Intel® Select Solutions Implementation Guide for Simulation and Modeling

Based on OpenHPC 1.3.4 for
CentOS 7.4 with
Intel® Xeon® Scalable Processors and
Intel® Omni-Path Fabric

CONTENTS

REVISIONS

Version	Date	Description	Authors
0.1	01/03/2018	Draft	Stephen Chang
0.2	04/23/2018	Draft	J.C. Chen
1.0	06/15/2018	Final v 1.0	J.C. Chen

1. Introduction

1.1.Summary

OpenHPC represents an aggregation of a number of common ingredients required to deploy and manage an HPC Linux cluster including provisioning tools, resource management, I/O clients, development tools, and a variety of scientific libraries. These packages have been pre-built with HPC integration in mind while conforming to common Linux distribution standards.

The documentation herein is intended to be reasonably generic, but uses the underlying motivation of a small, 4-node stateful cluster installation to provide a step-by-step instruction process. Several optional customizations are included, and the intent is that these collective instructions can be modified as needed for local site customizations.

This document will demonstrate an implementation guide of the Intel® Scalable System Framework Reference Architecture and OpenHPC software to meet Intel® Select Solutions for Simulation and Modeling requirements, complete with hardware and software Bill of Materials.

1.2. High Performance Computing (HPC) Cluster Architecture

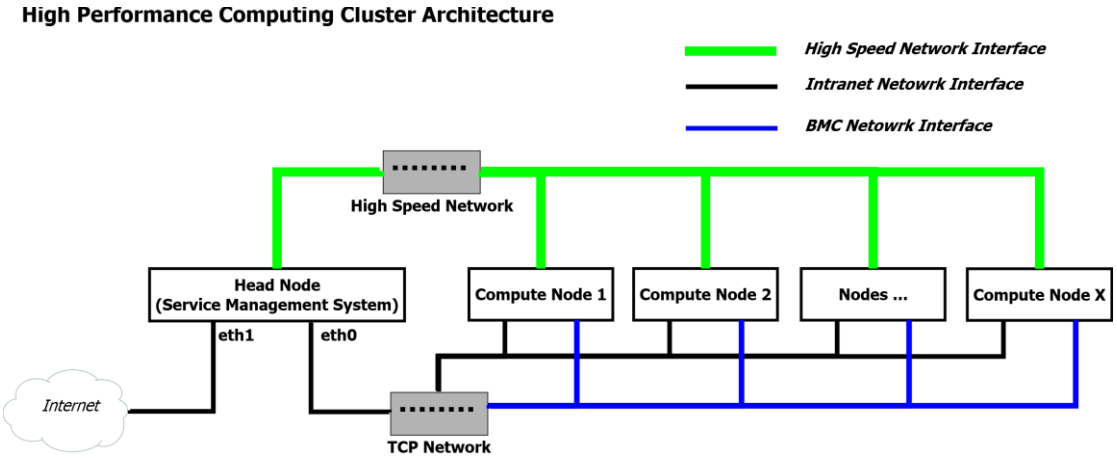


Figure 1: Overview of High Performance Computing (HPC) Cluster Architecture

HPC Cluster Architecture usually have one Head Node host and numbers of Compute Node Hosts. Head Node host may use as Services Management System (SMS) host and provide necessary services for Compute Nodes. Those Compute Nodes is scalable as requirement and Head Node should provide cluster management for these Compute Nodes. Before we start the implementation of our cluster environment, we need to plan and setup cluster roles for each node.

1.3. Hardware Bill of Materials

Hardware Bill of Materials for the Head Node

Qty	Item	Manufacturer	Model
1	Quanta Cloud Technology Inc. Server Chassis	Quanta Cloud	QuantaGrid D52BQ-2U
1	Quanta Cloud Server Board	Quanta Cloud	S5BQ-MB
	(2x) Intel® Xeon® Gold Processor	Intel	6148
	(12x) 32GB ECC DDR4 2666MHz	Micron	MTA18ASF1G72PZ
	(2x) Intel® SSD 240GB, 2.5-inch SATA III	Intel	S4500 Series
	(6x) Hitachi 6T, 3.5-inch SATA III	Toshiba	MG04SCA60EE
	(1x) Intel® Ethernet Connection	Intel	X722

Hardware Bill of Materials for the Compute Nodes

Qty	Item	Manufacturer	Model
4	Quanta Cloud Technology Inc. Server Chassis	Quanta Cloud	QuantaPlex T42S-2U
4	Quanta Cloud Server Board	Quanta Cloud	T42S-2U MB
	(2x) Intel® Xeon® Gold Processor	Intel	6148
	(4x) 32GB ECC DDR4 2666MHz	Micron	MTA18ASF1G72PZ
	(2x) Intel® SSD 240GB, 2.5-inch SATA III	Intel	S4500 Series
	(1x) Intel® Ethernet Connection	Intel	X722

Hardware Bill of Materials for the Infrastructure

Qty	Item	Manufacturer	Model
1	QuantaMesh BMS T3040-LY3	Quanta Cloud	40 100/1000/10GBT and 8 1/10GbE SFP+ ports
1	Intel® Omni-Path Edge Switch	Intel	100 Series 24-Port Managed

1.4. Software Bill of Materials

Software	Version
CentOS Installation DVD and Repositories	7.4
Intel® HPC Orchestrator Advanced	3A11.Q01
Intel® Parallel Studio XE Cluster Edition	2018
Intel® Cluster Checker	2018.3

1.5. Firmware Versions and BIOS Settings

Software	Version
BIOS for Head and Compute Node	7.4
Firmware for Head and Compute Nodes	3A11.Q01
Firmware for Intel® Omni-Path Switch	2018

2. Preparation

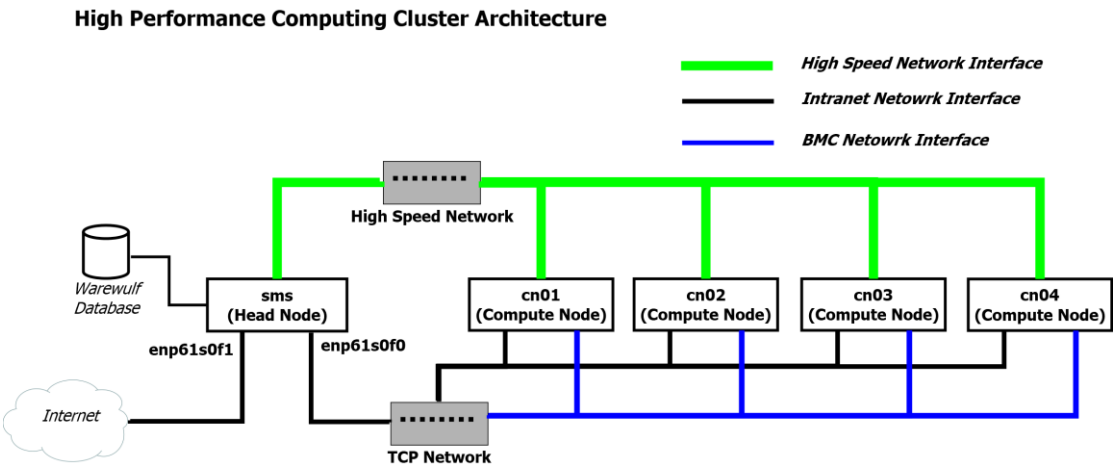


Figure 2: Overview of High Performance Computing (HPC) Cluster Implementation Architecture

2.1. Assembly

Planning to setup an OpenHPC Cluster environment which have one Head Node and four Compute Nodes. Head Node host will run as Services Management System (SMS) Host and provide provisioning services, resource management and related services. Following is the planning host information for our cluster environment.

Cluster/Subcluster	
Internal Subnet (Ethernet)	192.168.81.0
Internal Netmask (Ethernet)	255.255.255.0
BMC Subnet	192.168.100.0
BMC Netmask	255.255.0.0
Internal Subnet (Intel® Omni-Path)	192.168.82.0
Internal Netmask (Intel® Omni-Path)	255.255.255.0
Head Node	
Hostname	sms
Internal IP Address (Ethernet)	192.168.81.1
Internal Network Device (Ethernet)	enp61s0f0
External Network Device (Ethernet)	enp61s0f1
BMC IP Address	192.168.100.1
Hostname (Intel® Omni-Path)	sms-ib0
Internal IP Address (Intel® Omni-Path)	192.168.82.254
Internal Network Device (Intel® Omni-Path)	ib0
Compute Node	
Hostname	cn[01-04]
Internal IP Address (Ethernet)	192.168.81.[101-104]
Internal Network Device (Ethernet)	enp96s0f0
BMC IP Address	192.168.100.[101-104]
Internal IP Addresses (Intel® Omni-Path)	192.168.82.[101-104]
Internal Network Device (Intel® Omni-Path)	ib0

Please refer to the following table for the system environment variable for the command that used in this document.

