTS reference dataplane environment by running the following command.

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

Rebuild QAT drivers as follows:

```bash
cd $MYHOME
install_qat_drivers
```

(Select option '2 Install Acceleration' and select defaults when prompted)

Rebuild DPDK as follows:

```bash
cd $MYHOME
install_dpdk
```
Re-build all Docker images by running the commands shown below.

⚠️ The “no-cache” option must be used to ensure that docker images are fully re-built.

```bash
cd $MYHOME
build_docker_cmk --no-cache
build_docker_cloud_init --no-cache
build_docker_power_mgr --no-cache
build_docker_sriov_dp --no-cache
build_docker_qat --no-cache
build_docker_collectd --no-cache
build_docker_influxdb --no-cache
build_docker_grafana --no-cache
build_docker_vcmtsd_feat --no-cache
build_docker_vcmtsd_perf
```