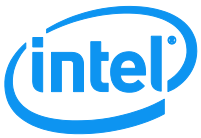


Intel® Visual Compute Accelerator VCA1283LVV

Product Software Guide

A *Reference document* used to assist with setting up the software for the Intel® Visual Compute Accelerator.

Revision 1.0

November 2015

Document Revision History

Date Published	Revision	Revision Change Description
November 2015	1.0	First Public Release

Disclaimer

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described 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.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel, the Intel logo, Xeon, and Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

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

Table of Contents

1. Intel® Visual Compute Accelerator Overview

The Intel® Visual Compute Accelerator combines the graphics capabilities of the Intel® Iris Pro Graphics within the Intel® Xeon E3-1200 v4 processor with the power of the Intel® Xeon® E5-2600 v3 family processors. Designed around visual computing workloads, this PCIe add-in card can handle the highest end 4K/HEVC transcoding jobs as well as many simultaneous AVC jobs. Built for cloud service providers, Telco service providers and broadcasters who need media transcoding acceleration for the growing video market.

Outstanding Performance / Power / Price per transcode

- Real time HEVC and AVC transcode performance delivered at low cost and power

Fit within Intel® Xeon® Processor E5 Infrastructure

- Multiple 1U and 2U Server System options from several OEM's
- PCIe* Gen3 x16 enables fast communication between host and adapter

Flexible and Portable software architecture

- Full access to integrated CPU & GPU for best quality/performance
- Virtual network allows host and card to communicate seamlessly
- Strong ISV ecosystem for ease of solution deployment

VCA1283LVV PCIe Card SPECIFICATIONS

PCIe Card Specification	Details
Form Factor	Full-Length, Full-Height, Double width PCIe Card
CPU	3 x Intel® Xeon® E3-1200v4, 47W TDP, 2.9GHz, 4 cores
Graphics	GT3e, Iris Pro Graphics P6300, 128MB eDRAM
PCH	LynxPoint-H PCH per CPU
Memory	DDR3L (1.35V), ECC SODIMMs, 2 channels per CPU, Up to 32GB per CPU
PCI Express Configuration	Gen3, x16, 4 lanes per CPU
BIOS	16MB SPI Flash
Operating System Support	CentOS 7.1, Xen support if using hypervisor

2. Software Components

2.1 Required Software

- CentOS 7.1 install media (http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Everything-1503-01.iso)
- Intel® Visual Compute Accelerator Software (http://downloadcenter.intel.com)
    - Updated kernel (kernel-3.10.0_<version>.VCA-1.x86_64.rpm)
    - VCA daemon (daemon-vca-<version>.x86_64.rpm)
    - VCASS drivers (vcass-modules-3.10.0-<version>.VCA-<version>-0.x86_64.rpm)
    - Boot Image (several to choose from. See Startup Procedure for a description of each image)

2.2 Optional Software

For performing BIOS updates

- VCA-BIOS-<version>.img

For building Intel® Media Server Studio

- kernel_devel-3.10.0_<version>.VCA-1.x86_64.rpm (required to build i915 driver)
- Intel® Media Server Studio from https://software.intel.com/en-us/intel-media-server-studio

For running sample transcodes

- Intel® Media Server Studio from https://software.intel.com/en-us/intel-media-server-studio

3. Installing Intel® Visual Compute Accelerator Software

1. Boot system and log into CentOS 7.1
    a. *Note: For instructions on how to install CentOS 7.1, refer to the later section under CentOS Tips and Tricks.*
2. Copy the required files to the server.
3. Open a terminal session
    a. Applications -> Utilities -> Terminal
4. Change directory to the directory where the required files were placed
5. Go into su mode (or type sudo before every command)

    ```
    su
    ```

    #enter root password
6. If updating from a previous version, remove older RPM's

    ```
    rpm -qa | grep -e daemon-vca -e vcass-modules | xargs yum -y erase
    ```
7. Install the kernel patch

    ```
    yum -y localinstall --disablerepo=* kernel-3*rpm
    ```

8. Install Intel® Visual Compute Accelerator driver and utility
   ```
   yum -y localinstall --disablerepo=* daemon-vca*rpm
   yum -y localinstall --disablerepo=* vcass-modules*rpm
   ```
9. Configure the new kernel to be the default at boot
   ```
   grub2-set-default 0
   ```
10. Reboot the system to enable the new kernel
    ```
    reboot
    ```

4. Using the software

1. Open a terminal session
   Applications -> Utilities -> Terminal
2. Change directory to where the required files were placed
3. Go into su mode (or type sudo before every command)
   ```
   su
   ```
   #enter root password
4. Verify nodes are ready for boot
   ```
   vcactl status
   ```
   #user will see a “bios_up” for each node (3 per card)
5. Select the Intel® Visual Compute Accelerator boot image. User has several options to choose from:
   I. Production Images (recommended to use)
      a. vca_baremetal_production<version>.img
         i. Baremetal CentOS 7.1 image. Recommended to use. This image loads the entire OS within a RAMDisk. As this is read only memory, changes are not persistent. If one or more folders need to be persistent, see section on “Manually creating persistent file system using NFS”. Note: This is different than the persistent image instructions where all folders are persistent.
         ii. This image does not contain MSS drivers and is built using production kernel (without MSS patches)
         iii. The image is fully tested by validation team (including stability and performance tests)
   II. Sample Images (demo images to show possible configurations)
      b. vca_baremetal_<version>.img
         i. Baremetal CentOS 7.1 image. This image loads the entire OS within a RAMDisk. As this is read only memory, changes are not persistent. If one or more folders needs to be persistent, see section on “Manually creating persistent file system using NFS”. Note: This is different than the persistent image instructions where all folders are persistent.
         ii. The image contains reference kernel (the one with applied MSS patches) and contains the latest publicly released MSS Community Edition (without media samples package)

iii. The image is tested by validation team in limited way (basic scenarios only)

c. vca_persistent_<version>.img
   i. This image also requires vca_rootfs_<version>-tree.tar.gz
   ii. This image boots a minimal image and all other folders are mounted back to an NFS share on the host. The name of the folders and share must be exactly as described. See section on "Setting up host for persistent file image". Note: this is different than "Manually creating persistent file system using NFS".
   iii. The image contains reference kernel (the one with applied MSS patches) and contains the latest publicly released MSS Community Edition (without media samples package)
   iv. The image is tested by validation team in limited way (basic scenarios only)

d. vca_xen_<version>.img
   i. Also requires vca_domu_<version>.tar.gz if using CentOS 7.1 as a guest OS
   ii. This image boots Xen hypervisor and CentOS 7.1 in domain 0. User can then load to domain U whatever guest OS they want, as long as it is supported by chipset, graphics drivers as well as the Intel® Media Server Studio. A CentOS 7.1 guest image with production kernel and MSS packages is provided.
   iii. The image contains reference kernel, but doesn't contain MSS (as graphics shall be accessed from DomU OS).
   iv. The image is fully tested by validation team

e. Custom image – See section 13 for instructions on how to build a custom image.

6. Boot the Intel® Visual Compute Accelerator with selected image.
   vcactl boot <img file>
7. Wait for the nodes to come fully online, then check the status of the nodes; when status is "net_device_ready", the systems should be ready to use.
   vcactl status
8. Verify the virtual network interfaces are online
   ip a
   # This should display ethX for each Intel® Visual Compute Accelerator node identified, where X is node number minus one
   # nodes would have the following entries: eth0, eth1, eth2, … ethX
9. Use SSH to connect to the nodes
   ssh 172.31.X.1

   # where X is the number of the node starting with 1 (i.e. 172.31.1.1 for first node, 172.31.2.1 for second node and so on)

IMPORTANT: The username and password for all images are pre-set to root as the username and "vista1" as the password. Customers who expose these nodes outside of the host machine are strongly recommended to build their own image and use persistent file

system where passwords are stored or implement LDAP. If using baremetal/volatile image and the password is changed, it will be reset to vista1 after the reboot.

10. If using vca_xen image, follow steps in the section 6 to start a virtual machine.

5. Use MSS with production kernel and image

VCA strategy is to test the VCA software stack against the most recent, publicly released MSS stack. But some customers may want to keep with older, well tested in their environment, MSS release, or have an access to some not public (obtained under NDA) MSS releases. To cover such needs, the production VCA software stack doesn't include MSS at all.

MSS contains two main parts (in terms of driver form):

a) an open source Kernel Mode Drivers (KMD) in form of text patches to be applied on used kernel
b) binary User Mode Drivers and libraries (UMD), to be only installed

Note: in the following commands, parts in triangle brackets < > represents a non constant value, such as kernel version (<kernel version>), build version (<build version>) or some location path (<temp directory>). Replace the whole string, including brackets, with a proper value.

1. Build GPU driver with MSS patches for production kernel

Production kernel for VCA software stack contains only a CentOS kernel and set of patches required to establish virtual network connection with a VCA card. Customer needs to apply patches for i915 driver, delivered with MSS distribution and rebuild a driver module to be compatible with the current kernel.

The procedure below may be done "offline" (not on VCA node), for example on a host or external computer. It can be done once for a given VCA kernel and MSS release and the built module can be copied to each booted VCA node. The module needs to be recompiled when:

a) new VCA kernel is released
b) new MSS is released

2. Install kernel devel package

Install kernel devel package for a given production kernel. The package contains source code and a symfile, allowing to compile modules to be compatible with the kernel.

Change directory to where the kernel-devel package was placed

```
yum install kernel-devel-3.10.0_<kernel version>.x86_64.rpm
```

3. Extract patches from MSS devel package

Download Intel® Media Server Studio (MSS) from https://software.intel.com/en-us/intel-media-server-studio. Unzip and untar the package. Depending on the version an additional unzip and untar of the SDK2015Production<version>.gz may be required.

MSS delivers set of patches to be applied on a kernel in a form of archive included into a devel package. Installation of the package isn't required, especially it has dependencies on non-VCA kernel headers package. Instead. It is sufficient to extract only an archive with patches. Extraction shall be done to some temporary directory.

Change directory to *SDK<year>Production<version>/CentOS* inside where MSS was placed. This folder should contain series of RPM's and other files.

```
mkdir temp
cd temp

rpm2cpio ../intel-linux-media-devel-<MSS version>.el7.x86_64.rpm
| cpio –idmv
./opt/intel/mediasdk/opensource/patches/kmd/3.10.0/intel-kernel-
patches.tar.bz2
```

Note: above command is all one line starting from rpm2cpio ending with bz2

Decompress the archive:

```
tar xf ./opt/intel/mediasdk/opensource/patches/kmd/3.10.0/intel-
kernel-patches.tar.bz2
```

4. Apply MSS patches to VCA kernel

Go to a directory with installed sources from a kernel-devel package

```
cd /usr/src/kernels/3.10.0-<kernel version>
```

Apply all MSS patches:

```
for patchfile in <temp directory>/intel-kernel-patches/*.patch;
do echo "*Applying basename $patchfile"; patch -p1 –I $patchfile;
done
```

5. Build the module

While staying within kernel sources directory (/usr/src/kernels/3.10.0-<kernel version>), invoke:

```
make M=drivers/gpu/drm/i915/ -j 31
```

Compilation will start and at the end the location of generated module (i915.ko) will be displayed:

```
Building modules, stage 2.

MODPOST 1 modules
CC     drivers/gpu/drm/i915//i915.mod.o
LD [M] drivers/gpu/drm/i915//i915.ko
```

6. Start MSS at VCA node

The following steps need to be done after each reset of a VCA node

7. Boot a production image

Change directory to where the VCA files were placed

```
vcactl boot vca_baremetal_production_<build version>.img
```

8. Copy MSS components to VCA node

Patched driver shall be copied to kernel modules directory

```
scp /usr/src/kernels/3.10.0-<kernel
version>/drivers/gpu/drm/i915/i915.ko root@<node's
ip>:/usr/lib/modules/3.10.0-<kernel
version>/kernel/drivers/gpu/drm/i915/
```

Note: repeat for each node

Change directory to *SDK<year>Production<version>/CentOS* inside where MSS was placed. This folder should contain series of RPM's and other files. Below example assumes copying files to /home/MSS but any folder can be used.

```
scp * root@<node's ip>:/home/MSS
```

Note: repeat for each node

If devel packages from MSS are needed to be installed (to compile other components to work with MSS, for example, external h265 codec), copy also a kernel-headers package from the build:

```
scp kernel-headers-3.10.0_<kernel version>.x86_64.rpm
root@<node's ip>:~
```

9. Install MSS at VCA node

Login to the VCA node using SSH. Load patched module

```
modprobe i915
```

Install MSS (only executables, without devel packages). Change directory to where MSS RPM's were placed above.

```
ls *rpm | egrep -v "samples|devel" | xargs yum -y localinstall
```

Or install MSS with devel packages.

```
yum install ./kernel-headers-3.10.0_<kernel version>.x86_64.rpm
ls *rpm | egrep -v "samples" | xargs yum -y localinstall
```

To properly set environmental variables, log out from the session and log in again using ssh

6. Running virtualization on Intel® VCA

IMPORTANT: After setup is completed, and operating system booted on Intel® VCA nodes, it is important that users do not restart the NFS on the host. This could cause a kernel panic for the nodes booted on the card.

