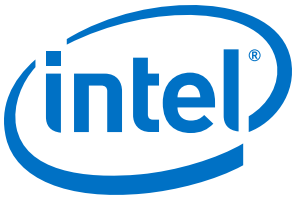


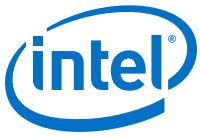
Visual Cloud Accelerator Card - Analytics Software Installation Guide

User Guide

Rev. 3.0

November 2019

Order No.: 611894-3.0

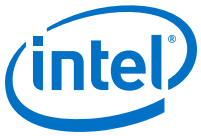


Contents

Revision History

Date	Revision	Description
November 2019	3.0	1.2 VCAC-A Contents of the Release on page 8: Updated contents of Release Package and added information about Dockerfiles. 2.1.2 Configure Docker Root Dir on CentOS on page 10: Added new chapter. 2.1.4 Build Host Kernel and VCAC-A Driver on page 12: Added information about addressing installation issues of docker; updated links; added build.sh workflow diagram. 2.1.5 Build System Image for VCAC-A on page 14: Added workflow diagram for vcad_build.sh; updated commands and links. 2.1.6 Skip building Docker Image(Optional) on page 15: Updated links and commands. 2.1.7 Customize System Image Size on page 15: Updated command. 2.2 Using VCAC-A System Image and Installation Package on page 15: Reorganized the chapter. 2.2.3 BIOS and EEPROM Update on page 17: Added new chapter. 2.2.3.1 BIOS and EEPROM Version Check on page 17: Added new chapter. 2.2.3.2 Online Update BIOS (optional) on page 17: Added new chapter. 2.2.3.3 Update EEPROM (optional) on page 18: Added new chapter. 2.2.5 VCAC-A Card and Host Network Setting Configuration on page 21: Added additional steps for configuring NAT. 2.2.6 VCAC-A Card Software Installation on page 22: Updated links and reorganized content. 2.2.7 Run Sanity Test on page 23: Reorganized the chapter. 2.2.7.2 Validate Setup with benchmark_app on page 24: Reorganized the chapter to explain how to run benchmark_app sample. 3.0 Run Media Analytics Pipeline with FFmpeg/GStreamer or HOST on page 25: Added new chapter on FFmpeg and GStreamer pipeline description and links. 3.1 Run Media Analytics Pipeline with HOST on page 25: Added new chapter on running pipeline with HOST. 3.2.1 Run Media Analytics Pipeline with FFmpeg on page 25: Added new chapter. 3.2.2 Run Media Analytics Pipeline with GStreamer on page 26: Added new chapter. Appendix B Appendix - Troubleshooting NAT Configuration (Optional) on page 28: Added new chapter on optional NAT configuration.
September 2019	2.0	1.2 VCAC-A Contents of the Release on page 8: Updated contents of Initial Release Package; added descriptions of folders. 1.3 VCAC-A Hardware Configuration on page 8: Updated hardware configuration table for hard drive information; added links to the card specifications and hardware customer support. 2.1.1 System Requirements to Prepare a Build on page 10: Added specification of Docker; updated topic for clarity. 2.1.4 Build Host Kernel and VCAC-A Driver on page 12: Added timing information for build process; updated links used for downloading packages. 2.1.5 Build System Image for VCAC-A on page 14: Added timing information for build process; updated links for downloading the binaries. 2.1.6 Skip building Docker Image(Optional) on page 15: Updated topic for clarity. 2.1.7 Customize System Image Size on page 15: Added new chapter. 2.2 Using VCAC-A System Image and Installation Package on page 15: Added commands to check BIOS and EEPROM version. 2.2.1 Setting up Intel® Xeon® Scalable Processor Server Host on page 15: Deleted extra commands which were optional. 2.2.2 VCAC-A Software Installation on Host on page 16: Updated file path of driver.

continued...



Date	Revision	Description
		2.2.4 VCAC-A Card Boot up with vcad Image on page 19: Added content which indicates health status of the VCAC-A card; added disk space requirement for vcad image; updated commands for clarity. 2.2.5 VCAC-A Card and Host Network Setting Configuration on page 21: Added steps for clarity. 2.2.6 VCAC-A Card Software Installation on page 22: Added information about sample application benchmark_app from OpenVINO; updated section for clarity. 2.2.7 Run Sanity Test on page 23: Added new chapter on running sanity test.
July 2019	1.0	Release 1.0
May 2019	0.1	Initial document version.

1.0 Introduction

Visual Cloud Accelerator Card - Analytics (VCAC-A) is a PCIe add on card comprising of Intel® Core™ i3-7100U Processor with Intel® HD Graphics 620 and 12 Movidius® VPUs. The addition of this card in any system focuses on computer vision, video decoding, and media analytics but is not only limited to them.

This document covers installation of Software release package for VCAC-A. The release package consists of source code, libraries, user mode service/agent components and kernel mode patches.

NOTE

Please get in touch with Intel Field Representative to receive latest Software Release Package.