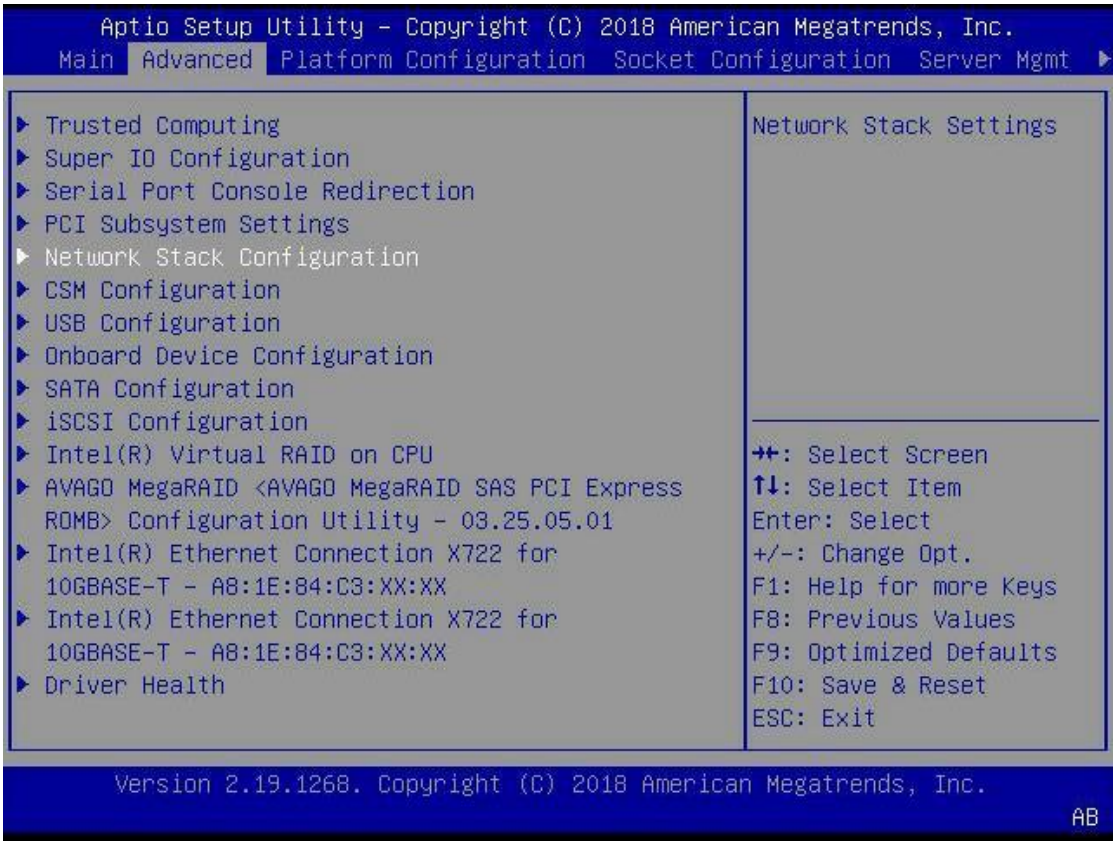
Environment Variable	Description	Example Value
${sms name}	SMS Host Hostname	sms
${sms_ip}	SMS Host Internal IP Address	192.168.81.1
${sms_eth_internal}	SMS Host Internal Ethernet Interface	enp61s0f0
${eth_provision}	SMS Host Provisioning Interface for Compute Nodes	enp61s0f0
${internal_netmask}	SMS Host Internal Network Subnet Netmask	255.255.0.0
${ntp_server}	Assigned NTP Server Name	time.stdtime.gov.tw
${bmc_username}	BMC User Name for IPMI Tool	bmc_admin
${bmc_password}	BMC User Password for IPMI Tool	bmc_password
${num_computes}	Compute Nodes Total Number	4
${c_ip[0]}, ${c_ip[1]}, … ${c_ip[n]}	Compute Nodes IP Address	192.168.81.[101-104]
${c_bmc[0]}, ${c_bmc[1]}, … ${c_bmc[n]}	Compute Nodes BMC IP Address	192.168.100.[101-104]
${c_mac[0]}, ${c_mac[1]}, … ${c_mac[n]}	Compute Nodes MAC Address	a8:1e:84:c3:99:99 …
${c_name[0]}, ${c_name[1]}, … ${c_name[n]}	Compute Nodes Host Name	cn01, cn02, cn03, cn04
${compute_regex}	Compute Node Name Regex Matching	cn[01-04]
${compute_prefix}	Prefix for Compute Node Names	cn
${sms_ipoib}	SMS Host IPoIB Address	192.168.100.1
${ipoib_network}	SMS Host IPoIB Subnet Netmask	255.255.0.0
${c_ipoib[0]}, ${c_ipoib[1]}, …${c_ipoib[n]}	Compute Nodes IPoIB Address	192.168.100.[101-104]
${kargs}	Kernel Boot Arguments	net.ifnames=1
${nagios_web_password}	Nagios Web Access Password	nagios_password

Table 7: System Environment Variable and Example Values

2.2.Information Collection and Setup

2.2.1. MAC Address Information

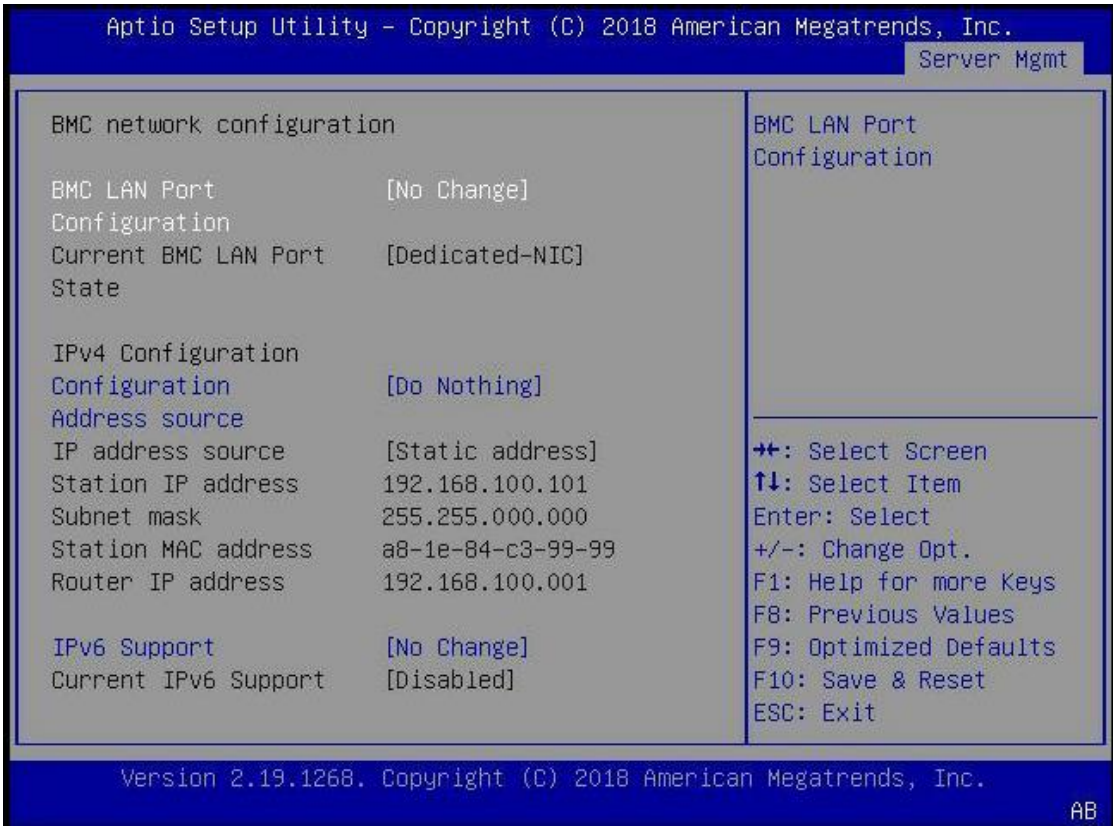
We will need MAC address information for the compute nodes provisioning. The MAC address information can be acquired from the BIOS setup window. Enter BIOS -> Advanced window and check Ethernet Connection Device MAC address. Eg. Intel(R) Ethernet Connection X722 for 10GBASE-T – A8:1E:84:C3:XX:XX. First Network Interface MAC address is A8:1E:84:C3:XX:XX. Please noted that network devices name may be different due to different hardware environment. Please collect each nodes' MAC address and record them for the setup processes.



2.2.2. BMC Lan Information

It is Baseboard Management Controller (BMC) that we are using to remote control cluster nodes. Please collect and setup BMC network environment for every node in the cluster. The BMC Lan Information may check/setup in the BIOS setup window on each node. Please collect BMC network setup information and record them for the implementation processes

BIOS/BMC network configuration	
IP address source	[Static address]
Station IP address	192.168.100.101
Subnet mask	255.255.255.000
Station MAC address	a8-1e-84-c3-XX-XX
Router IP address	192.168.100.001



2.2.3. BIOS Configuration

For the implementation, we assume that legacy boot modes are enabled on all nodes. And compute nodes have boot over first network device priority in BIOS setting.

Following is the example for the CPU configuration options in BIOS for cluster nodes.

BIOS/CPU Setting	
Hyper-Threading [ALL]	Enable
Execute Disable Bit	Enable
Enable Intel(R) TXT	Disable
VMX	Enable
Enable SMX	Disable
Hardware Prefetcher	Enable
Adjacent Cache Prefetch	Enable
DCU Streamer Prefetcher	Enable
DCU IP Prefetcher	Enable
LLC Prefetch	Disable

For stateless provisioning to compute nodes, compute nodes request to enable PXE Support and high priorities for booting form network. Please refer to following example for Compute Nodes BIOS setting for PXE support and fixed boot order example.

