GPU Configuration
Open-Source Mini-Clusters
CentOS8.3 Base OS, OpenHPC (v2.3)
xCAT/SLURM Edition for Linux* (x86_64)

Isayah Reed
December 2021
Revision 1.1

Table of Contents

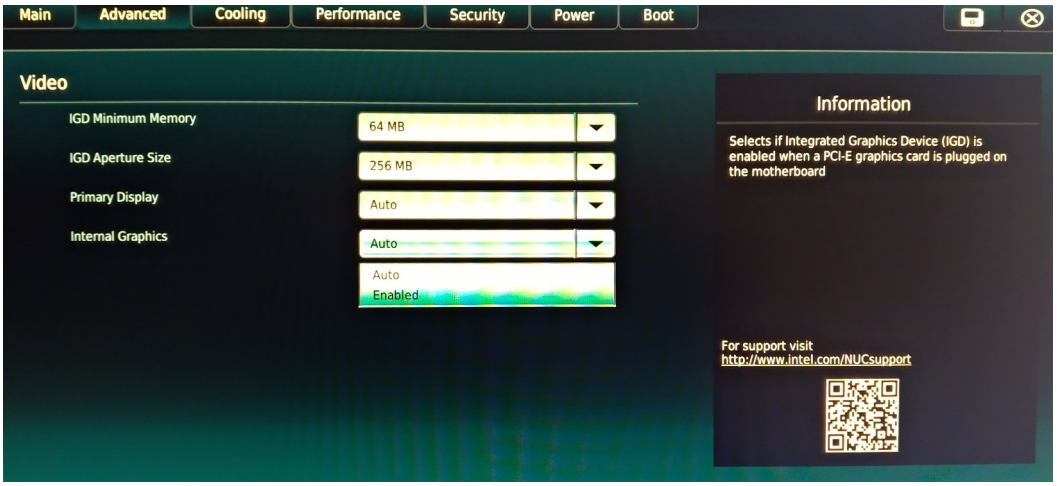
Introduction....2
Nvidia Driver: Installation....3
Low Bandwidth....4
Nvidia Driver: Compute Node Installation....5

Version	Description
v1.0	Initial draft. Moved Nvidia section from main cluster doc.