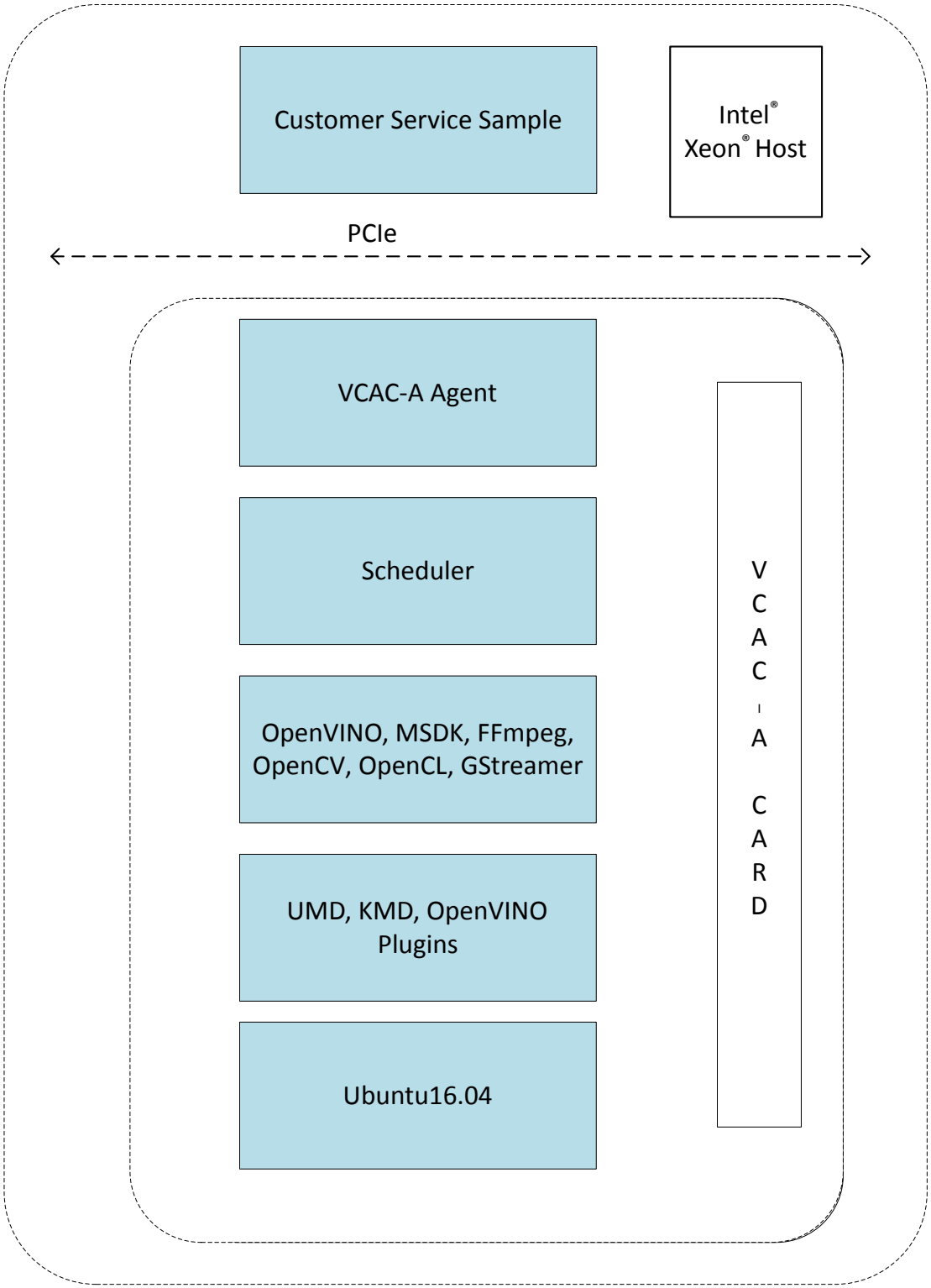
1.1 Reference Architecture

The VCAC-A software runs on the host and target (VCAC-A card) as follows:

1. VCAC-A service on the host
2. VCAC-A agent and all supported software on the VCAC-A card.

The architecture diagram shows the distribution of the software components.

Figure 1. Visual Cloud Analytics Accelerator Diagram



1.2 VCAC-A Contents of the Release

The contents of VCAC- A software release are listed in the table below. Intel drives these ingredients through continuous development and validation to ensure that software updates will perform correctly when integrated into a deployed system.

Download the release content from the link: https://github.com/OpenVisualCloud/VCAC-SW/archive/VCAC-A_R3.tar.gz

NOTE

The version of the release software might change. Please refer to the Release Notes Doc. No. 611358 accompanied with the latest Software Releases.

Table 1. VCAC-A Release Contents

Ingredient	Folder Name	Contents
VCAC-A Host Files (Intel_Media_Analytics_Host)	scripts	Scripts to build kernel and VCAC-A PCIe driver module and Dockerfile for creating the docker container of CentOS 7.4, called by the script to build kernel and driver modules
	tar	VCAC-A host supporting packages: kernel patch, VCAC-A PCIe driver patch
VCAC-A Software Package (Intel_Media_Analytics_Node)	scripts	Script to build the system image to be loaded on VCAC-A card and Dockerfile for creating the docker container of Ubuntu 16.04, called by the script to build system image to be loaded on VCAC-A.
	tar	Patches for kernel on card and patches for driver modules

Detailed software contents are listed as below:

VCAC-A host files:

- centos7.4-kernel3.10.0-patch.tar.gz
- vcass-modules-3.10.0-patch.tar.gz
- build.sh
- Dockerfile

VCAC-A card packages:

- vcass-modules-4.19-patch.tar.gz
- ubuntu16.04_kernel4.19_patch.tar.gz
- vcad_build.sh
- Dockerfile

1.3 VCAC-A Hardware Configuration

A Server with 2nd Generation Intel® Xeon® processor is recommended as the host to mount the VCAC-A card through PCIe.

NOTE

One of the example system configurations (assumed for all workloads described in this document) is listed in table below:

Table 2. System Hardware Configuration

Component Type	Part Name
KNOCK DOWN KIT	Intel® Server System R2312WFTZS 12x3.5in 2x10GbE
CPU	2nd Generation Intel® Xeon® Scalable Processor ***2**
MEMORY	16GB 2666 Reg ECC 1.2V DDR4 Micron MTA18ASF2G72PDZ-2G6D1 ***12**
ATA HARD DRIVE	480 GB Intel® SSD SATA or Equivalent Boot Drive
Add-in Card	VCAC-A card
NETWORK ADAPTER	On Board
CHASSIS COMPONENT	PCIe 2U Riser Spare Intel® A2UL16RISER2 (2 Slot)
CHASSIS COMPONENT	Passive Airduct Bracket Kit Intel® AWFCOPRODUCTBKT
CHASSIS COMPONENT	Passive Airduct Kit Intel® AWFCOPRODUCTAD
HEATSINK	Included

Table 3. VCAC-A Hardware Configuration

Component Type	Part Name
Form Factor	Full Height (126mm), ¾ Length (254mm) PCIe Adapter
Host Interface	PCIe Gen3 x4
On-board CPU	Intel® Core™ i3-7100U Processor, 2.4 GHz
Memory	2x DDR4 4GB SODIMM
VPU (Inference Acceleration)	12x Myriad X MA2485
Power Consumption	MAX75W
Thermal Cooling	Passive Heat Sink
Operating Temperature	0 º - 55 º C @ 15CFM airflow

Hardware specifications datasheet published by Intel is available here: http://intel.com/content/dam/www/public/us/en/documents/datasheets/media-analytics-vcac-a-accelerator-card-by-celestica-datasheet.pdf.