BIOS/PXE Support	
Network Stack	Enable
Ipv4 PXE Support	Enable
Ipv4 HTTP Support	Enable
Ipv6 PXE Support	Enable
Ipv6 HTTP Support	Enable

BIOS/FIXED BOOT ORDER Priorities for Compute Nodes	
Boot Option #1	[Network:UEFI: Slot Port0 HTTP IPv4 Intel(R) Ethernet Connection X722 for 10GBASE-T]
Boot Option #2	[Hard Disk:CentOS]
Boot Option #3	[CD/DVD]
Boot Option #4	[USB]

Please make sure that we have setup correct boot order for Head Node (SMS) host depends on how we will boot and install Base Operation System (BOS). Following example is planning to install BOS from CD/DVD source.

BIOS/FIXED BOOT ORDER Priorities for Head Node	
Boot Option #1	[CD/DVD]
Boot Option #2	[Hard Disk:CentOS]
Boot Option #3	[Network:UEFI: Slot Port0 HTTP IPv4 Intel(R) Ethernet Connection X722 for 10GBASE-T]
Boot Option #4	[USB]

3. OpenHPC Cluster Install and Setup

3.1 Service Management System (SMS) Installation

We will need to install or setup a Base Operation System (BOS) running as the Services Management System (SMS) in our OpenHPC Cluster environment. We can install the BOS from CentOS7.4 DVD, ISO Image, network file services, or use a pre-installed server.

3.1.1 Install Base Operation System (BOS)

You may boot up the system from physical CentOS 7.4 DVD and follow up the instruction and complete the Base Operation System installation.

Another way is use Baseboard Management Controller (BMC) to mount the CentOS 7.4 DVD ISO image and boot up the system. Then, follow up the instruction and complete the Base Operation System installation.

3.1.2 Use Pre-Installed Server

For pre-installed server, please verify if the system is provisioned with the required CentOS7.4 distribution.

3.1.3 Base Operation System (BOS) Environment

After the Base Operation System build up, we need to setup the basis environment for the SMS host.

In this implementation guide, we have setup RAID5 disk array in BIOS. The following steps will help and demonstrate how to create and use Logical Volume Manager to manage file system. Please note that it may be different hardware environment and the command shall be revised accordingly. Otherwise, it may cause the filesystem damage.

```
# partition manipulation for disk (Please make sure the disk device name is correct)
[sms]# parted /dev/sdc
# Initializing Physical Volumes
[sms]# pvcreate /dev/sdc
# Initializing Volume Groups in Physical Volumes
[sms]# vgcreate vg_work /dev/sdc
# Initializing Logical Volumes in Volume Group
[sms]# lvcreate -n lv_home -l 50%FREE vg_work
# Construct an XFS filesystem on Logical Volumes
[sms]# mkfs.xfs /dev/vg_work/lv_home
# Mount Constructed XFS filesystem and update File System Table
[sms]# mount -t xfs /dev/vg work/lv home /home
[sms]# echo "/dev/vg_work/lv_home /home xfs defaults 0 0" >> $CHROOT/etc/fstab
```

Check and verify the SMS host and Compute Nodes can be resolvable locally. Please add SMS and Compute Node Names and IP address in /etc/hosts file if those hosts information not exists.

```
# Add SMS and Compute Node Host Names and IP Address Information in /etc/hosts
[sms]# vi /etc/hosts
192.168.81.1 sms
192.168.81.101 cn01
192.168.81.102 cn02
192.168.81.103 cn03
192.168.81.104 cn04
```

Cluster provisioning services requires DHCP, HTTP and TFTP network protocols services from SMS host. In this implementation guide, we will disable SMS host firewall temporary in vent of block those related services from compute nodes. After setup the cluster environment, we shall setup and enable firewall for security concern. This part is out of discussion in this document.

```
# Disable and Stop Firewall Services on SMS host
[sms]# systemctl disable firewalld
[sms]# systemctl stop firewalld
```

3.2 OpenHPC Components Installation for Head Node

After we build and boot up the BOS, we will start to install and setup OpenHPC environment on our SMS host. The SMS will setup provisioning services, resource management services and related utilities.

The SMS host will be the head node in OpenHPC Cluster and provide the stateless environment for other compute nodes.

3.2.1. Setup Local OpenHPC Repository

To begin, enable use of the OpenHPC repository by adding it to the local list of available package repositories.

Note that this requires network access from your master server to the OpenHPC repository, or alternatively, that the OpenHPC repository be mirrored locally. In cases where network external connectivity is available, OpenHPC provides an ohpc-release package that includes GPG keys for package signing and repository enablement. The example which follows illustrates installation of the ohpc-release package directly from the OpenHPC build server.

```
[sms]# yum install http://build.openhpc.community/OpenHPC:/1.3/CentOS_7/x86_64/ohpc-release-1.3-1.el7.x86_64.rpm
```

Besides of OpenHPC package repository, we also need base OS distro repositories and EPEL repositories. They are available from the following web site.

- CentOS-7 - Base 7.4.1708 (http://mirror.centos.org/centos-7/7/os/x86_64)
- EPEL 7 (http://download.fedoraproject.org/pub/epel/7/x86_64)

Once we have setup the OpenHPC repository, we can start to install required packages from it. We will install OpenHPC Common Base Packages by following command.

```
# Install OpenHPC Base Packages
[sms]# yum -y install ohpc-base
```

3.2.2. Setup Provisioning Services (Warewulf)

For Saleable HPC Environment, we need to provide provisioning services. We will install Warewulf provisioning system which is designed for scalable system management including frameworks for system configuration, management provisioning/installation, monitoring, event notification, and so on. Please refer to following command to install the OpenHPC Warewulf packages.

```
# Install OpenHPC Warewulf Packages
[sms]# yum -y install ohpc-warewulf

# Initialize OpenHPC Warewulf Database and ssh_keys
[sms]# wwinit database
[sms]# wwinit ssh_keys



# Export /home and OpenHPC Public Packages from SMS host
[sms]# echo "/home *(rw,no_subtree_check,fsid=10,no_root_squash)" >> /etc/exports
[sms]# echo "/opt/ohpc/pub *(ro,no_subtree_check,fsid=11)" >> /etc/exports
[sms]# exportfs -a
[sms]# systemctl restart nfs-server
[sms]# systemctl enable nfs-server


# Import Files from SMS host to Compute Node Image (Warewulf Database)
[sms]# wwsh file import /etc/passwd
[sms]# wwsh file import /etc/group
[sms]# wwsh file import /etc/shadow

[sms]# wwsh file import /opt/ohpc/pub/examples/network/centos/ifcfg-ib0.ww
[sms]# wwsh -y file set ifcfg-ib0.ww --path=/etc/sysconfig/network-scripts/ifcfg-ib0


# Set provisioning interface as the default networking device
[sms]# echo "GATEWAYDEV=${eth_provision}" > /tmp/network.$$
[sms]# wwsh -y file import /tmp/network.$$ --name network

[sms]# wwsh -y file set network --path /etc/sysconfig/network --mode=0644 --uid=0
```

3.2.3. Setup Resource Management Services (PBS Pro)

PBS Professional (PBS Pro) software provide resource management services including workload management and optimizes job scheduling in HPC environments. We can install PBS Pro by following commands.

```
# Install OpenHPC PBS Pro Packages
[sms]# yum -y install pbspro-server-ohpc
```

3.2.4. Setup Omni-Path Support Services

Omni-Path (also Omni-Path Architecture, abbr. OPA) is a high-performance communication architecture owned by Intel. It offers low communication latency, low power consumption and a high throughput. Please use following command to add Omni-Path support.

```
# Install Intel Omni-Path Support
```

```
[sms]# yum -y install opensm-libs libibcm rdma-core-devel irqbalance libstdc++-devel
opensm-libs papi elfutils-libelf-devel librdmacm sysfsutils openssl atlas qperf perftest
infinipath-psm libgfortran gcc-gfortran libquadmath opa-basic-tools
[sms]# systemctl start rdma

# Install Intel Omni-Path IFS Software Pack
[sms]# tar xvfz IntelOPA-IFS.RHEL74-x86_64.10.6.1.0.2.tgz
[sms]# cd IntelOPA-IFS.RHEL74-x86 64.10.6.1.0.2
[sms IntelOPA-IFS.RHEL74-x86_64.10.6.1.0.2]# ./INSTALL -i opa -i ipoib -i psm_mpi -i
verbs_mpi -i pgas -i opadev -i hfi1_uefi -i fastfabric -i opafm
[sms IntelOPA-IFS.RHEL74-x86 64.10.6.1.0.2]# ./INSTALL -s
```

3.2.5. Setup Cluster Command Tool (ClusterShell)

ClusterShell provides a light, unified and robust command execution Python framework, which designed to run local or distant commands in parallel on server farms or on large Linux clusters environment. Please refer to following command to install and setup the ClusterShell package.

```
# Install ClusterShell
[sms]# yum -y install clustershell-ohpc

# Setup node definitions
[sms]# cd /etc/clustershell/groups.d
[sms]# mv local.cfg local.cfg.orig
[sms]# echo "adm: ${sms name}" > local.cfg
[sms]# echo "compute: ${compute prefix}[1-${num computes}]" >> local.cfg
[sms]# echo "all: @adm,@compute" >> local.cfg
```