The Intel® VCA can be run in a virtualized environment. Users may want to run in virtualized environment so that they can run their application in an OS other than CentOS 7.1 or they may be using virtualization orchestration tools to start their application. A single VM is the only supported configuration at this time. This section describes how to boot the XEN hypervisor and load a guest OS.

Example guide to set-up CentOS 7 NFS server or see section chapter on setting up NFS under CentOS Tips and Tricks

https://www.howtoforge.com/nfs-server-and-client-on-centos-7

Configuration steps:

1. Install nfs-utils package:

```
yum install nfs-utils
```

2. Create nfs mount points with node's filesystem (setup for 2 cards):

```
mkdir /mnt/xen
```

Note: Any folder can be created to host the Dom U files. For this document /mnt/xen is used as an example. If different folder is selected, change below occurrences of /mnt/xen as well.

3. Create or update /etc/exports with this content:

```
/mnt/xen *(rw,sync,no_root_squash,no_all_squash)
```

4. Copy Dom U image and required scripts to mount folder

```
tar -zxvf ../vca_domu_<version>.tar.gz -C /mnt/xen/
cp /usr/sbin/vca_st   art_host_domu.sh /mnt/xen/
cp /usr/lib/vca/card_gfx_vm.hvm /mnt/xen
cp /usr/lib/vca/card_vm.hvm /mnt/xen
```

5. Start the nfs server:

```
systemctl enable rpcbind
systemctl enable nfs-server
systemctl start rpcbind
systemctl start nfs
```

6. Make sure the firewall is properly configured:

```
firewall-cmd --permanent --zone=public --add-service=nfs
firewall-cmd --reload
```

Or you can disable firewall entirely:

```
systemctl stop firewalld
systemctl disable firewalld
```

7. Make copy of DomU image file for each node in the system

Each DomU image file will be written to by the respective virtual machine. Therefore to prevent more than one VM writing to the same image file, it is required that there be a different image file per VM. The script below assumes 6 nodes (2 cards) in the system and that the tar file is located in the /mnt/xen folder. Script also assumes that the user is using the reference DomU image file from the Intel® VCA software package.

```
for i in {1..6}
```

```
do
cp /mnt/xen/vca_domu* /mnt/xen/vca_domU_$i.img
done
```

8. Boot the nodes with the xen virtualized image:

```
vcactl boot vca_xen_<version>.img
```

wait for the nodes to be ready "`vcactl status`" should show the nodes status as "net_device_ready"

9. Start DomU for each node in the system

To start DomU with graphic passthrough enabled on node 0 and card 0:

```
vca_start_card_domu.sh -p <NFS share> -g -f card_gvx_vm.hvm -c 0 -n 0
-i vca_domu_1.img
```

To start DomU without graphic passthrough enabled:

```
vca_start_card_domu.sh -p <NFS share> -f card_vm.hvm -c 0 -n 0 -i
vca_domu_1.img
```

change –c, -n, and –i parameters for different nodes. –c is the card to start the VM on starting with 0. –n is the CPU within that card to start the VM on starting with 0. –i is the name of the image built with step #2 above.

See table below for set of options to vca_start command. If DomU IP address is not changed by the parameter below, the default IP will be 172.31.X.2 where X is the node starting with 1.

Option	Operand	Description
-f	filename	VM configuration file. Must be located within NFS shared directory
-g		Enable graphics passthrough
-p	Path	NFS share location (path at host); using /share if not configured
-a	IP	Card's IP address (if other than VCA xml defined is used)
-h	IP	Host's IP address (if other than VCA xml defined is used)
-u	IP	DomU IP address (if other than default is used); this is for routing purposes only
-N	IP	NFS address (if different than host)
-c	ID	Card ID (0, 1, 2, 3); using 0 if not provided
-n	ID	Node ID (CPU at card; 0, 1, 2); using 0 if not provided; used to generate DomU MAC address (if not provided with -m option) and and host's address (if not provided by -a/-h)
-i	filename	VM image filename. Overrides image configured in config file
-b		Use network bridging instead of routing
-k	filename	Kernel image filename. For booting VMs without bootloader (doesn't support graphics pass-through)

-r	filename	InitRamFs image name. For booting VMs without bootloader (doesn't support graphics pass-through)

9. Add routing on Dom0 on each node

To add routing on each node.
Connect to node:

```
ssh 172.31.x.1
```

where X is the number of the node starting with 1 (i.e. 172.31.1.1 for first node, 172.31.2.1 for second node and so on)
Execute command:

```
route add 172.31.1.253 gw 172.31.1.254
```

7. Command Line Reference

7.1 Syntax

Usage

- `vcactl`

Output

- `vcactl <command> <filename_param>`
- `vcactl <command> [<card_id>] <filename_param>`
- `vcactl <command> [<card_id><cpu_id>] <filename_param>`

Example

```
# vcactl
Wrong or missing parameters!
Usage:
        vcactl <command> <optional_params>
        vcactl <command> <card_id> <optional_params>
        vcactl <command> <card_id> <cpu_id> <optional_params>
available commands are:
    status : shows status of the cpu
    reset : resets the cpu
    wait : waits for cpu to boot OS
    wait-BIOS : waits for bios to be ready on desired cpu
    boot : boot OS for cpu using LBP
    boot-USB : boot OS for cpu using USB
    update-BIOS : update bios for the cpu
    update-MAC : updates mac address of the cpu with desired value
    script : set script parameter in configuration
    config-show : shows config for cpu
    config : set parameter in configuration
    config-use : restarts ping daemons with new configuration
    config-default : restore vca configuration to default values
    set-SMB-id : set SMB id for card
    temp : read temp from cpu node
    ICMP-watchdog : start/stop ICMP watchdog
EXAMPLE USAGE:
        vcactl reset
        vcactl set-SMB-id 1
        vcactl reset 0 2
        vcactl boot 1 2 /home/centOS7.img
```

7.2 vcactl status

Usage

- `vcactl status [<slotId><cpuid>]`

Description

- vcactl status: Reads status of all CPUs of all cards in the system
- vcactl status <slot id>: Reads status of all CPUs in the card
- vcactl status <slot id> <cpu id>: Reads status of the specified CPU

Result

- LINK_DOWN if CPU link to PLX is down (usually means that PLX eprom is invalid or the card CPU is down due to any reason)
- BIOS_DOWN if CPU BIOS failed upon initialization or still booting (usually when memory training fails such state is reported)
- BIOS_UP if link up and BIOS is ready for handshake
- BIOS_READY if handshake has succeeded w/ BIOS
- BIOS_DONE if BIOS is in temporary state after allocating resources for completing the operation
- FLASHING if BIOS is being updated
- BOOTING if OS is being booted
- OS_READY if OS boot completed
- NET_DEV_READY if Intel® VCA network stack is up
- RESETTING if Intel® VCA CPU is being reset
- ERROR if error occurred

If jumper on card is set for Golden BIOS `vcactl status` will show this information (only in BIOS_UP state). For User BIOS no additional information is displayed.

Example

```
# vcactl status
Card: 0 Cpu: 0 STATE: net_device_ready
Card: 0 Cpu: 1 STATE: net_device_ready
Card: 0 Cpu: 2 STATE: net_device_ready
Card: 1 Cpu: 0 STATE: net_device_ready
Card: 1 Cpu: 1 STATE: net_device_ready
Card: 1 Cpu: 2 STATE: net_device_ready
```

7.3 vcactl reset

Usage

- `vcactl reset [<slotId><cpuId>]`

Description

- Resets the specified CPUs in the system.

Result

- System console via debug port shall show BIOS booting logs and vcactl status shall become BIOS_UP

7.4 vcactl boot

Usage

- `vcactl boot [<slotId><cpuId>] [os_image_file.img]`

Description

- Boots OS from via Leverage Boot Protocol. Image is got from default location (see vcactl config os_image <image>) or from the image file specified in the command. The command requires that the card CPU is in BIOS_UP or BIOS_READY state.

Result

- OS is being booted on card X cpu Y from the image file via PCIe, during that phase BOOTING state is being reported (for short time BIOS_DONE state may be visible also)
- OS_READY state when the command successfully completes

Console outputs upon failure

- ERROR status reported (via vcactl status command)

7.5 vcactl wait

Usage

- `vcactl wait [<slotId><cpuId>]`

Description

- Waits until the OS boot is completed on the specified CPUs.

Result

- Command completes when OS boot is done, otherwise it is blocking. All CPUs shall finish in OS_READY state.

Console outputs upon failure

- ERROR status reported (via vcactl status command)

Example

```
# vcactl reset
# vcactl wait-BIOS
Card: 1 Cpu: 2 - BIOS is up and running!
Card: 0 Cpu: 2 - BIOS is up and running!
Card: 0 Cpu: 1 - BIOS is up and running!
Card: 1 Cpu: 0 - BIOS is up and running!
Card: 0 Cpu: 0 - BIOS is up and running!
Card: 1 Cpu: 1 - BIOS is up and running!
# vcactl boot vca_baremetal_1.0.279.img
# vcactl wait
Card: 1 Cpu: 0 - OS is ready!
Card: 0 Cpu: 1 - OS is ready!
Card: 0 Cpu: 2 - OS is ready!
Card: 0 Cpu: 0 - OS is ready!
Card: 1 Cpu: 1 - OS is ready!
Card: 1 Cpu: 2 - OS is ready!
# vcactl status
Card: 0 Cpu: 0 STATE: net_device_ready
Card: 0 Cpu: 1 STATE: net_device_ready
Card: 0 Cpu: 2 STATE: net_device_ready
Card: 1 Cpu: 0 STATE: net_device_ready
Card: 1 Cpu: 1 STATE: net_device_ready
Card: 1 Cpu: 2 STATE: net_device_ready
```

7.6 vcactl wait-BIOS

Usage

- `vcactl wait-BIOS [<slotId><cpuId>]`

Description

- Waits until the OS boot is completed on the specified CPUs.

Result

- Command completes when BIOS boot is done, otherwise it is blocking. All CPUs shall finish in BIOS_UP or BIOS_READY state.

Console outputs upon failure

- ERROR status reported (via vcactl status command)

7.7 vcactl update-BIOS

Usage

- `vcactl update-BIOS [<slotId><cpuId>]<biosImageFile>`

Description

- Updates BIOS (flash SPI user BIOS partition) on specified CPUs in the system.

Result

- Status of the CPU transitions from BIOS_UP or BIOS_READY to FLASHING and completes with BIOS_READY state. Temporarily during that operation BIOS_DONE might be reported also.
- If jumper on card is set for Golden BIOS `vcactl update-BIOS` will show warning that next booting will be done on Golden BIOS not the updated User BIOS.
  `Card: 0 Cpu: 0 - Warning: Jumper selects GOLD BIOS. Next reboot will boot it.`

Console outputs upon failure

- ERROR status reported (via vcactl status command)

7.8 vcactl temp

Usage

- `vcactl temp [<slotId><cpuId>]`

Description

- Retrieves temperature of one or more cpu's in the system.

Result

- Status of the CPU transitions from BIOS_UP or BIOS_READY to FLASHING and completes with BIOS_READY state. Temporarily during that operation BIOS_DONE might be reported also.