Hardware Customer Support is available from Celestica™ https://hardwaresupport.celestica.com.

NOTE

Customers need to work with their Celestica sales representative to get credentials for the website.

2.0 Installation

2.1 Build Host Kernel, VCAC-A Driver on Host and VCAC-A System Image

Following sections describe how to build host kernel, VCAC-A driver on host and VCAC-A system image with scripts included in the release package.

2.1.1 System Requirements to Prepare a Build

Build the kernel/driver and system image on the Intel® Xeon® host machine which has the VCAC-A card plugged in. They can also be built on any other machine with the following requirements of build environment:

- Linux based OS: the scripts are tested on CentOS 7.4
- Docker installed: Docker is used in the build process. Refer to https://docs.docker.com/install/ for how to install Docker Community version (docker-ce)
- User privilege: root account (or sudo) is required to run the script.

2.1.2 Configure Docker Root Dir on CentOS

By default, docker runs in folder `/var/lib/docker`. Check the location via the following command:

```
#docker info|grep Dir
Docker Root Dir: /var/lib/docker
```

The folder is under the `/dev/mapper/cl-root` folder and the size of the folder is 50GB in CentOS. The space is not enough for building the vcad and "no space left on device" might occur during compiling. It is recommended to change the Docker Root Dir to `/home` folder to avoid the size limit message:

```
#cd /home
#mkdir docker
```

Open configuration file " `/usr/lib/systemd/system/docker.service`", add "`--graph /home/docker`" after "`ExecStart=/user/bin/dockerd`"

```
[Service]
Type=notify

# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
```

```
ExecStart=/usr/bin/dockerd --graph /home/docker -H fd:// --containerd=/run/
containerd/containerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
```

Restart docker:

```
#systemctl daemon-reload
#systemctl restart docker
```

Check if Docker Root Dir has been changed via:

```
#docker info|grep Dir
Docker Root Dir: /home/docker
```

2.1.3 Setup Proxy for Docker (if needed)

Internet connection is needed to build Docker container, download source code and dependencies. If you are behind proxy, it needs to be setup for Docker.

2.1.3.1 Setup for Docker daemon

Take `systemd` for example: Create a `systemd` drop-in directory for the Docker service:

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

Create a file called `/etc/systemd/system/docker.service.d/http-proxy.conf` that adds the proxy environment variable:

```
[Service]
Environment="HTTP_PROXY=http://your.http-proxy-server.com:port"
"HTTPS_PROXY=http://your.https-proxy-server.com:port"
"NO_PROXY=localhost,127.0.0.1,*.your-company.com"
```

Flush changes:

```
#sudo systemctl daemon-reload
```

Restart Docker:

```
#sudo systemctl restart docker
```

Refer to https://docs.docker.com/config/daemon/systemd/ for details.

2.1.3.2 Setup for Docker client:

Create or edit the file `~/.docker/config.json`

```
{
  "proxies":
  {
```

```
            "default":
            {
              "httpProxy": "http://your.http-proxy-server.com:port",
              "httpsProxy": "http://your.https-proxy-server.com:port",
              "noProxy": "localhost,127.0.0.1,*.your-company.com"
            }
          }
        }
```

2.1.3.3 DNS setup in Docker

This is an optional step, in case build still cannot complete due to network issue. Setup the DNS for Docker.

Find your network's DNS server:

```
# nmcli dev show | grep 'IP4.DNS'
  IP4.DNS[1]: your.local.dns.server1
  IP4.DNS[2]: your.local.dns.server2
```

Open or create(if it doesn't exist), `/etc/docker/daemon.json` and add DNS settings:

```
# /etc/docker/daemon.json
  {
  "dns": ["your.local.dns.server1", "your.local.dns.server2"]
  }
```

Restart the Docker daemon:

```
 $ sudo service docker restart
```

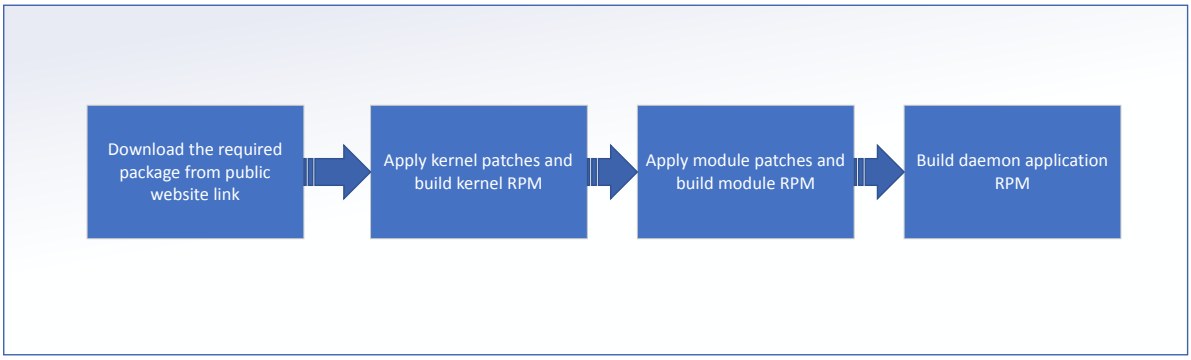
2.1.4 Build Host Kernel and VCAC-A Driver