3.2.6. Setup Cluster Configuration Tool (genders)

Genders is a static cluster configuration database used for cluster configuration management. It describes the layout and configuration of the cluster so that tools and scripts can sense the variations of cluster nodes. Please refer to following command to install and setup two genders, compute and bmc.

```
# Install OpenHPC genders
[sms]# yum -y install genders-ohpc

# Generate a sample genders file
[sms]# echo -e "${sms name}\tsms" > /etc/genders
[sms]# for ((i=0; i<$num_computes; i++)) ; do
echo -e "${c_name[$i]}\tcompute,bmc=${c_bmc[$i]}"
done >> /etc/genders
```

3.2.7. Setup Console Manager (ConMan)

ConMan is a serial console management program designed to support a large number of console devices and simultaneous users. It supports multi types of consoles connection including IPMI serial-over-lan for compute node consoles. Please refer to following command to install and setup ConMan.

```
# Install ConMan to provide a front-end to compute consoles and log output
[sms]# yum -y install conman-ohpc

# Configure conman for computes (note your IPMI password is required for console access)
[sms]# for ((i=0; i<$num computes; i++)) ; do
echo -n 'CONSOLE name="'${c_name[$i]}'" dev="ipmi:'${c_bmc[$i]}'" '
echo 'ipmiopts="'U:${bmc_username},P:${IPMI_PASSWORD:-undefined},W:solpayloadsize'"'
done >> /etc/conman.conf
```

```
# Enable and start conman
[sms]# systemctl enable conman
[sms]# systemctl start conman
```

3.2.8. Setup Node Health Check (NHC)

Node Health Check (NHC) is a tool for determining the health status of a node, including node misconfiguration, failure situations or hardware failures. NHC can mark “unhealthy” nodes offline and prevent jobs from failure. NHC can helps to increase the reliability and throughput of jobs for our OpenHPC Cluster environment.

```
# Install NHC on master and compute nodes
[sms]# yum -y install nhc-ohpc
[sms]# yum -y --installroot=$CHROOT install nhc-ohpc

# Generate NHC configuration file based on compute node environment
[sms]# pdsh -w cn[01-04] "/usr/sbin/nhc-genconf -H '*' -c -" | dshbak -c >
/etc/nhc/nhc.conf.new
```

3.2.9. Setup Monitoring Tool – Nagios

Nagios is an open source infrastructure monitoring package that monitors servers, switches, applications, and services and offers user-defined alerting facilities. As provided by OpenHPC, it consists of a base monitoring daemon and a set of plug-ins for monitoring various aspects of an HPC cluster. The following commands can be used to install and configure a Nagios server on the SMS host and add the facility to run tests and gather metrics from provisioned compute nodes.

```
# Install NHC on master and compute nodes
[sms]# yum -y install nhc-ohpc
[sms]# yum -y --installroot=$CHROOT install nhc-ohpc

# Install Nagios meta-package on master host
[sms]# yum -y install ohpc-nagios

# Install plugins into compute node image
[sms]# yum -y --installroot=$CHROOT install nagios-plugins-all-ohpc nrpe-ohpc

# Enable and configure NRPE in compute image
[sms]# chroot $CHROOT systemctl enable nrpe
[sms]# perl -pi -e "s/^allowed_hosts=/# allowed_hosts=/" $CHROOT/etc/nagios/nrpe.cfg
[sms]# echo "nrpe 5666/tcp # NRPE" >> $CHROOT/etc/services
[sms]# echo "nrpe : ${sms_ip} : ALLOW" >> $CHROOT/etc/hosts.allow
[sms]# echo "nrpe : ALL : DENY" >> $CHROOT/etc/hosts.allow
[sms]# chroot $CHROOT /usr/sbin/groupadd -r nrpe
[sms]# chroot $CHROOT /usr/sbin/useradd -c "NRPE user for the NRPE service" -d
/var/run/nrpe \
-r -g nrpe -s /sbin/nologin nrpe

# Configure remote services to test on compute nodes
[sms]# mv /etc/nagios/conf.d/services.cfg.example /etc/nagios/conf.d/services.cfg

# Define compute nodes as hosts to monitor
[sms]# mv /etc/nagios/conf.d/hosts.cfg.example /etc/nagios/conf.d/hosts.cfg
[sms]# for ((i=0; i<$num_computes; i++)) ; do
perl -pi -e "s/HOSTNAME$(($i+1))/${c_name[$i]}/ || s/HOST$(($i+1)) IP/${c_ip[$i]}/" \
/etc/nagios/conf.d/hosts.cfg
done

# Update location of mail binary for alert commands
[sms]# perl -pi -e "s/ \/bin\/mail/ \/usr\/bin\/mailx/g"
/etc/nagios/objects/commands.cfg

# Update email address of contact for alerts
[sms]# perl -pi -e "s/nagios\@localhost/root\@${sms_name}/"
/etc/nagios/objects/contacts.cfg
```

```
# Add check_ssh command for remote hosts
[sms]# echo command[check_ssh]=/usr/lib64/nagios/plugins/check_ssh localhost \
>> $CHROOT/etc/nagios/nrpe.cfg

# define password for nagiosadmin to be able to connect to web interface
[sms]# htpasswd -bc /etc/nagios/passwd nagiosadmin ${nagios_web_password}

# Enable Nagios on master, and configure
[sms]# chkconfig nagios on
[sms]# systemctl start nagios
[sms]# chmod u+s `which ping`
```

3.2.10. Setup Monitoring Tool - Ganglia

Ganglia is a scalable distributed system monitoring tool for high-performance computing systems such as clusters and grids. It allows the user to remotely view live or historical statistics (such as CPU load averages or network utilization) for all machines running the gmond daemon. The following commands can be used to enable Ganglia to monitor both the master and compute hosts.

```
# Install Ganglia meta-package on master
[sms]# yum -y install ohpc-ganglia

# Install Ganglia compute node daemon
[sms]# yum -y --installroot=$CHROOT install ganglia-gmond-ohpc

# Use example configuration script to enable unicast receiver on master host
[sms]# cp /opt/ohpc/pub/examples/ganglia/gmond.conf /etc/ganglia/gmond.conf
[sms]# perl -pi -e "s/<sms>/${sms name}/" /etc/ganglia/gmond.conf

# Add configuration to compute image and provide gridname
[sms]# cp /etc/ganglia/gmond.conf $CHROOT/etc/ganglia/gmond.conf
[sms]# echo "gridname MySite" >> /etc/ganglia/gmetad.conf

# Start and enable Ganglia services
[sms]# systemctl enable gmond
[sms]# systemctl enable gmetad
[sms]# systemctl start gmond
[sms]# systemctl start gmetad
[sms]# chroot $CHROOT systemctl enable gmond

# Restart web server
[sms]# systemctl try-restart httpd
```

3.2.11. Setup Network Time Protocol (NTP)

The Network Time Protocol (NTP) is used to synchronize the time from NTP services server. HPC cluster need synchronized clock to work together, and we need to setup and enable NTP services on SMS host. The other compute nodes can synchronize with SMS host and keep the same clock. Please refer to the following command to setup and enable NTP services on SMS host.

```
# Enable and Setup NTP Services
[sms]# systemctl enable ntpd.service
[sms]# echo "server ${ntp_server}" >> /etc/ntp.conf
[sms]# systemctl restart ntpd
```

3.2.12. Unlock Memory Limits

Unlock memory limits and increase the cluster performance on SMS host.

```
# Unlock Memory Limits
[sms]# perl -pi -e 's/# End of file/\* soft memlock unlimited\n$&/s'
/etc/security/limits.conf
[sms]# perl -pi -e 's/# End of file/\* hard memlock unlimited\n$&/s'
```

```
/etc/security/limits.conf
```

3.2.13. Enable Forwarding of System Logs

Enable receiving messages services from compute nodes on SMS host. Please refer to the following steps to enable receiving system logs on SMS host.

```
# Configure SMS to receive messages and reload rsyslog configuration
[sms]# perl -pi -e "s/\\#\\\$ModLoad imudp/\\\$ModLoad imudp/" /etc/rsyslog.conf
[sms]# perl -pi -e "s/\\#\\\$UDPServerRun 514/\\\$UDPServerRun 514/" /etc/rsyslog.conf
[sms]# systemctl restart rsyslog
```

3.3 Provisioning Installation for Compute Nodes

Once we have setup and enable provisioning services on SMS host, we need to prepare provisioning environment for Compute Nodes. We need to build Based Operation System (BOS), Virtual Node File System(VNFS), bootstrap Image and setup the configuration for Compute Nodes.