Example

```
# vcactl temp
Card 0 Cpu 0:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +36.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +35.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +35.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +34.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +36.0°C  (high = +105.0°C, crit = +105.0°C)

Card 0 Cpu 1:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +36.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +35.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +34.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +34.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +35.0°C  (high = +105.0°C, crit = +105.0°C)

Card 0 Cpu 2:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +40.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +39.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +38.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +38.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +38.0°C  (high = +105.0°C, crit = +105.0°C)

Card 1 Cpu 0:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +33.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +32.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +32.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +31.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +31.0°C  (high = +105.0°C, crit = +105.0°C)
```

7.9 vcactl config-show

Usage

- `vcactl config-show`

Description

- Shows the current configuration of the vcactld demon

Example

```
# vcactl config-show 0 0
Global configuration:
  auto-boot: 1
  debug-enabled: 0
  link-up-timeout-ms: 2000
  handshake-irq-timeout-ms: 30000
  alloc-timeout-ms: 100
  cmd-timeout-ms: 1000
  mac-write-timeout-ms: 1000
  default-daemon-script:
  wait-cmd-timeout-s: 60
  wait-bios-cmd-timeout-s: 120
  wait-bios-cmd-flashing-s: 1200
  ICMP-ping-inverval-s: 1
  ICMP-response-timeout-s: 10
card 0 cpu 0:
  os-image:
  script: /etc/vca_config.d/vca_00_default.sh
  daemon-script:
  ip: 172.31.1.1
  mask: 24
  gateway: 172.31.1.254
  host-ip: 172.31.1.254
  host-mask: 32
  cpu-max-freq-non-turbo: 0
```

7.10 vcactl update-MAC

Usage

- `vcactl update-MAC <slotId><cpuId> <macAddressCanonical>`

Description

- Updates MAC address (flash SPI GBE partition) on the specified CPU in the system. SlotId and CpuId are mandatory parameters for this command.

Result

- Status of the CPU transitions from BIOS_UP or READY to FLASHING and completes with BIOS_READY state.

Console outputs upon failure:

- ERROR status reported (via vcactl status command)

7.11 vcactl script

Notation

- `vcactl script [<slotId><cpuId>] <scriptFile>`

Description

- Configures bash script for the card CPU to perform Linux user-specific configuration that might be any configuration action, e.g.:
- IP address assignment per virtIO interface or enabling DHCP client
- Starting DomU Guest OS

7.12 vcactl set-SMB-id

Notation:

- `vcactl set-SMB-id <slotId> <smbId>`

Description

- Sets SMB-id of the temp sensor on the card. slodId is 0 to 3, smbId is 0 to

7.13 vcactl clear-SMB-event-log

Notation:

- `vcactl clear-SMB-event-log <slotId> <smbId>`

Description:

- Clears SMB log that can be accessed by vca_elog.py tool on nodes. Node should be in bios-up or bios-ready state.

7.14 vcactl config

Usage

- `vcactl config [<slotId><cpuId>] <parameter-name><parameter-value>`

Description

- Configures vcactld demon with ping intervals and levels, Dom0 and DomU IP address, so that the following parameters are known: <ICMP-ping-interval>, <ICMP-response-timeout>, <LBP-ping-interval>, <LBP-response-timeout>, <os-boot-image>

Available global options

Command	Available settings	Descriptions
auto-boot	1 – to enable 0 – to disable	Enable/disable auto booting
debug-enabled	1 – to enable 0 – to disable	Enable/disbale debug commands
link-up-timeout-ms	# of milliseconds	Set timeout for link-up after reset in milliseconds Default 30000 (30 seconds)
handshake-irq-timeout-ms	# of milliseconds	Set timeout for link-up after reset in milliseconds Default 30000 (30 seconds)
alloc-timeout-ms	# of milliseconds	Set timeout for RAMDISK allocation in BIOS Default 1000 miliseconds
cmd-timeout-ms	# of milliseconds	Set timeout for LBP command reaction in BIOS Default 1000 miliseconds
mac-write-timeout-ms	# of milliseconds	Set timeout for MAC update command processing time in BIOS Default 1000 miliseconds
default-daemon-script	Name and path to daemon script file	Name and path to daemon script file that is executed upon watchdog expiration
wait-cmd-timeout-s	# of seconds	Set timeout for WAIT command in vcactl (time needed to boot OS up) Default 60 seconds
wait-bios-cmd-timeout-s	# of seconds	Set timeout for WAIT-BIOS command in vcactl for BIOS UP time needed Default 60 seconds
wait-bios-cmd-flashing-s	# of seconds	Set timeout for WAIT-BIOS command in vcactl for completion of BIOS update

		Default 600 seconds
ICMP-ping-inverval-s	# of seconds	Set ICMP watchdog period Default is 1 second
ICMP-response-timeout-s	# of seconds	Set ICMP response timeout Default is 1 second

Available per card CPU options:

Command	Available settings	Descriptions
os-image	1 – to enable 0 – to disable	Path to bootable OS image
script	1 – to enable 0 – to disable	Path and script file name that is executed on card after OS is up there /etc/vca_config.d/vca_<slot_id><cpu_id>_default.sh
ip	IP address	Default is 172.31.<slot_id * 3 + cpu_id + 1>.1
mask	Mask length	Default is 24 (means 24 bits)
gateway	IP address	Default is 172.31.<slot_id * 3 + cpu_id + 1>.254
host-ip	IP address	Default is 172.31.<slot_id * 3 + cpu_id + 1>.254
host-mask	Mask length	Default is 24 (means 24 bits)
cpu-max-freq-non-turbo	# of 100MHz	Default value is 17 means 1700MHz

7.15 vcactl config-use

Usage

- `vcactl config-use`

Description

- Triggers vcactld demon to use the last configuration

8. Setting up host for persistent file image

IMPORTANT: After setup is completed, and operating system booted on Intel® VCA nodes, it is important that users do not restart the NFS on the host. This could cause a kernel panic for the nodes booted on the card.

This section describes required configuration of host to use the persistent file image. The setup here is required exactly as it is listed including folder names. Failure to setup exactly as

described will cause the OS to not boot properly and be unusable. See next chapter for information on how to manually set up persistent storage for a bare metal image file.

Example guide to set-up CentOS 7 NFS server or see section chapter on setting up NFS under CentOS Tips and Tricks

https://www.howtoforge.com/nfs-server-and-client-on-centos-7

Configuration steps:

1. Install nfs-utils package:

```
yum install nfs-utils
```

2. Create nfs mount points with node's filesystem (setup for 2 cards):

```
mkdir /mnt/vca_node_00
mkdir /mnt/vca_node_01
mkdir /mnt/vca_node_02
mkdir /mnt/vca_node_10
mkdir /mnt/vca_node_11
mkdir /mnt/vca_node_12
```

The directory, which is used for NFS exports, is hard-coded in persistent FS boot image for card. It consists of '/mnt/' and hostname of the node (vca_node_00, vca_node_01 and so on). So as seen above, full path of exported mountpoint for node 0 on card 0 is /mnt/vca_node_00. The hostname of the node, can be changed, because it is set in /etc/vca_config.d/vca_*_default.sh scripts for each of the node. The /mnt/ prefix cannot be changed.

Assumes (2) Intel® Visual Compute Accelerator card setup. If using more than 2 cards, create additional folders as necessary. If only using 1 card, do not enter last 3 lines above.

3. Create or update /etc/exports with this content:

```
/mnt/vca_node_00 *(rw,sync,no_root_squash,no_all_squash)
/mnt/vca_node_01 *(rw,sync,no_root_squash,no_all_squash)
/mnt/vca_node_02 *(rw,sync,no_root_squash,no_all_squash)
/mnt/vca_node_10 *(rw,sync,no_root_squash,no_all_squash)
/mnt/vca_node_11 *(rw,sync,no_root_squash,no_all_squash)
/mnt/vca_node_12 *(rw,sync,no_root_squash,no_all_squash)
```

Assumes (2) Intel® Visual Compute Accelerator card setup. If using more than 2 cards, create additional folders as necessary. If only using 1 card, do not enter last 3 lines above.

4. Copy contents of vca_rootfs_{build_version}-tree.tar.gz into each mountpoint:

```
tar -xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_00
tar -xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_01
tar -xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_02
tar -xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_10
tar -xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_11
tar -xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_12
```

Assumes (2) Intel® Visual Compute Accelerator card setup. If using more than 2 cards, create additional folders as necessary. If only using 1 card, do not enter last 3 lines above.

5. Start the server:

```
systemctl enable rpcbind
systemctl enable nfs-server
systemctl start rpcbind
systemctl start nfs
```

6. Make sure the firewall is properly configured:

```
firewall-cmd --permanent --zone=public --add-service=nfs
firewall-cmd --reload
```

Or you can disable firewall entirerly:

```
systemctl stop firewalld
systemctl disable firewalld
```

6. Boot the card normally using the vcactl boot <persistent image>

7. Verify net_device_up for each node by using "`vcactl status`" command

9. Manually creating persistent file system using NFS

IMPORTANT: After setup is completed, and operating system booted on Intel® VCA nodes, it is important that users do not restart the NFS on the host. This could cause a kernel panic for the nodes booted on the card.

This section describes how to manually set up persistent file system if using persistent image describe in above section cannot be used. Unlike above section, if following these steps, user will need to mount back to host on each reboot of the card.

Example guide to set-up CentOS 7 NFS server or see section chapter on setting up NFS under CentOS Tips and Tricks

https://www.howtoforge.com/nfs-server-and-client-on-centos-7

Configuration steps:

1. Install nfs-utils package:

```
yum install nfs-utils
```

2. Boot the nodes:

```
vcactl boot vca_baremetal_<version>.img
```

wait for the nodes to be ready "`vcactl status`" should show the nodes status as "net_device_ready"

```
ip a
```

should show ethX network devices for each VV node.

3. Perform the steps in the SW Configuration Guide section pertaining to setting up Passwordless SSH under CentOS Tips and Tricks. Test configuration by issuing following command:

```
for i in 1 2 3 ; do ssh root@172.31.${i}.1 hostname ; done
```

should not prompt for password and does return the hostname for each node.

4. Set up NFS folders, update /etc/exports, and export file system with single command

Note: The following is a helpful script to perform steps 5-7 below. If using this script, skip to step #8. Change "1 2 3" for the # of nodes in the system and "root etc home…" for folders to make persistent.

```
for i in 1 2 3 ; do for j in root etc home opt srv usr var; do mkdir -
p /media/share/node${i}/${j} ; if [ `grep -c /media/share/node${i}
```

```
/etc/exports` -eq "0" ] ; then echo
"/media/share/node${i}      *(no_root_squash,rw)" >> /etc/exports; fi
; if [ `grep -c /media/share/node${i}/${j} /etc/exports` -eq "0" ] ;
then echo "/media/share/node${i}/${j}     *(no_root_squash,rw)" >>
/etc/exports ; fi ; done ; done ; exportfs -a
```

5. Set up NFS shares folders

Note: Ignore this step if above script in step #4 was excecuted.

```
mkdir –p /media/share/node1/root
mkdir –p /media/share/node1/etc
mkdir –p /media/share/node1/home
```

#repeat for each folder you want to make persistent (i.e. root, etc, home, opt, srv, usr, var)

#repeat for each node in the system

6. Create or update /etc/exports with this content:

Note: Ignore this step if above script in step #4 was executed.

```
/media/share/node1 *(rw,sync,no_root_squash,no_all_squash)

/media/share/node1/root *(rw,sync,no_root_squash,no_all_squash)

/media/share/node1/etc *(rw,sync,no_root_squash,no_all_squash)

/media/share/node1/home *(rw,sync,no_root_squash,no_all_squash)
```

#repeat for each folder you want to make persistent (i.e. root, etc, home, opt, srv, usr, var)

#repeat for each node in the system

7. Export file system

Note: Ignore this step if above script in step #4 was executed.

```
exportfs -a
```

8. Copy the directories down from the nodes to the host shares

The following script goes into each node and each folder and copies the content back to the host.

```
for i in 1 2 3 ; do for j in root etc home opt srv usr var ; do ssh
root@172.31.${i}.1 "mount 172.31.${i}.254:/media/share/node${i}   /mnt
; cp -af /${j}   /mnt/ ; umount   /mnt" ; done ; done
```

9. Mount the directories to the nodes (must be done on every reboot)

The following script goes into each node and mounts each folder

```
for i in 1 2 3 ; do for j in root home opt srv usr var etc; do ssh
root@172.31.${i}.1 "cd / ; mount
172.31.${i}.254:/media/share/node${i}/${j} /${j}" ; done ; done
```

10. Verify the shares are mounted (recommended on every reboot)

```
for i in 1 2 3 ; do ssh root@172.31.${i}.1 "mount" ; done
```

Software can now be installed to the nodes normally (note: if using "yum", the steps in the SW Configuration Guide pertaining to setting up a local yum repository under the chapter CentOS Tips and Tricks should be configured here. On each subsequent reboot, steps 9 and 10 are required.