Build host kernel and VCAC-A driver modules with the script included in the release package:

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh
```

The diagram below shows each step during the whole build process with "build.sh" to give an general overview of the workflow.

Figure 2. build.sh workflow diagram



If user gets stuck when the container tries to do a yum install even with proxy set, follow Docker BKM https://docs.docker.com/install/linux/linux-postinstall/ to allow running docker without sudo access.

After compilation is completed, the output will be available in `/PATH/TO/PACKAGE/Intel_Media_Analytics_Host/build/host_packages` ("fb4dfe2" in the name of the example output files were randomly generated).

NOTE

The build process takes approximately 1 hour. The duration might vary depending upon the capabilities of the machine and local network speed to download the dependent packages.

- Host kernel packages:
  - `kernel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm`
  - `kernel-devel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm`
- VCAC-A driver module: `vcass-modules-3.10.0_1.fb4dfe2.VCA+-1.690990a-0.x86_64.rpm`
- VCAC-A Utility files: `daemon-vca-2.7.3-x86_64.rpm`

During the build process, following packages will be downloaded:

- VCAC-A driver modules "`VCAC-SW-VCAC-A_R2.tar.gz`" : https://github.com/OpenVisualCloud/VCAC-SW/archive/VCAC-A_R2.tar.gz
- Host kernel base "`kernel-3.10.0-693.17.1.el7.src.rpm`" : http://vault.centos.org/7.4.1708/updates/Source/SPackages/kernel-3.10.0-693.17.1.el7.src.rpm

If you want to skip downloading source code and dependencies, put the files under `/PATH/TO/PACKAGE/cache`, then pass "-c" flag to the build script (this could save the time used for downloading the packages.), e.g.

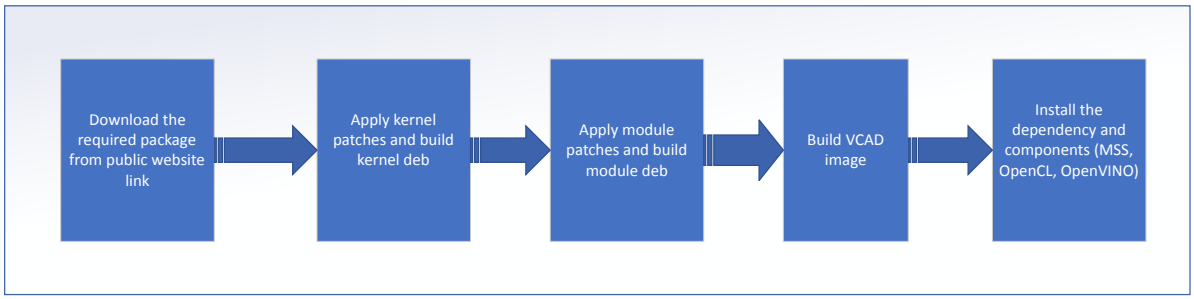
```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh -c
```

2.1.5 Build System Image for VCAC-A

Build VCAC-A system image with script included in the release package:

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -o FULL
```

Figure 3. vcad_build.sh workflow diagram



During the build process, following packages will be downloaded:

- VCAC-A on card kernel base "`linux-4.19.tar.gz`": https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.19.tar.gz
- Intel GPU firmware binary "`kbl_dmc_ver1_04.bin`": https://cgit.freedesktop.org/drm/drm-firmware/tree/i915/kbl_dmc_ver1_04.bin
- VCAC-A driver modules "VCAC-SW-VCAC-A_R2.tar.gz": https://github.com/OpenVisualCloud/VCAC-SW/archive/VCAC-A_R2.tar.gz
- numpy files: https://files.pythonhosted.org/packages/1f/c7/198496417c9c2f6226616cff7dedf2115a4f4d0276613bab842ec8ac1e23/numpy-1.16.4-cp27-cp27mu-manylinux1_x86_64.whl
- OpenCV files: https://files.pythonhosted.org/packages/77/30/36c3f0644fa9f42d92f079b972e990a5874c1fc2b2c0e9656eb88bb8d6dc/opencv_python-4.1.0.25-cp27-cp27mu-manylinux1_x86_64.whl

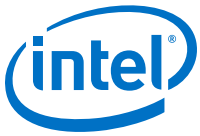
NOTE

The build process takes approximately 1 hour. The duration might vary depending upon the capabilities of the machine and local network speed to download the dependent packages.

The build script will always download the packages listed above for each time of execution. If the packages were already downloaded and you want to skip downloading source code and dependencies, put the files under `/PATH/TO/PACKAGE/cache`, then pass "-c" flag to the build script, e.g.

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -c -o FULL
```

After compilation is completed, the output will be available in `/PATH/TO/PACKAGE/Intel_Media_Analytics_Node/build/vcad/INSTALL/`:



- System image loaded on VCAC-A card:
  `vca_disk48_k4.19_ubuntu16.04_1.0.1.vcad.gz`

2.1.6 Skip building Docker Image(Optional)

Every time this build script is executed, docker images of the build environment will be generated. These images are - docker image of CentOS for building driver/kernel on host, ubuntu image for building kernel on the vcac-a card and vcad system image to be loaded on the card.

If the docker images have been generated in an earlier execution, user can save time required to build them again by the following executions:

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh -c -s
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -c -s -o
FULL
```

2.1.7 Customize System Image Size

The system image will be mapped from host disk via blockIO to VCAC-A card. The size of the vcad system image is set to 48GB by default. And the system image size is configurable through passing flag "-e" followed by the image size measured in GB, e.g.

```
 #sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -e 24 -o
FULL
```

NOTE

The default image size of 48GB is fully validated. User is recommend to keep the default image size.

2.2 Using VCAC-A System Image and Installation Package

This method requires that VCAC-A card is plugged in to the PCIe slot of the Intel® Xeon® Scalable Processor. Make sure to get the latest BIOS and EEPROM for VCAC-A card from card provider.