3.3.1. Setup Compute Node BOS Image

The OpenHPC build of Warewulf provide the tool, wwmkchroot, to build a BOS for Compute Nodes. Please refer to following commands to build a BOS files for Compute Nodes.

```
# Define Image Based chroot Location on SMS host
[sms]# export CHROOT=/opt/ohpc/admin/images/centos7.4

# Build initial chroot files
[sms]# wwmkchroot centos-7 $CHROOT
```

3.3.2. Setup OpenHPC Components

Once the Compute Node BOS build, we can install and setup OpenHPC environment on it.

```
# Install HPC Base Package for Compute Node
[sms]# yum -y --installroot=$CHROOT install ohpc-base-compute
# Copy SMS DNS Config File to Compute Node
[sms]# cp -p /etc/resolv.conf $CHROOT/etc/resolv.conf
```

3.3.3. Setup Provision Client Environment

After BOS and OpenHPC Installed, we need to setup provision files for Compute Nodes to access SMS host resources.

```
# Add NFS client mounts of /home and /opt/ohpc/pub to base image
[sms]# echo "${sms ip}:/home /home nfs nfsvers=3,nodev,nosuid,noatime 0 0" >>
$CHROOT/etc/fstab
[sms]# echo "${sms_ip}:/opt/ohpc/pub /opt/ohpc/pub nfs nfsvers=3,nodev,noatime 0 0" >>
$CHROOT/etc/fstab

# Enable Network Time Protocol (NTP) and Assign SMS host NTP Service for Compute Node
[sms]# echo "server ${sms_ip}" >> $CHROOT/etc/ntp.conf
[sms]# chroot $CHROOT systemctl enable ntpd
[sms]# yum -y --installroot=$CHROOT install ntp

# Add PBS Professional client support for Compute Node Image
```

```
[sms]# yum -y --installroot=$CHROOT install pbspro-execution-ohpc
[sms]# perl -pi -e "s/PBS_SERVER=\S+/PBS_SERVER=${sms_name}/" $CHROOT/etc/pbs.conf
[sms]# echo "PBS_LEAF_NAME=${sms_name}" >> /etc/pbs.conf
[sms]# chroot $CHROOT opt/pbs/libexec/pbs_habitat
[sms]# perl -pi -e "s/\$clienthost \S+/\$clienthost ${sms_name}/"
$CHROOT/var/spool/pbs/mom_priv/config
[sms]# echo "\$usecp *:/home /home" >> $CHROOT/var/spool/pbs/mom_priv/config
[sms]# chroot $CHROOT systemctl enable pbs

# Add kernel drivers to Compute Node Image
[sms]# yum -y --installroot=$CHROOT install kernel

# Include modules user environment in Compute Node Image
[sms]# yum -y --installroot=$CHROOT install lmod-ohpc
```

3.3.4. Setup Resource Management Client Environment

Once the BOS setup, we need to install and setup OpenHPC environment for Compute Node Image.

```
# Add PBS Professional client support for Compute Node Image
[sms]# yum -y --installroot=$CHROOT install pbspro-execution-ohpc
[sms]# perl -pi -e "s/PBS SERVER=\S+/PBS SERVER=${sms name}/" $CHROOT/etc/pbs.conf
[sms]# echo "PBS_LEAF_NAME=${sms_name}" >> /etc/pbs.conf
[sms]# chroot $CHROOT opt/pbs/libexec/pbs_habitat
[sms]# perl -pi -e "s/\$clienthost \S+/\$clienthost ${sms_name}/"
$CHROOT/var/spool/pbs/mom priv/config
[sms]# echo "\$usecp *:/home /home" >> $CHROOT/var/spool/pbs/mom priv/config
[sms]# chroot $CHROOT systemctl enable pbs

# Add Network Time Protocol (NTP) support to Compute Node Image
[sms]# yum -y --installroot=$CHROOT install ntp

# Add kernel drivers to Compute Node Image
[sms]# yum -y --installroot=$CHROOT install kernel

# Include modules user environment in Compute Node Image
[sms]# yum -y --installroot=$CHROOT install lmod-ohpc
```

3.3.5. Setup Omni-Path Support Environment

Please refer to the follow steps to install and setup Omni-Path support for the Compute Node Image.

```
# Add Omni-Path Drivers for Compute Node Image
[sms]# yum -y --installroot=$CHROOT install opensm-libs libibcm rdma-core-devel
irqbalance libstdc++-devel opensm-libs papi elfutils-libelf-devel librdmacm sysfsutils
openssl atlas qperf perftest infinipath-psm libgfortran gcc-gfortran libquadmath opa-
basic-tools
[sms]# yum -y --installroot=$CHROOT install ssf-orch-rhel73-2016.0-22.4.x86_64.rpm
[sms]# cd $CHROOT/root
[sms centos7.4]# tar xvfz /root/IntelOPA-IFS.RHEL74-x86 64.10.6.1.0.2.tgz
[sms centos7.4]# chroot $CHROOT <<  EndOfCommands
cd /root/IntelOPA-IFS.RHEL74-x86 64.10.6.1.0.2
./INSTALL -i opa -i ipoib -i psm_mpi -i verbs_mpi -i pgas -i opadev -i hfi1_uefi -i
fastfabric
./INSTALL -s
 EndOfCommands

[sms centos7.4]# cat >>/etc/warewulf/bootstrap.conf <<_EndOfFile_
drivers += updates, extra/ifs-kernel-updates, hfi1, rdmavt
firmware += updates/hfi1*
 EndOfFile
[sms centos7.4]# cd -
```

3.3.6. Unlock Client Memory Limits

Unlock memory limits and increase the cluster performance for Compute Note Image.

```
# Unlock Memory Limits for Compute Note Image
[sms]# perl -pi -e 's/# End of file/\* soft memlock unlimited\n$&/s'
$CHROOT/etc/security/limits.conf
[sms]# perl -pi -e 's/# End of file/\* hard memlock unlimited\n$&/s'
$CHROOT/etc/security/limits.conf
```

3.3.7. Enable Forwarding of System Logs

Enable forwarding of system logs from compute nodes to head node (SMS) host. Please refer to the following steps to enable forwarding setup on Compute Note Image.

```
# Define Compute Node forwarding destination
[sms]# echo "*.* @${sms_ip}:514" >> $CHROOT/etc/rsyslog.conf

# Disable most local logging on computes. Emergency and boot logs will remain on the
Compute Nodes
[sms]# perl -pi -e "s/^\*\.info/\\#\*\.info/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^authpriv/\\#authpriv/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^mail/\\#mail/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^cron/\\#cron/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^uucp/\\#uucp/" $CHROOT/etc/rsyslog.conf
```

3.4 Warewulf Provisioning Services Startup

Before we start the provisioning services, we need to setup configuration for Compute Nodes. Then prepare the bootstrap and Virtual Node File System (VNFS) Image for Compute Nodes.

3.4.1. Provisioning Configuration

In preparation for provisioning, we can now define the desired network settings for four Compute Nodes with the underlying provisioning system and restart the DHCP service. Please make sure that the system environment variables have been setup according to your system required. (Please refer to the §2 for related system environment variables meaning and example values) By default, Warewulf uses network interface names of the eth# variety and adds kernel boot arguments to maintain this scheme on newer kernels. Consequently, when specifying the desired provisioning interface via the $eth provision variable, it should follow this convention. Alternatively, if you prefer to use the predictable network interface naming scheme (e.g. names like enp61s0f0), additional steps are included to alter the default kernel boot arguments and take the eth# named interface down after bootstrapping so the normal init process can bring it up again using the desired name.

```
# Add Compute Nodes to Warewulf Data Store
[sms]# for ((i=0; i<$num computes; i++)) ; do
          wwsh -y node new ${c name[i]} --ipaddr=${c ip[i]} --hwaddr=${c mac[i]} -D
${eth provision}
      done
```

Warewulf uses network interface names of the eth# variety and adds kernel boot arguments by default. If we are not use eth# as network interface name, please use following commands to set up network interface name by system environment variable - ${eth_provision}.

```
# Additional step required if desiring to use predictable network interface
# naming schemes (e.g. enp61s0f0). Skip if using eth# style names.
[sms]# wwsh provision set "${compute_regex}" --kargs "net.ifnames=1,biosdevname=1"
[sms]# wwsh provision set --postnetdown=1 "${compute_regex}"
```

Warewulf can help provisioning to Compute Nodes from Head Node (SMS) host. We need to setup bootstrap image and Virtual Node File System (VNFS) Image for provisioning to Compute Nodes. Please refer to the following commands to setup VNFS, bootstrap Image, related files and IPoIB Network setting for our Compute Nodes.

```
# Define Provisioning File System and Bootstrap Image for Compute Nodes
[sms]# wwsh -y provision set "${compute_regex}" --vnfs=centos7.4 --bootstrap=`uname -r`
\ --files=dynamic_hosts,passwd,group,shadow,network

# Define IPoIB Network Settings for Compute Nodes
[sms]# for ((i=0; i<$num computes; i++)) ; do
        wwsh -y node set ${c_name[$i]} -D ib0 --ipaddr=${c_ipoib[$i]} --
netmask=${ipoib_netmask}
     done
[sms]# wwsh -y provision set "${compute_regex}" --fileadd=ifcfg-ib0.ww

# Restart dhcp / update PXE
[sms]# systemctl restart dhcpd
[sms]# wwsh pxe update
```

3.4.2. Bootstrap Image

Warewulf provide wwbootstrap tool to build bootstrap image for Compute Nodes Provisioning. Compute Node can boot over PXE and get the bootstrap image from Head Node (SMS) host. The bootstrap image includes kernel, associated modules and scripts to help Compute Node BOS boot up. Please refer to the following steps to build a bootstrap image for Compute Node.

```
# (Optional) Include drivers from kernel updates; needed if enabling additional kernel
modules on computes
[sms]# export WW_CONF=/etc/warewulf/bootstrap.conf
[sms]# echo "drivers += updates/kernel/" >> $WW_CONF

# (Optional) Include overlayfs drivers; needed by Singularity
[sms]# echo "drivers += overlay" >> $WW CONF

# Build bootstrap image
[sms]# wwbootstrap `uname -r`
```

3.4.3. Virtual Node File System (VNFS) Image

Warewulf can help to stateless provisioning to compute nodes. Warewulf provide a wwvnfs tool to build Virtual Node File System (VNFS) Image for compute nodes. The advantage of stateless provisioning is that the file system image is consistent across all compute nodes in the cluster. It will be more easy and scalable for cluster administration.