10. Running a sample transcode

1. Insure the file MediaSamples_Linux_6.0.16043175.175.tar.bz2 is on the host system, and "cd" to the directory containing this file.
2. Unpack the samples tarball with the following command:
   ```
   tar jxf MediaSamples_Linux_6.0.16043175.175.tar.bz2
   ```
3. Verify the nodes are fully operational before continuing (see previous sections of the document)
4. Copy the unpacked samples files to a node:
   ```
   scp -r MediaSamples_Linux_6.0.16043175.175 \
   root@172.31.1.1:/root/
   ```
   (Note: to test on more than one node, repeat this command for each subsequent node; e.g., 172.31.2.1, 172.31.3.1, etc…)
5. Connect to the node (this, and the following two steps, should be run together if testing on nodes beyond the first; as before, change the third octet as in the previous step):
   ```
   ssh root@172.31.1.1
   ```
6. Change into the correct directory:
   ```
   cd MediaSamples_Linux_6.0.16043175.175
   ```
7. Run the sample transcode:
   ```
   ./sample_multi_transcode_drm -i::h264
   content/test_stream.264 -o::h264 out.h264
   ```
8. Transcode should run without error. Should take less than 0.1 seconds to transcode.

11. Updating BIOS

1. Move the USER BIOS Update jumper for the appropriate node(s) to the shorted position (jumper across both pins, instead of on only one). Jumpers are at the front of the board, near the 8-pin auxiliary power connector.
2. Power on the host.
3. Reset the nodes (*Note: this assumes the host system has already been logged into with the "root" user account)*:

   a. `vcactl reset`
      # Wait for reset completion; use either
      `vcactl wait-BIOS`
      # or
      `vcactl status`
      # and wait for the node status to return to "bios_up"

   b. `vcactl update-BIOS <bios_img_filename>`
      # use the .img BIOS file for VCA-BIOS-<bios_img_filename>.img. Wait for update to complete; use either
      `vcactl wait-BIOS`
      # or
      `vcactl status`
      # and wait for the node status to return to "bios_up"

   c. After BIOS update is complete, the node(s) will reboot to come online with the new BIOS version.

12. Reading Intel® VCA events with vca_elog.py script

The vca_elog Python script prints SMBIOS Type 15 events. These events are logged by BIOS in its NVRAM area. They can be viewed in BIOS “Event Log”, or by this script.

By default this tool displays only error events.

12.1 Prerequisites

To run script, on system image a Python interpreter should be installed. This has already been done on all Intel® VCA Reference OS images.

The script is mainly designed to run on Intel® VCA nodes. It can be also run on other computers to offline interpret previous uploaded NVRAM area (use –i option).

To install a Python interpreter on CentOS computer, invoke the following command:

```
yum install python
```

12.2 Script usage:

```
vca_elog.py [-h] [-e | -a | -t TYPE [TYPE ...] | -p] [-s] [-d
DOWNLOAD_LOG | -i INPUT_FILE] [--debug] [-v]
```

Optional arguments:

```
-h, --help              show this help message and exit
-e, --errors            print error events only
-a, --all               print all events
-t TYPE [TYPE ...], --types TYPE [TYPE ...]
                        print specified events only
-p, --print_types       print event types with associated numbers for
use with --types option
-s, --statistics        print events statistics
-d DOWNLOAD_LOG, --download_log DOWNLOAD_LOG
                        download log data from NVRAM to binary file
-i INPUT_FILE, --input_file INPUT_FILE
                        parse file downloaded with download_log option
--debug                 print additional debug logs
-v, --version           print version
```

To print this help call:

```
vca_elog.py -h
```

12.3 Executing the script

When script is installed on image users can call by:

vca_elog.py [options]

When you copy it:

```
./path_to_vca_elog/vca_elog.py [options]
```

or

```
Python ./path_to_vca_elog/vca_elog.py [options]
```

12.4 Displaing events

```
vca_elog.py
```

or

```
vca_elog.py –e
```

This prints errors only. Example output is the following:

```
Single Bit ECC Memory Error
```

```
   Date: 2015-07-29 11:35:49
   S1:C1:D0
Single Bit ECC Memory Error
   Date: 2015-07-29 11:36:08
   S1:C0:D1
Single Bit ECC Memory Error
   Date: 2015-07-29 11:36:14
   S1:C1:D1
...
Other event filters, used interchangeably with -e are –a and –t TYPE
[TYPE..]
vca_elog.py –e
vca_elog.py –t 1 2 0x17
-e version prints all events
-t 1 2 0x17 variant prints only events with specified IDs
Example output:
System Boot
   Date: 2015-07-29 10:07:49
OEM0
   Date: 2015-07-29 10:07:49
OEM0
   Date: 2015-07-29 10:07:49
System Boot
   Date: 2015-07-29 11:30:41
Single Bit ECC Memory Error
   Date: 2015-07-29 11:35:49
   S1:C1:D0
Single Bit ECC Memory Error
   Date: 2015-07-29 11:36:08
   S1:C0:D1
Single Bit ECC Memory Error
   Date: 2015-07-29 11:36:14
   S1:C1:D1
...
```

Each event report has two or three lines (current version) and generally they are formatted in the following way:

```
1.    name error/event
2.    Date: date/no valid date
3.    [data]
First line is a name of the event and may be one of the following
strings:
1.    Single Bit ECC Memory Error
2.    Multi Bit ECC Memory Error
3.    Parity Memory Error
4.    Bus Time Out
5.    I/O Channel Check
6.    Software NMI
```

```
7.    POST Memory Resize
8.    POST Errors
9.    PCI Parity Error
10.   PCI System Error
11.   CPU Failure
12.   EISA Failsafe Timer Timeout
13.   Correctable Memory Log Disabled
14.   Logging Disabled for Event Type
15.   System Limit Exceeded
16.   Asyn HW Timer Expired
17.   System Configuration Information
18.   Hard Disk Information
19.   System Reconfigured
20.   Uncorrectable CPU Complex Error
21.   Log Area Reset
22.   System Boot
23.   OEM0
24.   OEM1
25.   OEM2
```

Second line is a date of the event. The date is printed in format: YYYY-MM-RR HH:MM:SS or no valid date is printed when the date was not set properly.

Last line is a custom data of the event. This version of software prints only data for event types 1 and 2

To identify which ids can be used with –t options, switch –p may be used. Output for –p option is:

12.5 Types of events:

Name	Type's ID dec	hex	Class
Single Bit ECC Memory Error	1	0x01	error
Multi Bit ECC Memory Error	2	0x02	error
Parity Memory Error	3	0x03	error
Bus Time Out	4	0x04	error
I/O Channel Check	5	0x05	status
Software NMI	6	0x06	status
POST Memory Resize	7	0x07	status
POST Errors	8	0x08	error
PCI Parity Error	9	0x09	error
PCI System Error	10	0x0a	error
CPU Failure	11	0x0b	error
EISA Failsafe Timer Timeout	12	0x0c	error
Correctable Memory Log Disabled	13	0x0d	status
Logging Disabled for Event Type	14	0x0e	status
System Limit Exceeded	16	0x10	status

```
 Asyn HW Timer Expired                17  0x11     status
 System Configuration Information     18  0x12     status
 Hard Disk Information                19  0x13     status
 System Reconfigured                  20  0x14     status
 Uncorrectable CPU Complex Error      21  0x15     error
 Log Area Reset                       22  0x16     status
 System Boot                          23  0x17     status
 OEM0                                224  0xe0     oem
 OEM1                                225  0xe1     oem
 OEM2                                226  0xe2     oem
```

12.6 Displaying statistics

```
vca_elog.py –s
```

and example output is:

STATISTICS:

```
Nb of log events: 279
Events statistics:
                                 Type's ID
 Name                            dec   hex     Class          Nb
-------------------------------------------------------------------
 Single Bit ECC Memory Error       1  0x01     error         266
 Multi Bit ECC Memory Error        2  0x02     error           0
 Parity Memory Error               3  0x03     error           0
 Bus Time Out                      4  0x04     error           0
 I/O Channel Check                 5  0x05     status          0
 Software NMI                      6  0x06     status          0
 POST Memory Resize                7  0x07     status          0
 POST Errors                       8  0x08     error           0
 PCI Parity Error                  9  0x09     error           0
 PCI System Error                 10  0x0a     error           0
 CPU Failure                      11  0x0b     error           0
 EISA Failsafe Timer Timeout      12  0x0c     error           0
 Correctable Memory Log Disabled  13  0x0d     status          0
 Logging Disabled for Event Type  14  0x0e     status          0
 System Limit Exceeded            16  0x10     status          0
 Asyn HW Timer Expired            17  0x11     status          0
 System Configuration Information 18  0x12     status          0
 Hard Disk Information            19  0x13     status          0
 System Reconfigured              20  0x14     status          0
 Uncorrectable CPU Complex Error  21  0x15     error           0
 Log Area Reset                   22  0x16     status          0
 System Boot                      23  0x17     status          3
 OEM0                            224  0xe0     oem            10
 OEM1                            225  0xe1     oem             0
 OEM2                            226  0xe2     oem             0
```

To print event report with statistics, –e, –a and –t options may be used.

12.7 Saving to a file to be parsed later

To upload event area from the NVRAM to file invoke:

```
vca_elog.py –u nvram_log15.bin
```

The script shall confirm properly realized dump with the following information:

```
Log was uploaded successfully to file nvram_log15.bin
```

When the tool cannot save the area to file, the following information is displayed:

```
Cannot open output binary file /tests/$$$/n$$vram_lo%g15.bin
```

The option is useful for reporting issues to a support team.

12.8 Parsing previously stored files

```
vca_elog.py –i nvram_log15.bin
```

By default, only errors are printed. To print all events or specified types, use options -a or –t.

13. Creating custom OS images

13.1 Overview

The Intel® Visual Computing Accelerator (VCA)card provisions the OS to the card CPU over the PCIe bus. The VCA's BIOS exposes a memory buffer, to which the host copies the selected OS image. The following components are copied to each node when boot is requested:

- EFI bootloader
- Kernel image
- InitRAMFS image

InitRAMFs is responsible for early system configuration, mounting the main Root File System and root pivoting to this mounted RootFS. In the VCA environment, there are two strategies for provisioning RootFS:

- Volatile OS file system, in which the RootFS image is embedded in the InitRamFS image. InitRamFS is responsible for extracting the RootFS and mounting it onto a RAM disk.
- Persistent OS file system, in which the RootFS image is provisioned over NFS. InitRamFS establishes virtual network over PCIe, mounts a remote NFS share and switches to that remote root directory.

13.2 Required Software

To create a custom image, the livecd-creator toolset is needed. It can be downloaded from:

http://people.centos.org/arrfab/CentOS7/LiveMedia/RPMS/

Download the following files:

- hfsplus-tools-540.1.linux3-4.el7.x86_64.rpm
- python-imgcreate-20.1-2.el7.x86_64.rpm
- livecd-tools-20.1-2.el7.x86_64.rpm

And install them using the following commands

```
sudo yum -y install syslinux-extlinux
sudo yum -y install ./hfsplus-tools-540.1.linux3-4.el7.x86_64.rpm
./python-imgcreate-20.1-2.el7.x86_64.rpm ./livecd-tools-20.1-
2.el7.x86_64.rpm
```

Download the following file:

sl-70-livecd.ks from https://svn.iac.ethz.ch/websvn/pub/websvn-pub/wsvn/livecd/trunk/SL7/livecd-config/?#a7a147e8b489ecfcf0a3ce076d8c002fb and rename it to vca_iso.ks.

13.3 Local yum CentOS repository

For purposes of automated image building, we don't recommend using an external CentOS repository, as it

- Increases build time (few hundred megabytes needs to be downloaded each time from internet)
- Leads to inconsistent package versions between image builds
- Requires a machine with consistent and not-limited access to Internet

Instead, local repositories shall be used. The instruction assumes their location at:

/usr/lib/vca/vca_repos/local_repo/

The current kick starter files (image creation recipes) uses the following local repositories:

Name	Location	Description
localrepo	/usr/lib/vca/vca_repos/local_repo/	Local version of CentOS repository, with official packages only
vca_extras	/usr/lib/vca/vca_repos/extras_repo	Additional packages, out of CentOS repository, not created by development team or changing very rarely
vca	/usr/lib/vca/vca_repos/vca_repo	Often (each build) changing packages, delivered by development team

Such repositories may be physically located on a local hard disk, or be a link to nfs repository at some other location.