2.2.1 Setting up Intel® Xeon® Scalable Processor Server Host

- Install CentOS7.4 in host Intel® Xeon® Scalable Processor server
- Set proxy in host (Optional, depends on local network environment):

```
# vim /etc/yum.conf
export https_proxy="local network https proxy"
export http_proxy="local network http proxy"
export ftp_proxy="local network ftp proxy"
```

- Before installing the VCAC-A software package on host, check PCIe device availability through the following:

```
# lspci | grep PLX
af:00.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)
b0:01.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)
b0:02.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)
b0:03.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)

# lspci | grep 295
18:00.1 System peripheral: Intel Corporation Device 2952 (rev ca)
1a:00.0 Bridge: Intel Corporation Device 2954 (rev ca)
af:00.1 System peripheral: Intel Corporation Device 2952 (rev ca)
b1:00.0 Bridge: Intel Corporation Device 2954 (rev ca)
```

2.2.2 VCAC-A Software Installation on Host

Follow the steps below to install the VCAC-A software on host:

- Copy the kernel and VCAC-A card driver package compiled in Build Host Kernel and VCAC-A Driver on page 12 to a temporary folder on the host server.
- Install the kernel packages:

```
#sudo yum -y localinstall kernel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm
#sudo yum -y localinstall kernel-devel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm
```

- Configure the new kernel to be the default at boot.

NOTE

This step is very important to boot the operating system with proper kernel.

```
#sudo grub2-set-default 0
```

- Install the VCAC-A driver and utility.

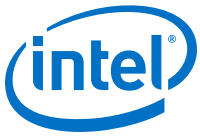
```
#sudo yum -y localinstall vcass-modules-3.10.0_1.fb4dfe2.VCA
+-1.690990a-0.x86_64.rpm
#sudo yum -y localinstall daemon-vca-2.7.3-x86_64.rpm
```

- Reboot the system to enable the new kernel.

```
#sudo reboot
```

- After reboot, confirm that the expected kernel on the host is being used.

```
#uname -r
3.10.0_1.fb4dfe2.VCA+
```



- (Optional) If updating from a previous VCAC-A version, remove the older RPMs with the following command:

```
#rpm -qa | grep -e daemon-vca -e vcass-modules | xargs yum -y erase
```

2.2.3 BIOS and EEPROM Update

Latest BIOS is available at https://hardwaresupport.celestica.com, and EEPROM is available at https://01.org/openvisualcloud/download (in section "Visual Cloud Accelerator Cards - Github Repos & Binary").

If following error is encountered, the EEPROM and BIOS need to be updated:

```
vcactl status
Card: 0 Cpu: 0  STATE: link_down, caterr
```

2.2.3.1 BIOS and EEPROM Version Check

Check the BIOS and EEPROM version via `vcactl info` command, sample output as below:

```
#vcactl info BIOS 0 0
Card 0 Cpu 0:
Version: VCAA.3.0
Release Date: 2019.06.24 13:47:34
```

```
#vcactl info hw 0 0
Card 0: VCAC-A,
EEPROM version: VCAC-A 1.3 (CRC:6cc483ef),
Serial Number: not-yet-readable
```

2.2.3.2 Online Update BIOS (optional)

Follow instructions below to update BIOS through `vcactl` commands via host in case the BIOS needs to be updated:

1. Check VCAC-A status via "`vcactl status 0 0`"

   If the card is not in "`bios_up`" state. Do step 2 and 3 to make sure the status is changed to "`bios_up`".

   If the card is in "`bios_up`" state, jump to step 4.

2. Power cycle the VCAC-A card:

   ```
   #vcactl pwrbtn-long 0 0
   #vcactl pwrbtn-short 0 0
   ```

3. Check VCAC-A status continuously with the following command till the status changes to "`bios_up`"

   ```
   # vcactl status 0 0
   Card: 0 Cpu: 0  STATE: bios_up
   ```

4. Update BIOS, this command takes about 2 minutes to be completed

```
#vcactl update-BIOS 0 0 /PATH/TO/BIOS/your_bios.img
Card: 0 Cpu: 0 - BIOS UPDATE STARTED. DO NOT POWERDOWN SYSTEM
Card: 0 Cpu: 0 - UPDATE BIOS SUCCESSFUL
Card: 0 Cpu: 0 - Node will power down and up automatically to make the change
active.
Please wait for 'bios_up' to start working with the node.
```

5. Check VCAC-A status continuously till the status is changed to "bios_up"

```
# vcactl status 0 0
Card: 0 Cpu: 0  STATE: bios_up
```

6. After status is changed to "bios_up", check BIOS version again to confirm if the bios was updated successfully

```
#vcactl info BIOS 0 0
```

2.2.3.3 Update EEPROM (optional)

Follow instructions below to update EEPROM through vcactl commands via host in case the EEPROM needs to be updated:

1. Check VCAC-A status via "vcactl status 0 0"

   If the card is not in "bios_up" state. Do step 2 and 3 to make sure the status is changed to "bios_up".

   If the card is in "bios_up" state, jump to step 4.