Please refer to the following steps to build VNFS Image for compute node provisioning in UEFI mode.

```
# Add GRUB2 Bootloader for Compute Nodes
[sms]# yum -y --installroot=$CHROOT install grub2-efi grub2-efi-modules

# Copy and Setup Warewulf EFI Filesystem Layout Example
[sms]# cp /etc/warewulf/filesystem/examples/efi_example.cmds
/etc/warewulf/filesystem/efi.cmds
[sms]# wwsh provision set --filesystem=efi "${compute_regex}"
[sms]# wwsh provision set --bootloader=sda "${compute_regex}"

# Build VNFS image for Compute Nodes
[sms]# wwvnfs --chroot $CHROOT
```

3.4.4. Boot Compute Nodes

Once we have built bootstrap and VNFS Image for Compute Nodes, we may reboot Compute Nodes through ipmitool tool as following commands. As Warewulf stateless provisioning require compute node boot over PXE, please make sure each compute node's BIOS setting are configured to boot over PXE in high periority.

```
# Reboot Compute Nodes through ipmitool
[sms]# for ((i=0; i<${num_computes}; i++)) ; do
          ipmitool -E -I lanplus -H ${c_bmc[$i]} -U ${bmc_username} chassis power reset
       done
```

Compute Nodes boot process may take few minutes depends on different model environment. We can use parallel shell tool, pdsh, to check the uptime status for compute nodes. Please refer to following command to check the uptime status for compute nodes.

```
# Check uptime status for Compute Nodes by Parallel Shell Tool
[sms]# pdsh -w cn[01-04] uptime
cn04:  22:09:17 up 0:06,  0 users,  load average: 0.00, 0.01, 0.05
cn03:  22:09:17 up 0:06,  0 users,  load average: 0.00, 0.01, 0.05
cn02:  22:09:17 up 0:06,  0 users,  load average: 0.00, 0.01, 0.05
cn01:  22:09:17 up 0:06,  0 users,  load average: 0.00, 0.01, 0.05
```

3.5 OpenHPC Development Components

3.5.1. Development Tools

OpenHPC provides many development tools. Please refer to the following steps to install some OpenHPC Development Tools as an example.

```
# Install OpenHPC autotools, EasyBuild, Spack Packages ...
[sms]# yum -y install ohpc-autotools
[sms]# yum -y install EasyBuild-ohpc
[sms]# yum -y install hwloc-ohpc
[sms]# yum -y install spack-ohpc
[sms]# yum -y install valgrind-ohpc
```

3.5.2. Compilers

Following example is to install some compilers for your OpenHPC environment. You can optional install these compilers as required. Please refer to the following command to install GNU and llvm compiler.

```
# Install GNU Compiler
[sms]# yum -y install gnu7-compilers-ohpc

# Install llvm compiler
[sms]# yum -y install llvm5-compilers-ohpc
```

3.5.3. OpenMPI

There are variety of MPI families available on OpenHPC platform. In this implementation guide, we will install OpenMPI. Please refer to the following steps to install some OpenHPC Development Tools as an example.

```
# Install OpenHPC autotools, EasyBuild, Spack Packages ...
[sms]# yum -y install openmpi3-gnu7-ohpc mpich-gnu7-ohpc
```

3.5.4. Performance Tools

OpenHPC provides many development tools. Please refer to the following steps to install some OpenHPC Development Tools as an example.

```
# Install OpenHPC autotools, EasyBuild, Spack Packages ...
[sms]# yum -y install ohpc-autotools
[sms]# yum -y install EasyBuild-ohpc
[sms]# yum -y install hwloc-ohpc
[sms]# yum -y install spack-ohpc
[sms]# yum -y install valgrind-ohpc
```

3.5.5. Default Development Environment

OpenHPC provides many development tools. Please refer to the following steps to install some OpenHPC Development Tools as an example.

```
# Install OpenHPC autotools, EasyBuild, Spack Packages ...
[sms]# yum -y install ohpc-autotools
[sms]# yum -y install EasyBuild-ohpc
[sms]# yum -y install hwloc-ohpc
[sms]# yum -y install spack-ohpc
[sms]# yum -y install valgrind-ohpc
```

3.6 Intel® Parallel Studio EX

ADVISORY: Support for Intel® Parallel Studio XE 2018.
Intel® Parallel Studio XE 2018 is not a component of the version of Intel® HPC Orchestrator Advanced used. Please direct all support questions directly to Intel® Parallel Studio support. Some applications included with Intel® HPC Orchestrator Advanced may not compile or run correctly with Intel® Parallel Studio XE 2018. These applications are using compiler or environment options that are no longer supported. Application source must be updated to replace discontinued or deprecated options. To find a list of these options, refer to Intel® Cluster Tools Deprecation Information and Deprecated and Removed Compiler Options.
Please refer to following steps to install and setup Intel® Parallel Studio Ex software.

```
# Extract Intel® Parallel Studio
[sms]# tar -zxvf parallel_studio_xe_2018_cluster_edition.tgz -C /usr/src

# Setup Intel® Parallel Studio License
[sms]# cp /root/psxe2018.lic /opt/intel/licenses

# Setup Intel® Parallel Studio Configuration
# update default install folder to /opt/ohpc/pub/intel folder
[sms]# sed -i \
-e "/^ACCEPT_EULA/ s/decline/accept/" \
-e "/^PSET INSTALL DIR/ s/\/opt\/intel/\/opt\/ohpc\/pub\/intel" \
-e "/^ARCH_SELECTED/ s/ALL/INTEL64/" \
-e "/^MPSS/ s/yes/no/" \
-e "/^ACTIVATION_TYPE/ s/exist_lic/license_file/" \
-e "s%#\(ACTIVATION_LICENSE_FILE=\)%\1/opt/intel/licenses/psxe2018.lic%" \
/usr/src/parallel_studio_xe_2018_cluster_edition/silent.cf

# Install Intel® Parallel Studio with Configuration File
[sms]# cd /usr/src/parallel studio xe 2018 cluster edition/
[sms parallel_studio_xe_2018_cluster_edition]# ./install.sh -s silent.cfg
[sms parallel_studio_xe_2018_cluster_edition]# cd ~
```

3.6.1. Test Job

After install and setup OpenHPC Cluster environment, we may try to test it by running demo MPI test program. We will create a test user account and test the environment in user level. You may add the user by following command if the account not exists yet.

```
# Add test user account for running test job
[sms]# useradd -m test
```

3.6.1.1. Interactive Execution

We will test cluster environment in user level (substitute test user account). Compile and run a MPI demo program. Submit the request to PBS resource manager services and run the MPI demo program. Refer to the following steps and demo program results.

```
# Switch to "test" user
[sms]# su - test

# Compile MPI "hello world" example
[test@sms ~]$ mpicc -O3 /opt/ohpc/pub/examples/mpi/hello.c

# Submit interactive job request
[test@sms ~]$ qsub -I -l select=3:mpiprocs=4
qsub: waiting for job 1.sms to start
qsub: job 1.sms ready

# PBS Pro will help to switch to first node, eg. cn01
# Use prun to launch MPI program from cn01 host
[prun] Master compute host = cn01
[prun] Resource manager = pbspro
[prun] Launch cmd = mpiexec -x LD_LIBRARY_PATH --prefix /opt/ohpc/pub/mpi
/openmpi3-gnu7/3.0.0 --hostfile /var/spool/pbs/aux/22.sms ./a.out (family
=openmpi3)

 Hello, world (12 procs total)
    --> Process #   0 of  12 is alive. -> cn01.cluster
    --> Process #   2 of  12 is alive. -> cn01.cluster
```

```
    --> Process #   3 of  12 is alive. -> cn01.cluster
    --> Process #   1 of  12 is alive. -> cn01.cluster
    --> Process #   4 of  12 is alive. -> cn02.cluster
    --> Process #   5 of  12 is alive. -> cn02.cluster
    --> Process #   8 of  12 is alive. -> cn03.cluster
    --> Process #   6 of  12 is alive. -> cn02.cluster
    --> Process #   9 of  12 is alive. -> cn03.cluster
    --> Process #   7 of  12 is alive. -> cn02.cluster
    --> Process #  10 of  12 is alive. -> cn03.cluster
    --> Process #  11 of  12 is alive. -> cn03.cluster
[test@cn01 ~]$
```

3.6.1.2. Batch Execution

We can refer to PBS example job script and modify the argument as required. Submit the job script request to PBS resource manager services and it will schedule and run the MPI demo program. Refer to the following steps.

```
# copy PBS example job script from /opt/ohpc/pub/examples/pbspro/job.mpi
[test@sms ~]$ cp /opt/ohpc/pub/examples/pbspro/job.mpi .

# Examine contents (and edit to set desired job sizing characteristics)
[test@sms ~]$ cat job.mpi
#!/bin/bash
#----------------------------------------------------------
# Job name
#PBS -N test

# Name of stdout output file
#PBS -o job.out

# Total number of nodes and MPI tasks/node requested
#PBS -l select=2:mpiprocs=4

# Run time (hh:mm:ss) - 1.5 hours
#PBS -l walltime=01:30:00
#----------------------------------------------------------
# Change to submission directory
cd $PBS O WORKDIR
# Launch MPI-based executable
prun ./a.out

# Submit PBS job for batch execution
[test@sms ~]$ qsub job.mpi
3.sms
```

4. Validation and Verification

4.1. Review the Validation Components

The validation process requires having the following software components installed.