If the directions earlier in the document for setting up a local yum repository have been followed, please run the following commands to create the required directory structure and mount the DVD ISO to the "local_repo" directory:

```
sudo mkdir –p /usr/lib/vca/vca_repos/{local_repo,vca_extras,vca}
sudo mount --bind /var/www/html/centos71 \
        /usr/lib/vca/vca_repos/local_repo
```

13.3.1 Create a local extras repository

The following packages and other third party packages shall be copied to a second repository, which shall be named “vca_extras” and is intended for packages with a slow change cycle (including packages produced by both Intel and third parties). Sample packages for the repo may be:

Validation tools:

- iperf
- iperf3
- fio

Intel MediaSDK:

- intel-linux-media
- intel-opencl-1.2
- intel-opencl-1.2-devel

And Xen packages:

- xen
- xen-hypervisor
- xen-runtime

Copy all desired rpm files (packages) to the directory /usr/lib/vca/vca_repos/vca_extras. Once all desired rpm files are in the directory, create the repository with the following command:

```
sudo createrepo /usr/lib/vca/vca_repos/vca_extras
```

13.3.2 Create a local modules repository

The third repository contains essential VCA packages, changing with almost each build. Therefore the repository needs to be updated very often. It contains:

- vca kernel (kernel RPM file)

- vca modules (daemon-vca RPM file)
- vca software stack (vcass RPM file)

If the directory already exists and contains some older packages, remove them all

```
sudo rm -rf /usr/lib/vca/vca_repos/vca/*
```

Copy all desired rpm files (packages) to the directory /usr/lib/vca/vca_repos/vca. Once all desired rpm files are in the directory, create the repository with the following command:

```
sudo createrepo /usr/lib/vca/vca_repos/vca
```

13.3.3 Prepare kickstart file

The kickstart file is an answer file for the for image-creator tool (part of livecd-tools package), describing how the Root FS image shall be built. Particularly, it contains a disk layout, list of packages to be installed and list of post-installation activities.

Make the following modifications to the vca_iso.ks file:

- Disable SELinux.
  - Replace “selinux --enforcing” with “selinux --disabled”.
- Limit filesystem size.
  - Replace 8192 in “part / --size 8192 --fstype ext4” with 1500.
- Enable autostart of network and SSH daemon services. Note there are no spaces between the double-hyphen and the option names (enabled, disabled) in this line.
  - Replace “services --enabled=NetworkManager --disabled=network,sshd” with “services --enabled=NetworkManager,network,sshd”
- Modify root password to “vista1”. Replace rootpw line with the following:
  - rootpw --iscrypted $6$0f2rNPn2oON$lO5Dh/Cb7s0H1GR./PZPPYT.xjJPTdIA.M.quAtA8fpMjRo.0rsGJmm8Zq83rOY0xnkzoWQAPhCtu5sj8ztgw1
- Delete all 3 defined repositories (lines starting with “repo”). Add the following ones (noting again that the -- must not be followed by a space (--baserepo) in each line below):
  - repo --name=localrepo --baseurl=file:/usr/lib/vca/vca_repos/local_repo
  - repo --name=vv --baseurl=file:/usr/lib/vca/vca_repos/vca_repo
  - repo --name=vv_extras --baseurl=file:/usr/lib/vca/vca_repos/extras_repo
- Delete all packages below the line reading “%packages. Add the following:
  - kernel-3.10*
  - vcass-modules-3*
  - intel-linux-media
  - intel-opencl-1.2
  - intel-opencl-1.2-devel
- Add a selection of desired tools, utilities, etc.. Sample list is the following:
  - firewalld

  - lrzsz
  - net-tools
  - pciutils
  - iptraf-ng
  - tcpdump
  - redhat-lsb-core
  - vim
  - mc
  - wget
  - mcelog
  - openssh-server
  - dhclient
  - yum
  - openssh-clients
  - rpm
  - nmap
  - lsscsi
  - lm_sensors
  - sysstat
  - nfs-utils
- Search for the line "%post". At the end of the section, before the line “%end”, add the following line, required to configure soft dependencies between VCA modules:
  `/sbin/vca_setup.sh card`

13.3.4 Create a custom Dracut module

In order to mount embedded Root File System on RAM disk at the moment of OS start, an additional Dracut module needs to be used. To provide such a module, create the following local directory:

```
mkdir –p dracut_module/50mountloopdev/
```

In the directory create two files, with the following content.
The first, module-setup.sh:

```
#!/bin/bash

check() {
        return 0
}

install() {
        inst_hook cmdline 20 "$moddir/mount_loop_device.sh"
}

depends () {
        return 0
}
```

```
Second file, module-setup.sh:
#!/bin/bash

modprobe loop
modprobe ext4
losetup /dev/loop0 /root/root_partition.img
```

13.4 Volatile OS file system - InitRamFs with embedded RootFs image

13.4.1 Create Root FS image

Using image-creator tool and previously created kickstart file, generate RootFS image, containing a full Linux installation:

sudo image-creator vca_iso.ks –n vca_rootfs

13.4.2 Create Init RAM FS image

Mount the RootFS image, so kernel modules can be used in initramfs

```
mkdir -p mounted_rootfs
sudo mount vca_rootfs.img mounted_rootfs
```

To find kernel version name invoke:

```
ls mounted_rootfs/boot/vmlinuz*
```

As an output, a single file shall be listed, for example:

```
mounted_rootfs/boot/vmlinuz-<version>.VCA
```

In this case, version is 3.10.0-1.1.0.79.VCA. Save it within KERNEL_NAME variable:

```
export KERNEL_NAME=<version>.VCA
```

Generate initramfs image:

```
sudo dracut -v --omit-drivers "plx87xx plx87xx_dma" \
--add-drivers "loop ext4 pl2303" --add mountloopdev \
--include `pwd`/vca_rootfs.img \
/root/root_partition.img `pwd`/vca_initramfs.img \
$KERNEL_NAME -f -k `pwd`/mounted_rootfs/lib/modules/$KERNEL_NAME
```

Remove custom dracut module from Dracut directory

```
rm -rf /lib/dracut/modules.d/50mountloopdev
```

13.4.3 Generate partition image

1. The initramfs image needs to be embedded in a FAT partition to allow the card's EFI to access the files on the image from the EFI bootloader. Use the following command to identify the size of the generated image, counted in disk blocks (note the first parameter is a lower-case "L", not a numeral "1"):

```
export COUNTED_SIZE=`ls vca_initramfs.img -l --block-size=512 \
 | awk '{ print $5 }'`
```

2. Add 102400 to the size and create an empty file:

```
dd if=/dev/zero of=vca_baremetal.bin \
count=$(( $COUNTED_SIZE + 102400 ))
```

3. Create FAT filesystem at the empty file and mount it:

```
mkdir mounted_isoimg
LOOPDEV=`sudo losetup -f --show vca_baremetal.bin`
mkfs.fat $LOOPDEV
sudo mount vca_baremetal.bin mounted_isoimg
```

4. Copy Init RAM FS and kernel to the partition image:

```
cp mounted_rootfs/boot/vmlinuz* mounted_isoimg
cp vca_initramfs.img mounted_isoimg
```

13.4.4 Add EFI bootloader to partition image

Mount a partition image, obtained from build, at /mnt

```
sudo mount <image from build> /mnt
```

Copy contents of efi directory to newly created partition image:

```
  cp –r /mnt/efi mounted_isoimg
```

List the content of mounted_isoimg/efi/BOOT/syslinux.cfg file. If other kernel version was used in the current building process, update the version in the file
Unmount partition image from the build

```
sudo umount /mnt
```

13.4.5 Cleanup

Unmount all resources

```
sudo umount mounted_isoimg
sudo umount mounted_rootfs
```

Copy vca_baremetal.bin to host of the system, where it is intended to be used.

13.5 Persistent OS file system - InitRamFs with RootFs mounted using NFS

13.5.1 Modify network Dracut module

In order to mount OS RootFS using NFS, the Dracut modules have to be modified. There are two files in 40network module (in CentOS, dracut-network package need to be installed): ifup.sh, parse-ip-opts.sh. Files are in /usr/lib/dracut/modules.d/40network
Below are unified diffs of changes which must be made:

```
--- a\parse-ip-opts.sh 2015-06-17 12:48:26.000000000 +0200
```

```
+++ b\parse-ip-opts.sh 2015-06-02 10:44:15.000000000 +0200
@@ -74,12 +74,13 @@
                  [ -z "$ip" ] && \
                      die "For argument 'ip=$p'\nValue '$autoopt'
without static configuration does not make sense"
                  [ -z "$mask" ] && \
                      die "Sorry, automatic calculation of netmask
is not yet supported"
                  ;;
              auto6);;
+             lbp);;
              dhcp|dhcp6|on|any) \
                  [ -n "$NEEDBOOTDEV" ] && [ -z "$dev" ] && \
                      die "Sorry, 'ip=$p' does not make sense for
multiple interface configurations"
                  [ -n "$ip" ] && \
                      die "For argument 'ip=$p'\nSorry, setting
client-ip does not make sense for '$autoopt'"
                  ;;

--- a\ifup.sh   2015-06-17 12:48:26.000000000 +0200
+++ b\ifup.sh   2015-06-02 10:44:15.000000000 +0200
@@ -353,12 +353,19 @@
                  do_dhcp -4 ;;
              dhcp6)
                  load_ipv6
                  do_dhcp -6 ;;
              auto6)
                  do_ipv6auto ;;
+             lbp)
+                 while [ ! -e "/sys/kernel/plx87xx/net_config" ];
do sleep 1; done
+                 . "/sys/kernel/plx87xx/net_config"
+                 [ -e "/sys/kernel/plx87xx/sys_config" ] && {
+                     . "/sys/kernel/plx87xx/sys_config"
+                 }
+                 ;;
              *)
                  do_static ;;
          esac
      done

      > /tmp/net.${netif}.up
```

13.5.2 Create Root FS image

Using image-creator tool and previously created kickstarter file, generate RootFS image, containing whole Linux installation:

```
image-creator vca_iso.ks –n vca_rootfs
```

Mount the RootFs image, so kernel modules can be used in initramfs

```
mkdir -p mounted_rootfs
sudo mount vca_rootfs.img mounted_rootfs
```

Create the rootfs archive for NFS mountpoint on host for nodes

```
tar -zcf rootfs-tree.tar.gz -C mounted_rootfs
```

13.5.3 Create InitRAMFS image

To find kernel version name invoke:

```
ls mounted_rootfs/boot/vmlinuz*
```

As an output, a single file shall be listed, for example:

```
mounted_rootfs/boot/vmlinuz-<version>.VCA
```

In this case, version is 3.10.0-1.1.0.79.VCA. Save it within KERNEL_NAME variable:

```
export KERNEL_NAME=<version>.VCA
```

Prepare vca.conf and blacklist files:

```
mkdir persistent_files
cat > persisitent_files/vca.conf << EOF
softdep plx87xx pre: vop
EOF
cat > persistent_files/blacklist << EOF
blacklist vca_csm
blacklist vca_mgr
EOF
```

Generate initramfs image:

```
dracut -v -f \
  --add 'nfs' \
  --add-drivers 'nfs nfsv3 nfsv4 virtio_net virtio_console loop
ext4 plx87xx plx87xx_dma vop vop_bus' \
  --include `pwd`/persistent_files/blacklist
/etc/modprobe.d/blacklist \
  --include `pwd`/persistent_files/vca.conf
/etc/modprobe.d/vca.conf \
  --prefix `pwd`/mounted_rootfs \
  --kmoddir `pwd`/mounted_rootfs/lib/modules/$KERNEL_NAME \
  `pwd`/vca_initramfs.img \
  $KERNEL_NAME
```

13.5.4 Generate partition image

1. The initramfs image needs to be embedded in a FAT partition to allow the card's EFI to access the files on the image from the EFI bootloader. Use the following command to identify the size of the generated image, counted in disk blocks (note the first parameter is a lower-case "L", not a numeral "1"):

```
export COUNTED_SIZE=`ls vca_initramfs.img -l --block-size=512 \
 | awk '{ print $5 }'`
```

2. Add 102400 to the size and create an empty file:

```
dd if=/dev/zero of=vca_baremetal.bin \
count=$(( $COUNTED_SIZE + 102400 ))
```

3. Create FAT filesystem at the empty file and mount it:

```
mkdir mounted_isoimg
LOOPDEV=`sudo losetup -f --show vca_baremetal.bin`
mkfs.fat $LOOPDEV
sudo mount vca_baremetal.bin mounted_isoimg
```

4. Copy Init RAM FS and kernel to the partition image:

```
cp mounted_rootfs/boot/vmlinuz* mounted_isoimg
cp vca_initramfs.img mounted_isoimg
```

13.5.5 Add EFI bootloader to partition image

- Mount a partition image, obtained from build, at /mnt
  sudo mount <image from build> /mnt

- Copy content of efi directory to newly created partition image:
  `cp –r /mnt/efi mounted_isoimg`
- List the content of mounted_isoimg/efi/BOOT/syslinux.cfg file. If other kernel version was used in the current building process, update the version in the file.
- Modify kernel command line in syslinux.cfg file. Example contents of syslinux.cfg:

```
DEFAULT vv_image
LABEL vv_image
    MENU LABEL VCA volatile image
    LINUX ../../vmlinuz-KERNEL_VERSION
    APPEND ip=eth0:lbp root=nfs:host:/mnt/%s
console=ttyS0,115200n8
    INITRD ../../vca_initramfs.img
```

  Where *KERNEL_VERSION* is current version of kernel used (e.g. 3.10.0-1.1.0.79.VCA).
  ip=eth0:lbp - use default network interface (in case of VCA card – the virtualIO interface) as rootfs source. Address and other network settings are set by VCA drivers.
  root=nfs:host:/mnt/%s – root is mounted using NFS protocol, from the server named ‘host’ and the path to the mountpoint is /mnt/{hostname of the current node}. Hostname of the current node and address of the ‘host’ server is set by vcactl application during boot of the node.
- Unmount partition image from the build
  `sudo umount /mnt`

13.5.6 Host preparation for persistent FS boot

It is assumed that CentOS 7.1 is installed on the host and the following instructions are for this system.