2. Power cycle the VCAC-A card:

```
#vcactl pwrbtn-long 0 0
#vcactl pwrbtn-short 0 0
```

3. Check VCAC-A status continuously with the following command till the status changes to "bios_up"

```
# vcactl status 0 0
Card: 0 Cpu: 0  STATE: bios_up
```

4. Update EEPROM with the following command.

   NOTE

   Please note that vcactl update-EEPROM command does not take "card id", so if there are multiple cards connected to one host, the EEPROM of all the cards will be updated.

```
#vcactl update-EEPROM /PATH/TO/EEPROM/your_eeprom.bin
update-EEPROM  eeprom_VCAA_v1.3_2.6.219.bin
Update EEPROM process started (for card 0). Do not power down system!
Update EEPROM for first PCIe switch successful!
Update EEPROM successful (for card 0). Reboot system is required to reload
EEPROM.
Update EEPROM process started (for card 1). Do not power down system!
```

```
Update EEPROM for first PCIe switch successful!
Update EEPROM successful (for card 1). Reboot system is required to reload
EEPROM.
```

If you see error like below :

```
ERROR: Found 0 matching eeprom binary entries for card 0 in input file, but
exactly 1 is required.
WARNING: Consider using `--skip-card-type-check' option
```

Then use command:

```
 #vcactl update-EEPROM /PATH/TO/EEPROM/your_eeprom.bin --force --skip-card-
type-check
```

5. Reboot host

```
reboot
```

6. Power cycle the card

```
#vcactl pwrbtn-long
#vcactl pwrbtn-short
```

7. Check VCAC-A status continuously with following command till the status is changed to "bios_up"

```
#vcactl status 0 0
```

8. After Card is up, check EEPROM version again to confirm if the bios was updated successfully

```
#vcactl info hw
```

2.2.4 VCAC-A Card Boot up with vcad Image

The system image will be mapped from host disk via blockIO to VCAC-A card. The size of the vcad system image is set to 48GB, so the host should have 48GB free space for the vcad image. This size can be customized as suggested here Customize System Image Size on page 15. Boot up the VCAC-A card with the vcad image with the steps listed below:

NOTE

LED lights on the VCAC-A card indicate the health status of the card:

- LED marked as "PW-LED" indicates the power status of the card.
- LED marked as "ERR" indicates that the card is in error state.

- Untar and load VCAC-A vcad image. The image was the output of steps done in Build System Image for VCAC-A on page 14 .

```
gzip -d vca_disk48_k4.19_ubuntu16.04_1.0.1.vcad.gz
#vcactl blockio open vcablk0 RW $PATH/vca_disk48_k4.19_ubuntu16.04_1.0.1.vcad
```

```
Note: If you have 2 VCAC-A cards on one host, use "0 0" and "1 0" in the
command to distinguish the cards:

#vcactl blockio open 0 0 vcablk0 RW $PATH/
vca_disk48_k4.19_ubuntu16.04_1.0.1.vcad

Note: One system image file cannot be loaded on two cards. Make a copy of the
system file, then load the copy to the 2nd card:

#cp vca_disk48_k4.19_ubuntu16.04_1.0.1.vcad
vca_disk48_k4.19_ubuntu16.04_1.0.1-copy.vcad
#vcactl blockio open 1 0 vcablk0 RW $PATH/vca_disk48_k4.19_ubuntu16.04_1.0.1-
copy.vcad
```

- Boot VCAC-A card

  Check VCAC-A status:

```
#vcactl status
```

  If status is anything other than "bios_up", for example "link_down" as in the following:

```
Card: 0 Cpu: 0  STATE:link_down,
```

  Then try the steps below:

```
#vcactl pwrbtn-long 0 0
#vcactl pwrbtn-short 0 0
```

NOTE

If there are more than 1 cards mounted on host machine, then the commands pwrbtn-long/pwrbtn-short will power cycle all the cards. It is important to pass the parameter "0 0" like above to power cycle only CARD 0.

  Check the status again via

```
#vcactl status
```

  If status is "bios_up".

```
Card: 0 Cpu: 0  STATE:bios_up
```

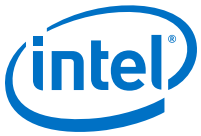
  Then boot the card via the following commands:

```
#vcactl reset 0 0 --force
#vcactl boot 0 0 vcablk0

Wait for about 30 seconds, then check the status, the STATE should turn into
"net_device_ready"

#vcactl status
Card: 0 Cpu: 0  STATE: net_device_ready
```

- If the device status is ready, you can login into the VCAC-A card using its IP address.



Check IP address of the VCAC-A. Following is an example of one host connected with two VCAC-A cards:

```
#vcactl network ip
Card 0 Cpu 0:
172.32.1.1
Card 1 Cpu 0:
172.32.2.1
```

- Login to VCAC-A Card 0 through `ssh`:

```
#ssh root@172.32.1.1
(password:vista1)
```

2.2.5 VCAC-A Card and Host Network Setting Configuration

Network settings need to be configured in VCAC-A card and host. Following steps are required to achieve them:

Log in to the host machine and configure NAT:

- Create the proxy file to add necessary proxy settings if proxy is required to connect to external network.

```
#touch /etc/yum.repos.d/10proxy
#vim 10proxy
Acquire::http::Proxy " local network http proxy ";
#scp /etc/yum.repos.d/10proxy root@172.32.1.1:/etc/apt/apt.conf.d/10proxy
```

- Setup DNS server on host and copy the configuration file to VCAC-A card

```
#vim /etc/resolv.conf
nameserver name server 1 ip
nameserver name server 2 ip
nameserver name server 3 ip
search sh.intel.com
#scp /etc/resolv.conf root@172.32.1.1:/etc
```

- Disable the firewall

```
#systemctl stop firewalld.service
#systemctl disable firewalld.service
#systemctl status firewalld.service
```

- Stop the Network Manager

```
#systemctl stop NetworkManager
```

- Enable forwarding in kernel

```
#echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Add rules to NAT (**run every time after reboot**)

```
# iptables -t nat -A POSTROUTING -s 172.32.1.1 -d 0/0  -j MASQUERADE
```

(Optional) If ip forwarding does not work with the above command, try adding FORWARD rules, for example:

```
iptables -A FORWARD -i eno1 -o eth0 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A FORWARD -i eth0 -o eno1 -j ACCEPT
```

Refer to Appendix - Troubleshooting NAT Configuration (Optional) on page 28 for further details.

Log into the VCAC-A card, and set up proxy for it:

- Set up proxy for VCAC-A card

```
#vim ~/.bashrc
export https_proxy=" local network https proxy"
export http_proxy=" local network http proxy"
export ftp_proxy=" local network ftp proxy"
```

2.2.6 VCAC-A Card Software Installation

As the VCAC-A card is boot up with vcad image, login to VCAC-A card through `ssh`, and install the VCAC-A software as below:

- Install dependencies