• Intel® MPI Library version 2018.0 or higher

• Intel® Math Kernel Library version 2018.0 or higher

• Intel® Cluster Checker version 2018.3 or higher

• Intel® Cluster Checker extensions for Intel® Select Solutions for Simulation and Modeling

The 2018.0 version of both Intel® MPI Library and Intel® Math Kernel Library are supplied in this implementation guide via Intel® Parallel Studio XE Cluster Edition 2018.0. A standalone installation of Intel® Cluster Checker 2018.2 and its extensions provide the latter two components listed above. Intel® Cluster Checker is used extensively in validation. Intel® Cluster Checker is a tool that inspects a comprehensive range of characteristics that indicate cluster health. It examines the system at both the node and cluster level, making sure all components work together and can deliver optimal performance. Intel® Cluster Checker separates the tasks of data collection and analysis into two separate tools. As such, the typical use model of Intel® Cluster Checker is an iterative 3-step process: collect system data, analyze system data, and fix diagnosed issues. Once issues are fixed, it is necessary to start a new cycle of the process by collecting new system data. As a means of extension, Intel® Cluster Checker supports framework definitions (FWDs). FWDs allow the specification of collection and analysis components to include when using the tool. The extensions for Intel® Select Solutions for Simulation and Modeling are implemented as framework definitions.

4.2. Assess Cluster Health

Use Intel® Cluster Checker (CLCK) to diagnose general cluster health issues.

4.2.1. Modify the Intel® Cluster Checker Configuration File

A default configuration file is provided with the installation of Intel® CLCK. If no configuration file is specified when Intel® CLCK is run, the default configuration is used. Modify the configuration file with some general options as well as some options specific to this implementation guide. Review the inline comments in the file below for justifications of the specified options.

Open the /opt/intel/clck/2018.2/etc/clck.xml file for editing.

Replace the file contents with the configuration given below

```
# Modify the Intel® Cluster Checker Configuration File
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<analyzer>
<config>
</config>
<suppressions>
```

```
<suppress>
<!-- This sign should be suppressed unless using the tool to collect and
analyze low-level hardware data that is only available to
privileged users. -->
<id>dmidecode-insufficient-privileges</id>
</suppress>
<suppress>
<!-- In CLCK 2018.2, there is a known issue that nodes without an
Intel(R) Omni-Path HFI adapter will still check for the
presence of a fabric subnet manager. This sign is suppressed only
on the head node since it does not have an HFI. -->
<node_id>frontend</node_id>
<id>opa-subnet-manager-not-running</id>
</suppress>
</suppressions>
</analyzer>
<collector>
<!-- Specify the head node private network interface. -->
<network_interface>eno1</network_interface>
</collector>
<database>
</database>
</configuration>
```

Save the file and exit.

4.2.2. Switch to the test user

To run Intel® CLCK using the cluster health framework definition, switch to the test user.

The tool requires a nodefile containing the hostnames of nodes to be checked. This nodefile is easily generated by reusing the genders node data. For health assessment, the nodefile is populated with the hostnames of all cluster nodes including the head node.

To run these Intel® Cluster Checker extensions, switch to the test user. The command below will overwrite the nodefile previously used when assessing cluster health. You may wish to verify the newly generated nodefile contains only the hostnames of four compute nodes before proceeding with data collection.

```
# Switch to the test user
[sms]# su - test

# Create the nodefile
[test@sms]# echo "$(nodeattr -c sms) # role:head" > nodefile

[test@sms]# echo "$(nodeattr -c compute | tr ',' '\n')" >> nodefile

# Source the environment script for Intel® CLCK
[test@sms]# source /opt/intel/clck/2018.2/bin/clckvars.sh
```

4.2.3. Collect Data to Assess Cluster Health

To access the compute nodes for data collection, it is necessary to first gain access by obtaining an allocation through Slurm*. Run these commands to obtain an allocation, collect data from all nodes using the cluster health framework definition, and then relinquish the allocation.

```
# Collect Data to Assess Cluster Health
salloc --nodelist=$(nodeattr -c compute)
```

```
clck-collect -f nodefile -F health

exit
```

4.2.4. Assess Cluster Health

Now assess the health of the cluster by executing the Intel® CLCK analyze tool using the cluster health framework definition.

```
# Assess Cluster Health
clck-analyze -f nodefile -F health -p diagnosed_signs
```

The analyzer will print a list of diagnoses, diagnosed signs, and undiagnosed signs. Fix issues before continuing validation. For a healthy system, the analyzer will report "PASS: All checks passed." in the final lines of output.

4.3. Assess Validation Requirements as Privileged User

To assess validation requirements with Intel® Cluster Checker, there are two phases. The first phase requires execution of Intel® Cluster Checker as a privileged user to collect hardware data.

4.3.1. Switch to the privileged User

To run Intel® Cluster Checker, return to root user by logging out of test user.

```
# Switch to the Privileged User
logout
```

Unlike the health checks, the nodefile used in this section contains the hostnames of the four compute nodes being used. The command below writes all compute node hostnames to file. Since this implementation guide is for a cluster with exactly four compute nodes, such a method is suitable.

```
# Create the nodefile
echo "$(nodeattr -c compute | tr ',' '\n')" > nodefile

# Source the environment script for Intel® CLCK

source /opt/intel/clck/2018.2/bin/clckvars.sh
```

4.3.2. Specify a shared directory for temporary Intel® CLCK files

Intel® CLCK requires a shared directory for housing temporary application files. Ordinarily, a user home directory satisfies this requirement. Since the root user does have a shared home directory, we must specify the shared directory by setting the CLCK_SHARED_TEMP_DIR environment variable. Use the test user home directory.

```
# Specify a shared directory for temporary Intel® CLCK files
export CLCK_SHARED_TEMP_DIR=/home/test
```

4.3.3. Collect Data to Assess the Validation Requirements as Privileged User

To collect validation requirement data, use the configuration file and framework definition that are installed with the extensions package. These files contains settings that are both specific to and necessary for validation. The commands below obtain an allocation in Slurm*, collect the validation requirements data, then relinquish the allocation.

```
# Collect data to assess the validation requirements
salloc --nodelist=$(nodeattr -c compute)

clck-collect -f nodefile -F select-solutions-for-hpc-priv \

-c /opt/intel/clck/2018.2/etc/clck_select-solutions-for-hpc.xml

exit
```

4.3.4. Assess Validation Requirements as a Privileged User

Now assess the validation requirements by running the Intel® CLCK analyze tool using the validation framework definition specific to privileged data collection.

```
# Assess validation requirements as a privileged user
clck-analyze -f nodefile -F select-solutions-for-hpc-priv \

-c /opt/intel/clck/2018.2/etc/clck_select-solutions-for-hpc.xml -p diagnosed_signs \

-o clck_select-solutions-for-hpc-priv.log
```

The analyzer highlights validation issues as diagnosed or undiagnosed signs. Analysis of a system that meets these validation requirements will trigger no diagnoses, diagnosed signs, or undiagnosed signs, and will also report “PASS: All checks passed.” in the final lines of output. Note that a number of informational signs should also be printed. Informational signs are not indicative of a validation issue; rather, they report information that is intended to be submitted for verification. The command above specifies to create a logfile of the output to submit for verification. The resultant file is named clck_select-solutions-for-hpc-priv.log. An example of this file for a validated system is given in Appendix A

4.4. Assess Validation Requirements as Normal User

The second phase checks that Intel® Scalable System Framework Reference Architecture requirements are fulfilled and that the minimum performance thresholds are met for DGEMM, HPL, IMB_PINGPONG, STREAM and HPCG. These tests must be run as user.

4.4.1. Switch to the test User

To run these Intel® Cluster Checker extensions, switch to the test user. The command below will overwrite the nodefile previously used when assessing cluster health. You may wish to verify the newly generated nodefile contains only the hostnames of four compute nodes before proceeding with data collection.

```
# Switch to the test user
su - test


# Create the nodefile
echo "$(nodeattr -c compute | tr ',' '\n')" > nodefile



# Source the environment script for Intel® CLCK
source /opt/intel/clck/2018.2/bin/clckvars.sh
```

4.4.2. Collect Data to Assess the Validation Requirements as test User

To collect validation requirement data, use the configuration file and framework definition that are installed with the extensions package. These files contain settings that are both specific to and necessary for validation. The commands below obtain an allocation in Slurm*, collect the validation requirements data, then relinquish the allocation.

```
# Collect data to assess the validation requirements
salloc --nodelist=$(nodeattr -c compute)

clck-collect -f nodefile -F select-solutions-for-hpc-user \

-c /opt/intel/clck/2018.2/etc/clck_select-solutions-for-hpc.xml

exit
```

4.4.3. Assess Validation Requirements as a Normal User

Now assess the validation requirements by running the Intel® CLCK analyze tool using the validation framework definition specific to normal (i.e. unprivileged) data collection.

```
# Assess validation requirements as a privileged user
clck-analyze -f nodefile -F select-solutions-for-hpc-user \

-c /opt/intel/clck/2018.2/etc/clck_select-solutions-for-hpc.xml -p diagnosed_signs \

-o clck_select-solutions-for-hpc-user.log
```

Again, analysis of a system that meets these validation requirements will trigger no diagnoses, diagnosed signs, or undiagnosed signs, and will also report “PASS: All checks passed.” in the final lines of output. More informational signs should be displayed for the purpose of verification. The command above specifies to create a logfile of the output to submit for verification. The resultant file is named clck_select-solutions-for-hpc-user.log. An example of this file for a validated system is given in Appendix B.