2. ***Install nfs-utils package:***
   ```
   yum install nfs-utils
   ```
3. ***Create nfs mount points with node's filesystem (setup for 2 cards):***
   mkdir /mnt/vca_node_00
   mkdir /mnt/vca_node_01
   mkdir /mnt/vca_node_02
   mkdir /mnt/vca_node_10
   mkdir /mnt/vca_node_11
   mkdir /mnt/vca_node_12

   The directory, which is used for NFS exports, is hard-coded in persistent FS boot image for card. It consists of '/mnt/' and hostname of the node (vca_node_00, vca_node_01 and so on). So as seen above, full path of exported mount point for node 0 on card 0 is /mnt/vca_node_00. The hostname of the node, can be changed, because it is set in /etc/vca_config.d/vca_*_default.sh scripts for each of the node. The/mnt/ prefix cannot be changed in current implementation.
4. ***Create /etc/exports with this content:***
   ```
   /mnt/vca_node_00
   172.31.0.0/255.255.0.0(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_01
   172.31.0.0/255.255.0.0(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_02
   172.31.0.0/255.255.0.0(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_10
   172.31.0.0/255.255.0.0(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_11
   172.31.0.0/255.255.0.0(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_12
   172.31.0.0/255.255.0.0(rw,sync,no_root_squash,no_all_squash)
   ```
5. ***Extract contents of vca_rootfs_{build_version}-tree.tar.gz (OS tree archive from SW release) into each mount point:***
   ```
   tar -xf vca_rootfs_<version>-tree.tar.gz -C /mnt/vca_node_00
   tar -xf vca_rootfs_<version>-tree.tar.gz -C /mnt/vca_node_01
   tar -xf vca_rootfs_<version>-tree.tar.gz -C /mnt/vca_node_02
   tar -xf vca_rootfs_<version>-tree.tar.gz -C /mnt/vca_node_10
   tar -xf vca_rootfs_<version>-tree.tar.gz -C /mnt/vca_node_11
   tar -xf vca_rootfs_<version>-tree.tar.gz -C /mnt/vca_node_12
   ```
6. ***Start the server:***
   ```
   sudo systemctl enable rpcbind
   sudo systemctl enable nfs-server
   sudo systemctl start rpcbind
   sudo systemctl start nfs
   ```
7. ***Make sure the firewall is properly configured:***
   ```
   sudo firewall-cmd --permanent --zone=public --add-service=nfs
   sudo firewall-cmd --reload
   ```

14. CentOS Tips and Tricks

This section goes through several useful CentOS configurations to assist the user in setting up optional components on the Intel® Visual Compute Accelerator. Scripts have been written below to save time. These scripts are not supported nor guaranteed to work and may stop working on future versions. Intel will make a best effort attempt to keep this section up to date.

14.1 Installing CentOS

1. Connect CentOS 7.1 install media to server
2. Boot server; if necessary, select install media as boot device
3. At the "CentOS 7" boot menu, select "Install CentOS 7" (up-arrow key, or press <I>), then press <Enter>; installer will load, which will take a couple of moments.
4. At the "WELCOME TO CENTOS 7" menu, select the language (and dialect, if appropriate), then click the "Continue" button in the lower right corner of the screen.
5. At the "INSTALLATION SUMMARY" menu, click "INSTALLATION DESTINATION"
    a. At the "INSTALLATION DESTINATION" screen:
        i. Below "Device Selection", click the 240GB host OS drive.
        ii. Below "Other Storage Options" -> "Partitioning", select "I will configure partitioning"
        iii. Click the "Done" button in the upper left corner of the screen
    b. At the "MANUAL PARTITIONING" screen:
        i. Click the "Click here to create them automatically" link
        ii. Click the "/home" partition at the top of the left side
        iii. Click the "-" button at the bottom of the left side
        iv. Click the "/" partition
        v. On the right side, delete the contents of the "Desired Capacity" text box
        vi. Click the "Update Settings" button in the middle of the right side of the screen.
        vii. Click the "Done" button in the upper left corner of the screen
        viii. At the "SUMMARY OF CHANGES" dialog box, click the "Accept Changes" button in the lower right corner of the dialog box
6. At the "INSTALLATION SUMMARY" screen, click "Software Selection"
    a. From the left pane, "Base Environment", select
        "Server with GUI"
    b. From the right pane, "Add-Ons for Selected Environment", select
        "File and Storage Server"
        "Performance Tools" and
        "Development Tools"
    c. Click the "Done" button in the upper left corner of the screen
7. At the "INSTALLATION SUMMARY" screen, click "NETWORK & HOSTNAME"

   a. Select the connected (*Note: disconnected network devices will be marked "unplugged"*) network device (e.g., "Ethernet (enp3s0f0)")
   b. Click the slider button "Off" so that it moves to "On"
   c. Enter a system name in the "Hostname" text box in the lower left corner of the screen
   d. Click the "Done" button in the upper left corner of the screen
8. At the "INSTALLATION SUMMARY" screen, click the "Begin Installation" button in the lower right corner of the screen.
9. At the "CONFIGURATION" screen, click "ROOT PASSWORD"
   a. Enter a secure password in both the "Root Password" and "Confirm" text boxes. (*Note: pre-loaded test image uses the insecure password: vista1*)
   b. Click the "Done" button in the upper left corner of the screen (*Note: insecure passwords require the "Done" button be clicked twice.*)
10. At the "CONFIGURATION" screen, click "USER CREATION"
   a. Enter the user's "Full name" and "Username" in the appropriate text boxes.
   b. If the user should be configured with SUDO access at creation, check the "Make this user administrator" checkbox
   c. Enter a secure password in both the "Password" and "Confirm password" text boxes.
   (*Note: pre-loaded test image uses the username: vista and the insecure password: vista1 and the user is configured for sudo use.*)
   d. Click the "Done" button in the upper left corner of the screen (*Note: insecure passwords require the "Done" button be clicked twice.*)
11. Once the installation completes, a "Reboot" button will be displayed in the lower right corner of the screen. When that is displayed, click the "Reboot" button, then disconnect the installation media from the server when the POST screen is displayed.
12. When the OS loads, an "INITIAL SETUP" screen will be displayed.
13. At the "INITIAL SETUP" screen, click "LICENSE INFORMATION" (below "LOCALIZATION")
   a. At the "License Agreement" screen, click the checkbox next to "I accept the license agreement"
   b. Click the "Done" button in the upper left corner of the screen.
14. At the "INITIAL SETUP" screen, click the "FINISH CONFIGURATION" button in the lower right corner of the screen.
15. At the "Kdump" screen, click the "Forward" button in the lower right corner of the screen.

14.2 SSH Password-less configuration

1. (Complete one time per host): Create SSH password-less files

```
cd
sudo ssh-keygen
# press <Enter> at each prompt until the system returns to the
# command prompt
sudo cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys
```

```
sudo chmod 600 /root/.ssh/authorized_keys
sudo cp -af /root/.ssh ~/
sudo chown -R vista:vista ~/.ssh
```

2. (Complete each time volatile nodes boot, if not using persistent filesystem through NFS): Copy the .ssh directory to each Intel® Visual Compute Accelerator node. At the "Are you sure you want to continue connecting" prompt, enter "yes", then press <Enter>. At the password prompt, enter the password "vista1" (this is the password for the root user on the Intel® Visual Compute Accelerator node). The following command will copy the directory to all three nodes; enter the password at each prompt.
(*Note: the quote before "ip a" and after "-l" is a back-tick; on an en_US keyboard, it is the key left of number 1 on the number row, an unshifted ~*)

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
scp -r .ssh root@172.31.${i}.1:/root/ ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
scp -r .ssh root@172.31.1.1:/root/
scp -r .ssh root@172.31.2.1:/root/
scp -r .ssh root@172.31.3.1:/root/
scp -r .ssh root@172.31.4.1:/root/
scp -r .ssh root@172.31.5.1:/root/
scp -r .ssh root@172.31.6.1:/root/
```

14.3 Setting up hostnames

1. Issue the following command to set the hostname on each Intel® Visual Compute Accelerator node:

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "hostname node${i}.localdomain ; echo
node${i}.localdomain > /etc/hostname" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "hostname node1.localdomain ; echo
node1.localdomain > /etc/hostname"
ssh root@172.31.2.1 "hostname node1.localdomain ; echo
node1.localdomain > /etc/hostname"
ssh root@172.31.3.1 "hostname node1.localdomain ; echo
node1.localdomain > /etc/hostname"
ssh root@172.31.4.1 "hostname node1.localdomain ; echo
node1.localdomain > /etc/hostname"
ssh root@172.31.5.1 "hostname node1.localdomain ; echo
node1.localdomain > /etc/hostname"
ssh root@172.31.6.1 "hostname node1.localdomain ; echo
node1.localdomain > /etc/hostname"
```

14.4 Creating a local YUM Repository

1. Copy the CentOS 7 ISO to the system.

2. Issue the following command to mount the ISO to /media: (*Note: change the name CentOS-7.iso to the iso filename on the system.*)

```
sudo mount -o loop CentOS-7.iso /media
```

3. Issue the following commands to install the web server, start the service, and open the firewall:

```
ssh root@localhost 'echo [localbase] >
/etc/yum.repos.d/base.repo'
ssh root@localhost 'echo name=localbase >>
/etc/yum.repos.d/base.repo'
ssh root@localhost 'echo baseurl=file:///media >>
/etc/yum.repos.d/base.repo'
ssh root@localhost 'echo gpgcheck=0 >>
/etc/yum.repos.d/base.repo'
sudo yum -y install --disablerepo=* --enablerepo=localbase
httpd
sudo systemctl enable httpd
sudo systemctl start httpd
for i in public external dmz work home internal trusted; do
sudo firewall-cmd --zone=$i --add-service=http --permanent
; done
sudo firewall-cmd --reload
```

4. Issue the following commands to remount the ISO to the web server file structure, and mount it on reboots (*Note: the last command in this step is currently causing the system to fail boot; please see the next step for workaround*):

```
sudo umount /media
sudo mkdir /var/www/html/centos70
sudo mount -o loop CentOS-7.iso /var/www/html/centos70
ssh root@localhost "echo mount -o loop
/home/vista/vvgold/CentOS-7.iso  /var/www/html/centos70" >>
/etc/rc.local
```

5. Modify the local repo file to point to the new location under /var/www/html/centos70:

```
sudo sed -i -e s=media=var/www/html/centos70=
/etc/yum.repos.d/base.repo
```