```
#apt update
#apt install -y build-essential curl wget libssl-dev ca-certificates git
libboost-all-dev gcc-multilib g++-multilib libgtk2.0-dev pkg-config libpng12-
dev libcairo2-dev libpango1.0-dev libglib2.0-dev libgstreamer0.10-dev
libusb-1.0-0-dev i2c-tools libgstreamer-plugins-base1.0-dev libavformat-dev
libavcodec-dev libswscale-dev libgstreamer1.0-dev libusb-1.0-0-dev i2c-tools
libjson-c-dev usbutils ocl-icd-libopencl*  ocl-icd-opencl-dev python-pip
libjasper-dev python3 python3-pip
#export LC_ALL=C
#pip install opencv-python
```

- Load SMBus driver

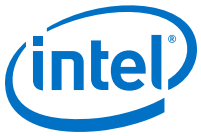
```
# modprobe i2c-i801
# modprobe i2c-dev
```

Check SMBus status. User should be able to see `smbus` device

```
#i2cdetect -l

i2c-3   i2c             i915 gmbus dpb                          I2C adapter
i2c-1   i2c             Synopsys DesignWare I2C adapter         I2C adapter
i2c-6   smbus           SMBus I801 adapter at f040              SMBus adapter
i2c-4   i2c             i915 gmbus dpd                          I2C adapter
i2c-2   i2c             i915 gmbus dpc                          I2C adapter
i2c-0   i2c             Synopsys DesignWare I2C adapter         I2C adapter
i2c-5   i2c             DPDDC-B                                 I2C adapter
```

- Load HDDL driver



Check driver status:

```
#lsmod | grep myd

Correct output should be like below:
myd_vsc                24576  0
myd_ion                61440  0
```

If not, do the following steps:

```
#insmod /lib/modules/4.19.0-1.29bfe52.vca+/kernel/drivers/ion/myd_ion.ko
#insmod /lib/modules/4.19.0-1.29bfe52.vca+/kernel/drivers/usb/myd/myd_vsc.ko
(Note: "29bfe52" is a randomly generated number during compilation)
```

- Make sure 12 usb (Movidius VPU) devices can be found (device 2485)

```
# lsusb
Bus 012 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 011 Device 023: ID 03e7:2485
Bus 011 Device 022: ID 03e7:2485
Bus 011 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 009 Device 023: ID 03e7:2485
Bus 009 Device 022: ID 03e7:2485
Bus 009 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 010 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 008 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 007 Device 021: ID 03e7:2485
Bus 007 Device 022: ID 03e7:2485
Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 005 Device 022: ID 03e7:2485
Bus 005 Device 023: ID 03e7:2485
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 023: ID 03e7:2485
Bus 003 Device 022: ID 03e7:2485
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 023: ID 03e7:2485
Bus 001 Device 022: ID 03e7:2485
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

- Set test environment

```
# source /etc/profile
# ldconfig
```

- Run script from OpenVINO to set environment for OpenVINO

```
#source /opt/intel/openvino/bin/setupvars.sh
```

2.2.7 Run Sanity Test

Test for VCAC-A software setup before proceeding further. To do this, try to run an OpenVINO sample application named `benchmark_app`.

2.2.7.1 Build OpenVINO Sample

After setup of software ingredients for VCAC-A is done, sample from OpenVINO can be built by following the steps below:

- Install dependencies

```
#cd /opt/intel/openvino/install_dependencies
# ./install_openvino_dependencies.sh
```

- Build Samples

```
#cd /opt/intel/openvino/inference_engine/samples
# ./build_samples.sh
```

After Build is completed, binaries for OpenVINO samples are in folder `/root/inference_engine_samples_build/intel64/Release`.

2.2.7.2 Validate Setup with benchmark_app

After sample application build is complete, `benchmark_app` is available in `/root/inference_engine_samples_build/intel64/Release`.

Follow instructions below to run `benchmark_app` for sanity test:

Download the model files:

```
 # cd /root/inference_engine_samples_build/intel64/Release
wget https://download.01.org/opencv/2019/open_model_zoo/
R3/20190905_163000_models_bin/vehicle-detection-adas-0002/FP16/vehicle-detection-
adas-0002.bin
wget https://download.01.org/opencv/2019/open_model_zoo/
R3/20190905_163000_models_bin/vehicle-detection-adas-0002/FP16/vehicle-detection-
adas-0002.xml
```

Run commands as below ("`car_1.bmp`" is a sample image included as part of OpenVINO release):

```
# source /opt/intel/openvino/bin/setupvars.sh
# cd /root/inference_engine_samples_build/intel64/Release
#
./benchmark_app -i /opt/intel/openvino/deployment_tools/demo/car_1.bmp -m vehicle-
detection-adas-0002.xml -d HDDL -niter 1000 -nireq 128
```

The output of this command can be found here in Appendix - Sample Output Message for benchmark_app on page 27.

3.0 Run Media Analytics Pipeline with FFmpeg/ GStreamer or HOST

Video is the fastest growing data and to support real-time analytics on video streams, it is essential to accelerate complex and intensive operations like media decode/ encode and inference. VCAC-A card is designed to accelerate these operations and provide dense solution.

To construct an end to end pipeline, Intel supports popular multimedia industry frameworks - FFmpeg and GStreamer. These frameworks allow to build complex applications combining variety of multimedia blocks using vast libraries to process video, audio and inference.

3.1 Run Media Analytics Pipeline with HOST

HOST (Heterogeneous Optimized Scheduling Tool), which is part of the release, is an optimization tool to build efficient and flexible pipelines for a variety of workloads (like decoding, inferencing, etc) based on API/SDKs (MSDK, OpenVINO). It provides heterogeneous scheduler to deploy routines in task to different hardware components (GPU/CPU/VPU).