4.5.Submit Files for Verification

Through the validation process, the following documents are made available for verification purposes.

- /opt/intel/clck/2018.2/etc/clck_select-solutions-for-hpc.xml
- /root/clck_select-solutions-for-hpc-priv.log
- /home/test/clck_select-solutions-for-hpc-user.log

Follow the process outlined in Intel® Select Solutions for Simulation and Modeling Reference Design to submit these documents for verification.

Appendix A
Example Output When Validating as Privileged User

```
# A Example Output When Validating as Privileged User
Intel(R) Cluster Checker 2018 Update 3 (build 20171223)

Database: clck_default, SQLite file /root/clck_select-solutions-for-hpc.db.

4 checks requested: cpu, lspci_hfi_width, lspci_hfi_width_info, memory

Reading the database for the following checks:

   cpu...  done (0.04 seconds)

   lspci_hfi_width...  done (0.00156 seconds)

   lspci_hfi_width_info...  done (0.000999 seconds)

   memory...  done (0.0347 seconds)

Analyzing: 100%, rules completed/remaining: 7/0

Nodes being tested:

cn[01-04].cluster

0 diagnoses


0 diagnosed signs




0 undiagnosed signs


12 informational signs:

  1-4. CPU Model Name: "Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz"; Sockets: 2

       [ Id: summary-cpu ]

  5-8. Intel(R) Omni-Path Host Fabric Interfaces (HFIs): "integrated x16" "separate x16"

       [ Id: summary-lspci_hfi_width_info ]

  9-12. Total Memory: 404.308GB; DIMMs Per Socket:

       [ Id: summary-memory ]

This analysis took 0.196293 seconds.

PASS:  All checks passed.
```

Appendix B
Example Output When Validating as Unprivileged User

```
# Example Output When Validating as Unprivileged User
Intel(R) Cluster Checker 2018 Update 3 (build 20171223)

Intel(R) Scalable System Framework version: 2016.0

Database: clck_default, SQLite file work_test3.db.



25 checks requested: all to all, cpu, dgemm, environment, hpcg cluster,
hpcg cluster data, hpcg single, hpl, hpl info, imb pingpong, imb pingpong info, iozone,
kernel, libraries, lsb tools, memory, mount, mpi local, perl, python, shells,
ssf_version, storage, stream, tcl



Reading the database for the following checks:

   all_to_all...  done (0.00122 seconds)

   cpu...  done (0.00819 seconds)

   dgemm...  done (0.0021 seconds)

   environment...  done (0.0915 seconds)

   hpcg_cluster...  done (0.000499 seconds)

   hpcg_cluster_data...  done (0.000425 seconds)

   hpcg_single...  done (0.000789 seconds)

   hpl...  done (0.000737 seconds)

   hpl_info...  done (0.000707 seconds)

   imb_pingpong...  done (0.00442 seconds)

   imb_pingpong_info...  done (0.00347 seconds)

   iozone...  done (0.00528 seconds)

   kernel...  done (0.00103 seconds)

   libraries...  done (0.0416 seconds)

   lsb_tools...  done (0.00153 seconds)

   memory...  done (0.00648 seconds)

   mount...  done (0.00156 seconds)

   mpi_local...  done (0.00134 seconds)

   perl...  done (0.000758 seconds)

   python...  done (0.000377 seconds)

   shells...  done (0.000423 seconds)
```

```
   ssf_version...  done (0.000429 seconds)
   storage...  done (0.00324 seconds)
   stream...  done (0.00476 seconds)
   tcl...  done (0.000473 seconds)
Nodes being tested:
cn[01-04].cluster
0 diagnoses
0 diagnosed signs
0 undiagnosed signs
28 informational signs:


  1-4. CPU Model Name: "Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz"; Sockets: 2
       [ Id: summary-cpu ]
  5. DGEMM Performance: 2152.270 GFLOP/s
       [ Id: summary-dgemm ]
  6. DGEMM Performance: 2347.490 GFLOP/s
       [ Id: summary-dgemm ]
  7. DGEMM Performance: 2300.420 GFLOP/s
       [ Id: summary-dgemm ]
  8. DGEMM Performance: 2310.570 GFLOP/s
       [ Id: summary-dgemm ]
  9. HPCG 4-Node Performance: 149.42 GFLOP/s
       [ Id: summary-hpcg_cluster ]
  10. HPCG 1-Node Performance: 38.18 GFLOP/s
       [ Id: summary-hpcg_single ]
  11. HPCG 1-Node Performance: 37.65 GFLOP/s
       [ Id: summary-hpcg_single ]
  12. HPCG 1-Node Performance: 37.85 GFLOP/s
       [ Id: summary-hpcg_single ]
  13. HPCG 1-Node Performance: 37.90 GFLOP/s
       [ Id: summary-hpcg_single ]
  14. HPL 4-Node Performance: 8293.34 GFLOP/s
       [ Id: summary-hpl_cluster ]
  15. IMB pingpong bandwidth: 11714.50 MB/s and latency: 1.29 us
```

```
      [ Id: summary-imb_pingpong ]
  16. IMB pingpong bandwidth: 11789.80 MB/s and latency: 1.30 us
      [ Id: summary-imb_pingpong ]
  17. IMB pingpong bandwidth: 11763.40 MB/s and latency: 1.27 us
      [ Id: summary-imb_pingpong ]
  18. IMB pingpong bandwidth: 11781.90 MB/s and latency: 1.26 us
      [ Id: summary-imb_pingpong ]
  19. IMB pingpong bandwidth: 11833.00 MB/s and latency: 1.28 us
      [ Id: summary-imb_pingpong ]
  20. IMB pingpong bandwidth: 11704.70 MB/s and latency: 1.26 us
      [ Id: summary-imb_pingpong ]
  21-24. Total Memory: 404.308GB; DIMMs Per Socket:
      [ Id: summary-memory ]
  25. STREAM Performance: 191711.00 MB/s
      [ Id: summary-stream ]
  26. STREAM Performance: 192002.00 MB/s
      [ Id: summary-stream ]
  27. STREAM Performance: 193108.00 MB/s
      [ Id: summary-stream ]
  28. STREAM Performance: 191997.00 MB/s
      [ Id: summary-stream ]


This analysis took 0.520334 seconds.
PASS: All checks passed.
```

About QCT

QCT (Quanta Cloud Technology) is a global data center solution provider extending the power of hyperscale data center design in standard and open SKUs to all datacenter customers.

Product lines include servers, storage, network switches, integrated rack systems and cloud solutions, all delivering hyperscale efficiency, scalability, reliability, manageability, serviceability and optimized performance for each workload.

QCT offers a full spectrum of data center products and services from engineering, integration and optimization to global supply chain support, all under one roof.

The parent of QCT is Quanta Computer Inc., a Fortune Global 500 technology engineering and manufacturing company.

http://www.QCT.io

United States

QCT LLC., Silicon Valley office
1010 Rincon Circle, San Jose, CA 95131
TOLL-FREE: 1-855-QCT-MUST
TEL: +1-510-270-6111
FAX: +1-510-270-6161
Support: +1-510-270-6216

QCT LLC., Seattle office
13810 SE Eastgate Way, Suite 190, Building 1,
Bellevue, WA 98005
TEL: +1-425-633-1620
FAX: +1-425-633-1621

China

云达科技, 北京办公室（Quanta Cloud Technology）
北京市朝阳区东大桥路 12 号润诚中心 2 号楼
TEL: +86-10-5920-7600
FAX: +86-10-5981-7958

云达科技, 杭州办公室（Quanta Cloud Technology）
浙江省杭州市西湖区古墩路浙商财富中心 4 号楼 303 室
Room 303 · Building No.4 · ZheShang Wealth Center
No. 83 GuDun Road, Xihu District, Hangzhou, Zhejiang , China
TEL: +86-571-2819-8660

Japan

Quanta Cloud Technology Japan 株式会社
日本国東京都港区芝大門二丁目五番八号
牧田ビル 3 階
Makita Building 3F, 2-5-8, Shibadaimon ,
Minato-ku, Tokyo 105-0012, Japan
TEL: +81-3-5777-0818
FAX: +81-3-5777-0819

Taiwan

雲達科技（Quanta Cloud Technology）
桃園市龜山區文化二路 211 號 1 樓
1F, No. 211 Wenhua 2nd Rd., Guishan Dist.,
Taoyuan City 33377, Taiwan
TEL: +886-3-286-0707
FAX: +886-3-327-0001

Germany

Quanta Cloud Technology Germany GmbH
Hamborner Str. 55, 40472 Düsseldorf ,
Germany
TEL: + 492405-4083-1300

Other regions

Quanta Cloud Technology
No. 211 Wenhua 2nd Rd., Guishan Dist.,
Taoyuan City 33377, Taiwan
TEL: +886-3-327-2345
FAX: +886-3-397-4770