6. Make sure any repos previously loaded on the nodes are moved somewhere safe

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "mv /etc/yum.repos.d/* /home" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "mv /etc/yum.repos.d/* /home"
ssh root@172.31.2.1 "mv /etc/yum.repos.d/* /home"
ssh root@172.31.3.1 "mv /etc/yum.repos.d/* /home"
ssh root@172.31.4.1 "mv /etc/yum.repos.d/* /home"
ssh root@172.31.5.1 "mv /etc/yum.repos.d/* /home"
ssh root@172.31.6.1 "mv /etc/yum.repos.d/* /home"
```

7. Create a repository file to copy to the Intel® Visual Compute Accelerator nodes:

```
echo [host] > vv.repo
echo name=host >> vv.repo
echo baseurl=http://172.31.1.254/centos70/ >> vv.repo
echo gpgcheck=0 >> vv.repo
```

8. Copy it to the nodes, fix up nodes to point to the right IP address:

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
scp vv.repo root@172.31.${i}.1:/etc/yum.repos.d/ ; done
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "sed -i -e s/1.1.254/1.${i}.254/
/etc/yum.repos.d/vv.repo" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
scp vv.repo root@172.31.1.1:/etc/yum.repos.d/
ssh root@172.31.1.1 "sed -i -e s/1.1.254/1.1.254/
/etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.2.1:/etc/yum.repos.d/
ssh root@172.31.2.1 "sed -i -e s/1.1.254/1.2.254/
/etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.3.1:/etc/yum.repos.d/
ssh root@172.31.3.1 "sed -i -e s/1.1.254/1.3.254/
/etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.4.1:/etc/yum.repos.d/
ssh root@172.31.4.1 "sed -i -e s/1.1.254/1.4.254/
/etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.5.1:/etc/yum.repos.d/
ssh root@172.31.5.1 "sed -i -e s/1.1.254/1.5.254/
/etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.6.1:/etc/yum.repos.d/
ssh root@172.31.6.1 "sed -i -e s/1.1.254/1.6.254/
/etc/yum.repos.d/vv.repo"
```

9. (Optional) Verify each node is able to see the yum repository:

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "yum repolist" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "yum repolist"
ssh root@172.31.2.1 "yum repolist"
ssh root@172.31.3.1 "yum repolist"
ssh root@172.31.4.1 "yum repolist"
ssh root@172.31.5.1 "yum repolist"
ssh root@172.31.6.1 "yum repolist"
```

14.5 Creating NFS Share

1. Install NFS on the host system

```
sudo yum -y install --disablerepo=* --enablerepo=localbase
nfs-utils
```

2. Open up the firewall to allow NFS access:

```
for i in public external dmz work home internal trusted; do
sudo firewall-cmd --zone=$i --add-service=nfs --permanent ; done
sudo firewall-cmd --reload
```

3. Share a directory in /etc/exports (*Note: example uses /tmp - replace /tmp with selected share directory in these steps*)

```
ssh root@localhost "echo '/tmp     *(no_root_squash,rw)' >>
/etc/exports"
```

4. Reset NFS service (fully stop and disable, then enable and start), to avoid possible issues:

```
sudo systemctl stop nfs-server
sudo systemctl disable nfs-server
sudo systemctl enable rpcbind
sudo systemctl enable nfs-server
sudo systemctl start rpcbind
sudo systemctl start nfs-server
```

Note

Restarting NSF using presented commands or by:

```
service nfs stop
service nfs restart
```

can hang booted Dom0 DomU images. We recommend to prepare NFS before booting OS or use more safe command:

```
exportsfs -a
```

5. Install NFS on the Intel® Visual Compute Accelerator nodes:

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "yum -y install nfs-utils" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "yum -y install nfs-utils"
ssh root@172.31.2.1 "yum -y install nfs-utils"
ssh root@172.31.3.1 "yum -y install nfs-utils"
ssh root@172.31.4.1 "yum -y install nfs-utils"
ssh root@172.31.5.1 "yum -y install nfs-utils"
ssh root@172.31.6.1 "yum -y install nfs-utils"
```

6. Mount the share on the Intel® Visual Compute Accelerator nodes (*Note: this command mounts to the /media directory on the nodes; change /tmp in the command to the directory shared in step 4, and /media in the command to the directory to which the NFS share should be mounted.*)

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "mount 172.31.${i}.254:/tmp  /media"
; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "mount 172.31.1.254:/tmp  /media"
ssh root@172.31.2.1 "mount 172.31.2.254:/tmp  /media"
ssh root@172.31.3.1 "mount 172.31.3.254:/tmp  /media"
ssh root@172.31.4.1 "mount 172.31.4.254:/tmp  /media"
ssh root@172.31.5.1 "mount 172.31.5.254:/tmp  /media"
```

```
ssh root@172.31.6.1 "mount 172.31.6.254:/tmp  /media"
```

14.6 Setting System Time

1. Set Host system time zone (example: Pacific time - America/Los_Angeles)
   ```
   sudo timedatectl set-timezone America/Los_Angeles
   ```
2. Set Host system time (replace MM with the current two-digit month, DD with the current two-digit date, HH with the current two-digit, 24-hour format hour, MM with the current two-digit minute, YYYY with the current four-digit year; e.g., 032316522015 sets the time to 4:52 pm on 23 March 2015):
   ```
   sudo date MMDDHHMMYYYY
   sudo hwclock --systohc
   ```
3. Set node time zones (to find the correct time zone name, run the command "`timedatectl list-timezones`"):
   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )); do
   ssh root@172.31.${i}.1 "timedatectl set-timezone
   America/Los_Angeles" ; done
   ```
   (*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)
   ```
   ssh root@172.31.1.1 "timedatectl set-timezone
   America/Los_Angeles"
   ssh root@172.31.2.1 "timedatectl set-timezone
   America/Los_Angeles"
   ssh root@172.31.3.1 "timedatectl set-timezone
   America/Los_Angeles"
   ssh root@172.31.4.1 "timedatectl set-timezone
   America/Los_Angeles"
   ssh root@172.31.5.1 "timedatectl set-timezone
   America/Los_Angeles"
   ssh root@172.31.6.1 "timedatectl set-timezone
   America/Los_Angeles"
   ```
4. Get current host system time, push it to the nodes:
   ```
   NOW=`date +%m%d%H%M%Y` ; for (( i=1 ; i<=`sudo vcactl
   status | wc -l` ; i++ )) ; do ssh root@172.31.${i}.1 "date
   $NOW" & done
   ```
   (*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)
   ```
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.1.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.2.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.3.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.4.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.5.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.6.1 "date $NOW"
   ```

14.7 Configuring NTP (Network Time Protocol)

1. Install ntp package on host system (each command starts with "sudo" or "ssh", enter on a single line, even if split in this document):

```
sudo yum -y install --disablerepo=* --enablerepo=localbase
ntp
```

2. Configure NTP server on host system (each command starts with "sudo" or "ssh", enter on a single line, even if split in this document):

```
sudo sed -i -e 's/^server/#server/' /etc/ntp.conf
ssh root@localhost 'echo "server 127.0.0.1" >>
/etc/ntp.conf'
ssh root@localhost 'echo "restrict 172.31.0.0 mask
255.255.0.0 nomodify notrap" >> /etc/ntp.conf'
sudo systemctl restart ntpd
sudo systemctl enable ntpd
```

3. Install and configure ntp package on Intel® Visual Compute Accelerator nodes:

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "yum -y install ntp; sed -i -e
's/^server/#server/' /etc/ntp.conf ; echo server
172.31.${i}.254 >> /etc/ntp.conf ; systemctl enable ntpd;
systemctl restart ntpd" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "yum -y install ntp; sed -i -e
's/^server/#server/' /etc/ntp.conf ; echo server
172.31.1.254 >> /etc/ntp.conf ; systemctl enable ntpd;
systemctl restart ntpd"
ssh root@172.31.2.1 "yum -y install ntp; sed -i -e
's/^server/#server/' /etc/ntp.conf ; echo server
172.31.2.254 >> /etc/ntp.conf ; systemctl enable ntpd;
systemctl restart ntpd"
ssh root@172.31.3.1 "yum -y install ntp; sed -i -e
's/^server/#server/' /etc/ntp.conf ; echo server
172.31.3.254 >> /etc/ntp.conf ; systemctl enable ntpd;
systemctl restart ntpd"
ssh root@172.31.4.1 "yum -y install ntp; sed -i -e
's/^server/#server/' /etc/ntp.conf ; echo server
172.31.4.254 >> /etc/ntp.conf ; systemctl enable ntpd;
systemctl restart ntpd"
ssh root@172.31.5.1 "yum -y install ntp; sed -i -e
's/^server/#server/' /etc/ntp.conf ; echo server
172.31.5.254 >> /etc/ntp.conf ; systemctl enable ntpd;
systemctl restart ntpd"
ssh root@172.31.6.1 "yum -y install ntp; sed -i -e
's/^server/#server/' /etc/ntp.conf ; echo server
172.31.6.254 >> /etc/ntp.conf ; systemctl enable ntpd;
systemctl restart ntpd"
```

14.8 Turn on affinity

Affinity setting can be changed on host to balance simultaneous transfers between host and all nodes. We recommend to turn it on.

4. Open file /etc/sysconfig/irqbalance
5. Find line:
   `#IRQBALANCE_ARGS=`
6. Change this line to (do not forget to remove #):
   `IRQBALANCE_ARGS="-h exact"`
7. Save file
8. Reboot host.

14.9 Host BIOS additional configuration

We recommend to set BIOS parameters (if available) to following settings:

- Memory Mapped IO above 4GB: Enabled
- Memory Mapped IO size: 256GB

14.10 Configuring Windows Client to Access VNCViewer

1. Install XMing (http://sourceforge.net/projects/xming/) and PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) on the Windows client.
2. Launch XMing
3. Launch PuTTY
    a. Enter the IP address of the Intel® Visual Compute Accelerator Host system in the "Host Name (or IP address)" text box
    b. Enter a name for the system in the "Saved Sessions" text box
    c. In the left-side navigation pane, Expand Connection -> SSH, then click "X11" below "SSH"
    d. In the right-side settings pane, check the "Enable X11 forwarding" checkbox, then enter ":0" (without quotes) in the "X display location" text box
    e. In the left-side navigation pane, click "Window"
    f. In the right-side settings pane, enter "10000" (without quotes) in the "Lines of scrollback" text box (replacing whatever value is there, standard default is 200)
    g. In the left-side navigation pane, scroll to the top, and click "Session"
    h. In the right-side settings pane, click the "Save" button next to the "Saved Sessions" list.
    i. At any future point, the connection to the Intel® Visual Compute Accelerator system can be opened by launching PuTTY then either:
        i. Double-clicking the name from the "Saved Sessions" list, or
        ii. Clicking the name from the "Saved Sessions" list, then clicking "Load", then clicking "Open"

14.11 Install and configure VNC

1. Install and configure vnc package on Intel® Visual Compute Accelerator nodes:

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "yum -y install tigervnc-server
xauth; cp /lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver@:1.service; sed -i -e
s='<USER>'=root= /etc/systemd/system/vncserver@:1.service;
echo vista1 > /tmp/passwd; vncpasswd -f < /tmp/passwd >
.vnc/passwd; chmod 600 .vnc/passwd" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "yum -y install tigervnc-server xauth;
cp /lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver@:1.service; sed -i -e
s='<USER>'=root= /etc/systemd/system/vncserver@:1.service;
echo vista1 > /tmp/passwd; vncpasswd -f < /tmp/passwd >
.vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.2.1 "yum -y install tigervnc-server xauth;
cp /lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver@:1.service; sed -i -e
s='<USER>'=root= /etc/systemd/system/vncserver@:1.service;
echo vista1 > /tmp/passwd; vncpasswd -f < /tmp/passwd >
.vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.3.1 "yum -y install tigervnc-server xauth;
cp /lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver@:1.service; sed -i -e
s='<USER>'=root= /etc/systemd/system/vncserver@:1.service;
echo vista1 > /tmp/passwd; vncpasswd -f < /tmp/passwd >
.vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.4.1 "yum -y install tigervnc-server xauth;
cp /lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver@:1.service; sed -i -e
s='<USER>'=root= /etc/systemd/system/vncserver@:1.service;
echo vista1 > /tmp/passwd; vncpasswd -f < /tmp/passwd >
.vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.5.1 "yum -y install tigervnc-server xauth;
cp /lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver@:1.service; sed -i -e
s='<USER>'=root= /etc/systemd/system/vncserver@:1.service;
echo vista1 > /tmp/passwd; vncpasswd -f < /tmp/passwd >
.vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.6.1 "yum -y install tigervnc-server xauth;
cp /lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver@:1.service; sed -i -e
s='<USER>'=root= /etc/systemd/system/vncserver@:1.service;
echo vista1 > /tmp/passwd; vncpasswd -f < /tmp/passwd >
.vnc/passwd; chmod 600 .vnc/passwd"
```