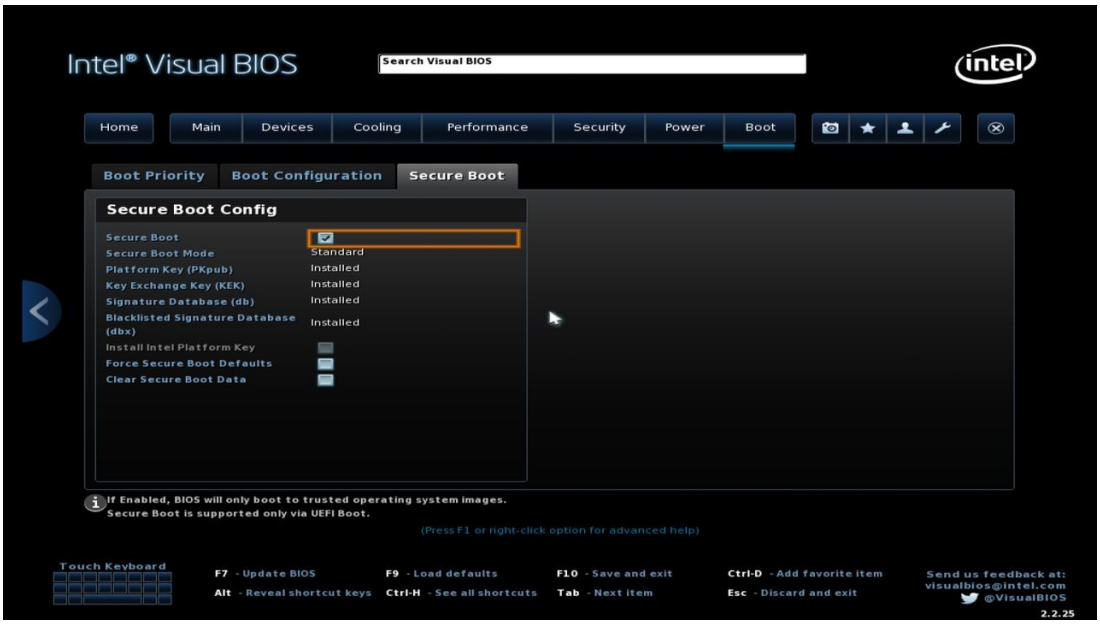
Introduction

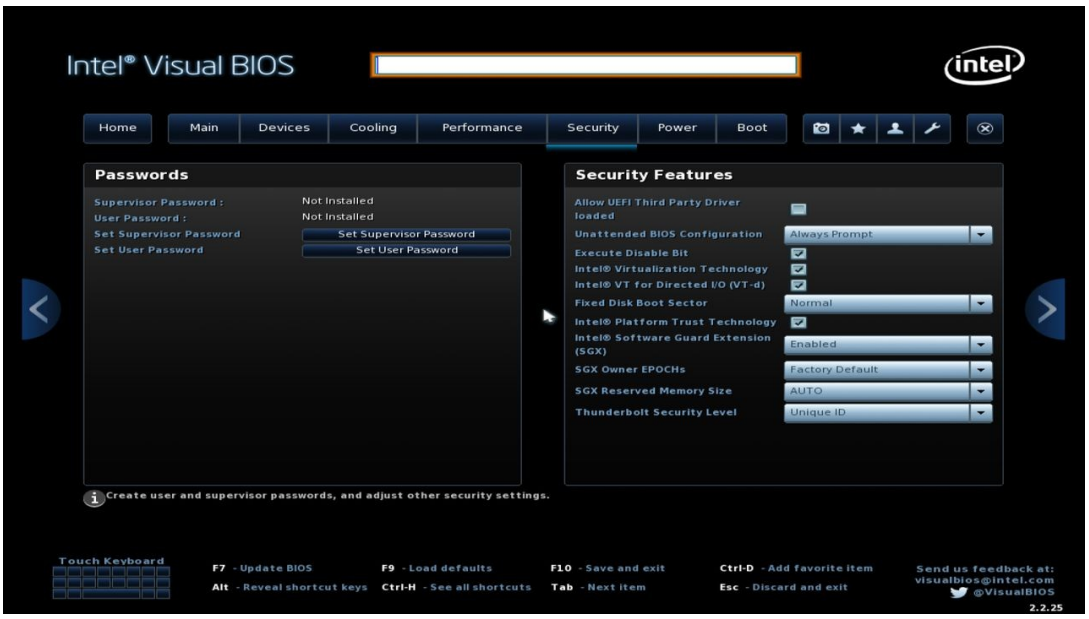
GPU installation and configuration is independent of any instructions in the top-level cluster setup document and does not require packages from the cluster setup. Instructions in this document can be performed before, or without, setting up the cluster environment.

If a discrete gpu is installed on the management node, the BIOS should be set to use internal graphics.



The device(s) with the GPU should have secure boot turned OFF in the BIOS, and select the BIOS option to allow 3rd party drivers during boot.





Nvidia Driver: Installation

To ensure compatibility with our hardware and cluster setup, we target specific versions of the Nvidia drivers and cuda that have been validated with hardware in this cluster environment, instead of installing the latest versions.

Before the driver can be installed, update the OS tools and libraries, add the required repository for driver tools we will need, and install the wget tool that will be needed for downloading certain files. It is possible that the update either added or changed kernel modules, which would require a reboot.

```
$> dnf clean all
$> dnf -y update
$> dnf -y --enablerepo=extras install epel-release
$> dnf -y install wget pciutils dkms
$> reboot
```

The Nvidia driver requires that kernel headers and kernel development packages for the current version of the kernel are installed before the driver is installed or rebuilt.

```
$> dnf -y --enablerepo=baseos install kernel-devel-$(uname -r) kernel-headers-$(uname -r)
```

Check the installed graphics devices.

```
$> lspci | grep VGA
```

There should be at least two listed devices: one is the integrated Intel graphics processor, and another entry should be the discrete GPU device.
Checking the installed graphics driver shows that the Nvidia driver is not being used.

```
$> lshw -c video | grep 'configuration'
```