Media Analytics end to end pipeline could be constructed with HOST. The contents and guide are available in IPS, contact Intel representative if interested with this solution.

3.2 Run Media Analytics Pipeline

There are two options to install FFmpeg or GStreamer Media Analytic plugin:

- Install in Docker image.
- Install directly in system.

If chosen to install the FFmpeg/GStreamer Video Analytics in docker image, refer to the following link for the extra steps to setup VCAC-A to run docker containers:

https://github.com/OpenVisualCloud/Dockerfiles/blob/master/VCAC-A/README.md

3.2.1 Run Media Analytics Pipeline with FFmpeg

Intel provides FFmpeg video analytics plugin which is built on Intel OpenVINO's Inference Engine. It brings deep learning capabilities like object detection, classification and recognition to open-source framework FFmpeg and it helps developers and customers to build highly efficient and scalable video analytics applications.

For the detailed instructions, please refer to the Getting Started guide at https://github.com/VCDP/FFmpeg-patch/wiki/Getting-Started-Guide. It contains the steps to build FFmpeg with the analytics plugin and the examples to run analytics pipelines.

3.2.2 Run Media Analytics Pipeline with GStreamer

GStreamer is an open source framework for processing of video/audio streaming data. It provides tools and APIs for pipeline management and many plug-ins (over 250 plug-ins with more than 1000 elements) for building extensible media pipeline. Intel provides GStreamer Video Analytics (GVA) plugin, that contains multiple GStreamer elements to construct video analytics pipelines. GVA plugin has two types of elements:

1. The elements that provide inference operations such as detection, classification, identification using OpenVINO Inference Engine.

2. The elements that provide handling of inference output.

All the GStreamer elements are highly optimized for Intel Hardware and it is an open source project. The repository link below provides Wiki where you can find `Getting Started Guide`, `README`, `API reference and Performance Optimization technique` for Intel Platforms. The repository also contains code samples for object detection, face detection etc.

gst-video-analytics: https://github.com/opencv/gst-video-analytics

Appendix A Appendix - Sample Output Message for benchmark_app

Sample output message after running benchmark_app:

```
----------<Message for step 1 and 2 are truncated>--------------------

[Step 3/11] Reading the Intermediate Representation network
[ INFO ] Loading network files
[Step 4/11] Resizing network to match image sizes and given batch
[ INFO ] Network batch size: 1, precision: MIXED
[Step 5/11] Configuring input of the model
[Step 6/11] Setting device configuration
[Step 7/11] Loading the model to the device
[05:43:19.2564][3713]I[ServiceStarter.cpp:40] Info: Waiting for HDDL Service
getting ready ...
[05:43:19.2565][3713]I[ServiceStarter.cpp:45] Info: Found HDDL Service is running.
[HDDLPlugin] [05:43:19.2566][3713]I[HddlClient.cpp:256] Hddl api version: 2.2
[HDDLPlugin] [05:43:19.2566][3713]I[HddlClient.cpp:259] Info: Create Dispatcher2.
[HDDLPlugin] [05:43:19.2568][3717]I[Dispatcher2.cpp:148] Info: SenderRoutine
starts.
[HDDLPlugin] [05:43:19.2568][3713]I[HddlClient.cpp:270] Info: RegisterClient
HDDLPlugin.
[HDDLPlugin] [05:43:19.2571][3713]I[HddlClient.cpp:275] Client Id: 2
[Step 8/11] Setting optimal runtime parameters
[ WARNING ] Number of iterations was aligned by request number from 1000 to 1024
using number of requests 128
[Step 9/11] Creating infer requests and filling input blobs with images
[HDDLPlugin] [05:43:24.1477][3713]I[HddlBlob.cpp:166] Info: HddlBlob initialize
ion ...
[HDDLPlugin] [05:43:24.1477][3713]I[HddlBlob.cpp:176] Info: HddlBlob initialize
ion successfully.
[ INFO ] Network input 'data' precision U8, dimensions (NCHW): 1 3 384 672
[ WARNING ] Some image input files will be duplicated: 128 files are required but
only 1 are provided

----------------------------<Truncated Here>-------------------------
[Step 10/11] Measuring performance (Start inference asyncronously, 128 inference
requests, limits: 1024 iterations)
[Step 11/11] Dumping statistics report
[ INFO ] Statistics collecting was not requested. No reports are dumped.
Count:      1024 iterations
Duration:   3535.34 ms
Latency:    414.678 ms
Throughput: 289.647 FPS
[HDDLPlugin] [05:43:28.4873][3718]I[Dispatcher2.cpp:212] Info: Listen Thread wake
up and to exit.
[HDDLPlugin] [05:43:28.4873][3713]I[Dispatcher2.cpp:81] Info: Client dispatcher
exit.
[HDDLPlugin] [05:43:28.4874][3713]I[HddlClient.cpp:203] Info: Hddl client
unregistered.
```

Appendix B Appendix - Troubleshooting NAT Configuration (Optional)

Under normal conditions, following command run after every reboot should set NAT configuration:

```
#iptables -t nat -A POSTROUTING -s 172.32.1.1 -d 0/0 -j
MASQUERADE
```

Following will be optional steps in case NAT configuration does not work. Please check relevant settings of the local network environment or customer's own platform network setup.

1. echo 1 > /proc/sys/net/ipv4/ip_forward
2. ip route show

   This step helps to check the network interface where the ip traffic is routed on the host. For example, across eno1 or eno2 and so on.
3. /sbin/iptables -t nat -A POSTROUTING -o <eno1 or eno2> -j MASQUERADE
4. /sbin/iptables -A FORWARD -i <eno1 or eno2 > -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
5. /sbin/iptables -A FORWARD -i eth0 -o <eno1 or eno2> -j ACCEPT

Glossary

HDDL

High Density Deep Learning

NAT

Network Address Translation

PCIe

Peripheral Component Interconnect Express

SMBus

System Management Bus

VCAC-A

Visual Cloud Accelerator Card - Analytics

VPU

Visual Process Unit