2. Install vnc client package on host:

```
sudo yum -y install --disablerepo=* --enablerepo=localbase
tigervnc
```

3. Configure Firewall on nodes:

```
for i in 1 2 3 4 5 6 ; do ssh root@172.31.${i}.1 \
"firewall-cmd --permanent --add-port=5901/tcp ; \
firewall-cmd –reload"; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "firewall-cmd --permanent --add-
port=5901/tcp ; firewall-cmd –reload"
ssh root@172.31.2.1 "firewall-cmd --permanent --add-
port=5901/tcp ; firewall-cmd –reload"
ssh root@172.31.3.1 "firewall-cmd --permanent --add-
port=5901/tcp ; firewall-cmd –reload"
ssh root@172.31.4.1 "firewall-cmd --permanent --add-
port=5901/tcp ; firewall-cmd –reload"
ssh root@172.31.5.1 "firewall-cmd --permanent --add-
port=5901/tcp ; firewall-cmd –reload"
ssh root@172.31.6.1 "firewall-cmd --permanent --add-
port=5901/tcp ; firewall-cmd –reload"
```

4. Configure Firewall on host (two commands):

```
sudo firewall-cmd --permanent --add-port=5901/tcp
sudo firewall-cmd –reload
```

14.12 Launching VNC

1. Launch VNC server on Intel® Visual Compute Accelerator nodes:

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do
ssh root@172.31.${i}.1 "vncserver &" ; done
```

(*Note: the following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.*)

```
ssh root@172.31.1.1 "vncserver &"
ssh root@172.31.2.1 "vncserver &"
ssh root@172.31.3.1 "vncserver &"
ssh root@172.31.4.1 "vncserver &"
ssh root@172.31.5.1 "vncserver &"
ssh root@172.31.6.1 "vncserver &"
```

2. Connect to the host with PuTTY from a Windows client, and launch vncviewer:
   a. Launch PuTTY on the Windows client
   b. Double-click the session saved in the previous section
   c. Log in as "root", with the "root" user's password
   d. Issue the command (Note: 5901 may be 5902 or 5903):

```
vncviewer 172.31.X.1:5901
```

      # where X is replaced by the node number to which a connection is desired.
   e. (*Note: this will fail if XMing is not running on the Windows client before the VNC viewer command is issued.*)

15. IP multicast with VCA nodes

VCA network stack implements standard network device and thus it is capable of sending and receiving packets to multicast addresses as well as IGMP packets.

Default VCA network topology consists of separate sub-networks for each node. In such setup the host system must act as a multicast router for the nodes to be able to reach each other via multicast.

15.1 Multicast routing with mrouted

This section contains example configuration for multicast routing using mrouted daemon.

15.1.1 Notes on mrouted

Mrouted is a daemon used to manage dynamic multicast routing, using DVMRP (Distance Vector Multicast Routing Protocol). It is an open source tool, released on Standford's license, available at http://troglobit.github.io/mrouted.html

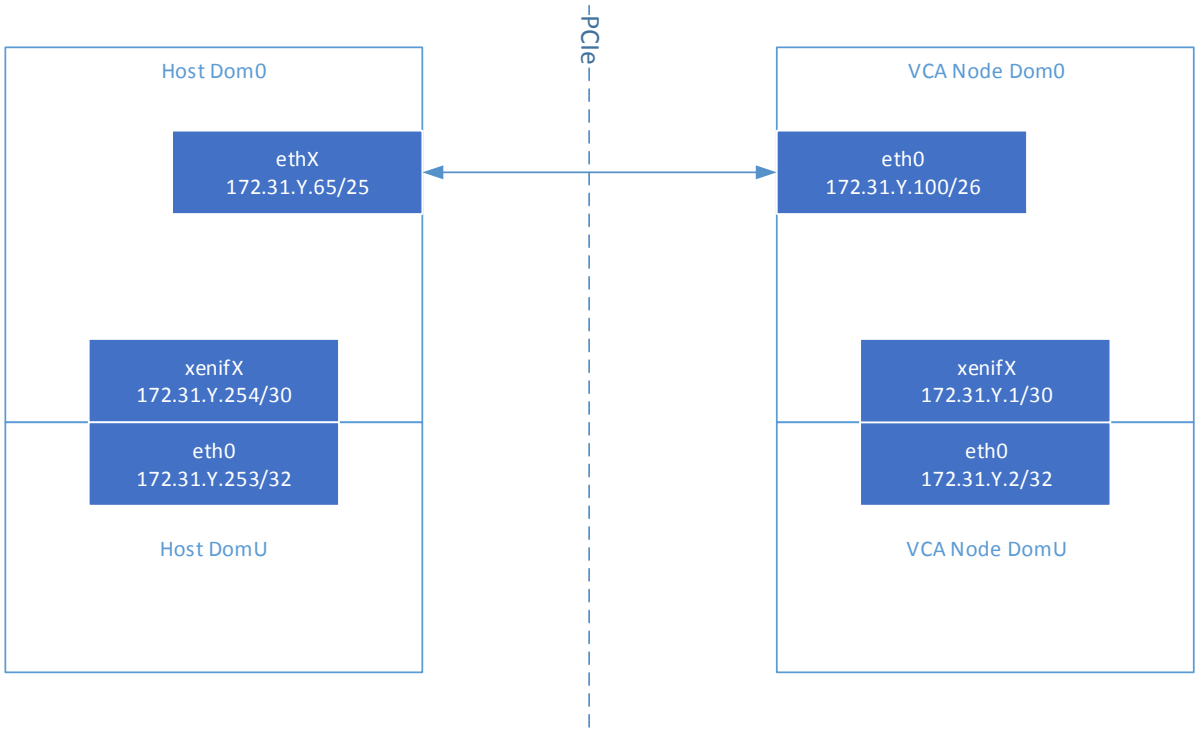
The following limitations of mrouted daemon have been discovered so far:

- mrouted relies only on network addressing and masks and doesn't analyse routing tables
- does not support interfaces with 32-bit masks
- mrouted ignores network interface if its ip range is sub-range of other interaface address range. For exapmple interface eth0 172.31.1.1/32 would not be considered for routing if xenif0 172.31.1.2/24 is present.
- Network addressing needs to be changed when DomU is used on host or VCA nodes so that all networks are separated – see the next paragraph.
- mrouted scans for network interfaces only during daemon start
- mrouted must be manually started by user, when all VCA nodes intended for multicast using are already up (can connect to their unicast addresses); because of this mrouted cannot be a service autostarted within host OS boot
- mrouted doesn't detect interface down and up
- when any VCA node using multicast is reset, mrouted daemon needs to be restarted after bringing up the node
- mrouted clears multicast routing table at exit

- when mrouted is restarted, all clients must re-register into multicast group, otherwise they won't be able to receive multicast messages
- multicast messages sent by host cannot be received by card's nodes (under investigation, as network packet analysis shows, that multicast packets arrives to the nodes)

15.1.2 Example configuration for Xen Domain U

If DomU needs to be used, network addresses of host and card interfaces need to be changed, as well as mask width. This is implication of mrouted limitation, which can operate only on networks with exclusive address ranges. Suggested network addressing is the following:



User must manually apply those settings. It can be done by simple copying /etc/vca_config.d/vca_xen_multicast_config.xml to /etc/vca_config.d/vca_config.xml prior to card nodes booting.

15.1.3 Preparation

1. Before using mulicasts for the firs time, user must install mrouted daemon on the host. The RPM package can be found within VCA release in INSTALL\Binaries\mrouted\ directory.
2. Boot VCA node using vcactl boot command (baremetal or xen image).
3. If Xen domain U is required on VCA node, start it with --multicast option

```
vca_start_card_domu.sh -p <nfs path> -c <card id> -n <CPU ID> -i
<image name> -f <config file name> --multicast
```

4. To start DomU on host no additional parameters are needed for multicast support:

   ```
   vca_start_host_domu.sh -c <card id> -n <CPU ID> -i <image name> -
   f <config file name>
   ```

5. Start mrouted daemon:

   ```
   mrouted
   ```

   Note: After restarting DomU on host, mrouted daemon on host needs to be restarted. There is no such need in case of restarting DomU on card (assuming that the Dom0 on card hasn't been restarted)

15.1.4 Restarting mrouted daemon

There is no option in mrouted daemon to restart it. User needs to kill the existing instance with system tools and start the daemon manually once again:

```
killall mrouted

mrouted
```

15.1.5 Checking multicast routes

To check registered multicast groups known to mrouted on available network adapters, invoke:

```
mrouted -r
```

Sample output would look like this:

```
vifs_with_neighbors = 1
[This host is a leaf]


Virtual Interface Table
Vif  Name  Local-Address                              M  Thr  Rate   Flags
 0   eth0  172.31.1.100    subnet: 172.31.1.64/26      1   1      0
                            peers: 172.31.1.65 (3.255) [0] have-genid up  1:34:37
           group host (time left): 224.94.1.1      172.31.1.100    ( 0:03:40)
                                   224.0.0.2       172.31.1.65     ( 0:03:39)
                                   224.0.0.4       172.31.1.65     ( 0:03:40)
                     IGMP querier: 172.31.1.65        up  1:34:32 last heard  0:00:45
ago
```

```
                      Nbr bitmaps: 0x0000000000000001
                    pkts/bytes in : 1913/2831956
                    pkts/bytes out: 0/0
 1 xenif0  172.31.1.1      subnet: 172.31.1.0/30       1   1      0   querier leaf
           group host (time left): 224.94.1.2      172.31.1.2      ( 0:03:39)
                                   224.0.0.4       172.31.1.1      ( 0:03:37)
                                   224.0.0.2       172.31.1.1      ( 0:03:37)
                     IGMP querier: 172.31.1.1         (this system)
                      Nbr bitmaps: 0x0000000000000000
                    pkts/bytes in : 0/0
                    pkts/bytes out: 540/808920


Multicast Routing Table (9 entries)
 Origin-Subnet      From-Gateway    Metric Tmr Fl In-Vif  Out-Vifs
 172.31.1.0/30                         1    75 ..   1    0*
 172.31.1.64/26                        1    75 ..   0    1*
 192.168.122/24     172.31.1.65        2    25 ..   0    1*
 172.31.6/24        172.31.1.65        2    25 ..   0    1*
 172.31.5/24        172.31.1.65        2    25 ..   0    1*
 172.31.4/24        172.31.1.65        2    25 ..   0    1*
 172.31.3/24        172.31.1.65        2    25 ..   0    1*
 172.31.2/24        172.31.1.65        2    25 ..   0    1*
 10.102.108/23      172.31.1.65        2    25 ..   0    1*
```

Virtual Interface Table shows list of all managed interfaces (here: eth0 and xenif0) with their addresses and subnet masks. For each interface, registered multicast groups are shown (eth0: 224.94.1.1, 224.0.0.2 and 224.0.0.4; xenif0: 224.94.1.2, 224.0.0.4 and 224.0.0.2)

To check all active routings invoke:
`ip mroute`

Sample output:

```
(172.31.1.100, 224.94.1.1)        Iif: eth0

(10.102.109.202, 224.94.1.2)      Iif: eth0       Oifs: xenif0
```

This means that only a single stream is active, transmitting data to multicast group 224.94.1.2, originated by an external host with address 10.102.109.202, received on eth0 interface and passed to xenif0 interface (because xenif0 interface has registered to 224.94.1.2 group).

15.2 Sample usage

A sample application supporting multicasts is iperf tool (please notice iperf3 doesn't support multicast, only iperf does). All images delivered with VCA release already contain *iperf*. On host it might be required to install it manually.

To start multicast listener invoke:
```
iperf -s -u -B <group address> -i 1
```

for example:

```
iperf -s -u -B 224.94.1.1 -i 1
```

To start multicast transmitter invoke:

```
iperf -c <group address> -u -T 32 -t 3 -i 1
```

for example:

```
iperf -c 224.94.1.1 -u -T 32 -t 3 -i 1
```

Order of operations is the following:

1. Boot host
2. Boot card's nodes (and DomUs, if needed)
3. Start mrouted daemon
4. On all nodes and on server stop firewalld service (or provide proper firewall exceptions for IGMP and iperf 5001 port)
5. Start iperf multicast listeners on selected nodes
6. Start iperf multicast transmitters