The default graphics driver for linux is the open-source nouveau. Using the Nvidia driver requires deactivating the nouveau driver and replacing with the proprietary Nvidia driver. One way to do this is with the Nvidia driver package.
Add the Nvidia repository and install the Nvidia driver package:

```
$> dnf config-manager --add-repo  \
      https://developer.download.nvidia.com/compute/cuda/repos/rhel8/x86_64/cuda-rhel8.repo
$> dnf -y module install nvidia-driver:470-dkms
```

A reboot is required for the driver to load. Before rebooting, confirm that the driver is installed. The dkms output should show the Nvidia driver, as well as the associated kernel version.

```
$> dkms status
$> reboot
```

After rebooting, verify that the Nvidia driver is installed and working.

```
$> nvidia-smi
```

The output should show information about the external GPU installed. Finally, install the cuda environment.

```
$> dnf -y install cuda-11-2
```

Low Bandwidth

If internet speed or low bandwidth is a problem, an alternate method for installing the Nvidia GPU driver is to download the cuda file and build it locally. The cuda package from the previous method is ~3.8GB, so downloading over the network would cause issues if internet speed is low.

```
$> wget -c http://developer.download.nvidia.com/compute/cuda/11.2.2/local_installers/
cuda_11.2.2_460.32.03_linux.run
$> sh ./cuda_11.2.2_460.32.03_linux.run
```

The local installer is a self-contained file with all components needed for cuda and the graphics driver. It only needs to be downloaded once. This method is limited to the listed versions: cuda version 11.2 and Nvidia driver version 460. Changing the versions requires downloading and executing a different file – for example:

```
$> wget -c https://developer.download.nvidia.com/compute/cuda/11.4.1/local_installers/
cuda_11.4.1_470.57.02_linux.run
```

Verify that the driver and cuda is installed.

```
$> dkms status
```

The output should show the Nvidia driver and kernel version.
A reboot is required for the changes to take effect after using the local installer.

```
$> reboot
```

After reboot, verify that the driver is active and cuda is installed.

```
$> lshw -c video | grep 'configuration'
$> nvidia-smi
$> /usr/local/cuda/bin/nvcc -V
```

If the driver is installed but not working then the nvidia-smi will fail.
The lshw command should now show that the nouveau driver is no longer being used, instead outputting '*Nvidia*' as the driver where the nouveau driver information was previously listed.

Nvidia Driver: Compute Node Installation

Instructions to install the Nvidia driver on compute nodes are similar to instructions for the head node. Add the Nvidia repository and install the Nvidia driver package:

```
$> dnf --installroot=$CHROOT config-manager --add-repo  \
      https://developer.download.nvidia.com/compute/cuda/repos/rhel8/x86_64/cuda-rhel8.repo
$> dnf --installroot=$CHROOT -y module install nvidia-driver:470-dkms
```

Unlike the management node, it is not recommended to install cuda-11-2 into the diskless image. While this can be done by performing similar instructions listed in the section for the management node (`$> dnf--installroot=$CHROOT -y install cuda-11-2`), the cuda-11-2 package is large - 3.1GB - resulting in too much space used on the compute nodes in a smaller cluster environment. Instead, share the cuda folder from the head node across NFS network. To save space on the diskless compute nodes, the cuda folders on the head node will be shared with the compute nodes through NFS.

```
$> echo "10.10.1.10:/usr/local/cuda /usr/local/cuda nfs nfsvers=3,nodev,nosuid 0 0" >>
$CHROOT/etc/fstab
$> echo "/usr/local/cuda *(ro,no_subtree_check,fsid=14)" >> /etc/exports
$> systemctl restart nfs-server
$> packimage centos8-x86_64-netboot-compute
$> pdsh -w c[1-2] reboot
```

Note the fsid=14. This number may need to be changed, depending on other folders shared. Check /etc/exports to see if FSID 14 has been reserved for a different folder. If it is, then change to the lowest number that is not being used.

Similar to the head node, check with nvidia-smi

```
$> nvidia-smi
```

Sample output: