OpenHPC (v1.2)
Cluster Building Recipes

CentOS7.2 Base OS
Warewulf/SLURM Edition for Linux* (x86_64)

Document Last Update: 2016-11-13
Document Revision: 4f4aa01
Legal Notice

Copyright © 2016, OpenHPC, a Linux Foundation Collaborative Project. All rights reserved.

This documentation is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit [http://creativecommons.org/licenses/by/4.0](http://creativecommons.org/licenses/by/4.0)

Intel, the Intel logo, and other Intel marks 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.
Install Guide (v1.2): CentOS7.2/x86_64 + Warewulf + SLURM

Contents
1 Introduction ........................................... 5
  1.1 Target Audience .................................. 5
  1.2 Requirements/Assumptions ......................... 5
  1.3 Inputs ........................................... 6
2 Install Base Operating System (BOS) .................. 7
3 Install OpenHPC Components .......................... 7
  3.1 Enable OpenHPC repository for local use .............. 7
  3.2 Installation template ............................... 8
  3.3 Add provisioning services on master node .......... 8
  3.4 Add resource management services on master node .. 9
  3.5 Add InfiniBand support services on master node ... 9
  3.6 Complete basic Warewulf setup for master node ... 10
  3.7 Define compute image for provisioning ............. 11
    3.7.1 Build initial BOS image ....................... 11
    3.7.2 Add OpenHPC components ..................... 11
    3.7.3 Customize system configuration ............... 12
    3.7.4 Additional Customization (optional) .......... 13
      3.7.4.1 Increase locked memory limits ........... 13
      3.7.4.2 Enable ssh control via resource manager ... 13
      3.7.4.3 Add Lustre client ........................ 13
      3.7.4.4 Add Nagios monitoring .................... 14
      3.7.4.5 Add Ganglia monitoring .................... 15
      3.7.4.6 Add ClusterShell ......................... 16
      3.7.4.7 Add mrsh .................................. 16
      3.7.4.8 Add genders ................................ 16
      3.7.4.9 Add ConMan ................................ 16
      3.7.4.10 Enable forwarding of system logs ......... 17
    3.7.5 Import files .................................. 17
  3.8 Finalizing provisioning configuration .............. 18
    3.8.1 Assemble bootstrap image ...................... 18
    3.8.2 Assemble Virtual Node File System (VNFS) image .... 18
    3.8.3 Register nodes for provisioning ............... 18
    3.8.4 Optional kernel arguments ..................... 19
    3.8.5 Optionally configure stateful provisioning .... 19
  3.9 Boot compute nodes ................................ 20
4 Install OpenHPC Development Components ............. 21
  4.1 Development Tools ................................ 21
  4.2 Compilers ........................................ 21
  4.3 MPI Stacks ....................................... 21
  4.4 Performance Tools ................................ 22
  4.5 Setup default development environment ............ 22
  4.6 3rd Party Libraries and Tools ..................... 22
  4.7 Optional Development Tool Builds .................. 23
5 Resource Manager Startup ............................ 24
6 Run a Test Job
  6.1 Interactive execution .......................................................... 26
  6.2 Batch execution .................................................................. 27

Appendices
  A Installation Template ............................................................ 28
  B Rebuilding Packages from Source ............................................ 29
  C Package Manifest ................................................................. 30
  D Package Signatures ............................................................... 41
1 Introduction

This guide presents a simple cluster installation procedure using components from the OpenHPC software stack. 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 using a mix of open-source components. The documentation herein is intended to be reasonably generic, but uses the underlying motivation of a small, 4-node stateless cluster installation to define a step-by-step process. Several optional customizations are included and the intent is that these collective instructions can be modified as needed for local site customizations.

**Base Linux Edition:** this edition of the guide highlights installation without the use of a companion configuration management system and directly uses distro-provided package management tools for component selection. The steps that follow also highlight specific changes to system configuration files that are required as part of the cluster install process.

1.1 Target Audience

This guide is targeted at experienced Linux system administrators for HPC environments. Knowledge of software package management, system networking, and PXE booting is assumed. Command-line input examples are highlighted throughout this guide via the following syntax:

```
[sms]# echo "OpenHPC hello world"
```

Unless specified otherwise, the examples presented are executed with elevated (root) privileges. The examples also presume use of the BASH login shell, though the equivalent commands in other shells can be substituted. In addition to specific command-line instructions called out in this guide, an alternate convention is used to highlight potentially useful tips or optional configuration options. These tips are highlighted via the following format:

**Tip**

The solution is to increase the size of the manuals. –Mark V. Shaney

1.2 Requirements/Assumptions

This installation recipe assumes the availability of a single head node *master*, and four *compute* nodes. The *master* node serves as the overall system management server (SMS) and is provisioned with CentOS7.2 and is subsequently configured to provision the remaining *compute* nodes with Warewulf in a stateless configuration. The terms *master* and SMS are used interchangeably in this guide. For power management, we assume that the compute node baseboard management controllers (BMCs) are available via IPMI from the chosen master host. For file systems, we assume that the chosen master server will host an NFS file system that is made available to the compute nodes. Installation information is also discussed to optionally include a Lustre file system mount and in this case, the Lustre file system is assumed to exist previously.

An outline of the physical architecture discussed is shown in Figure 1 and highlights the high-level networking configuration. The *master* host requires at least two Ethernet interfaces with *eth0* connected to the local data center network and *eth1* used to provision and manage the cluster backend (note that these
interface names are examples and may be different depending on local settings and OS conventions). Two logical IP interfaces are expected to each compute node: the first is the standard Ethernet interface that will be used for provisioning and resource management. The second is used to connect to each host’s BMC and is used for power management and remote console access. Physical connectivity for these two logical IP networks is often accommodated via separate cabling and switching infrastructure; however, an alternate configuration can also be accommodated via the use of a shared NIC, which runs a packet filter to divert management packets between the host and BMC.

In addition to the IP networking, there is a high-speed network (InfiniBand in this recipe) that is also connected to each of the hosts. This high speed network is used for application message passing and optionally for Lustre connectivity as well.

1.3 Inputs

As this recipe details installing a cluster starting from bare-metal, there is a requirement to define IP addresses and gather hardware MAC addresses in order to support a controlled provisioning process. These values are necessarily unique to the hardware being used, and this document uses variable substitution ($\{variable\}$) in the command-line examples that follow to highlight where local site inputs are required. A summary of the required and optional variables used throughout this recipe are presented below. Note that while the example definitions above correspond to a small 4-node compute subsystem, the compute parameters are defined in array format to accommodate logical extension to larger node counts.
2 Install Base Operating System (BOS)

In an external setting, installing the desired BOS on a master SMS host typically involves booting from a DVD ISO image on a new server. With this approach, insert the CentOS7.2 DVD, power cycle the host, and follow the distro provided directions to install the BOS on your chosen master host. Alternatively, if choosing to use a pre-installed server, please verify that it is provisioned with the required CentOS7.2 distribution.

While it is theoretically possible to provision a Warewulf cluster with SELinux enabled, doing so is beyond the scope of this document. Even the use of permissive mode can be problematic and we therefore recommend disabling SELinux on the master SMS host. If SELinux components are installed locally, the selinuxenabled command can be used to determine if SELinux is currently enabled. If enabled, consult the distro documentation for information on how to disable.

3 Install OpenHPC Components

With the BOS installed and booted, the next step is to add desired OpenHPC packages onto the master server in order to provide provisioning and resource management services for the rest of the cluster. The following subsections highlight this process.

3.1 Enable OpenHPC repository for local use

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.

```bash
[sms]# yum install http://build.openhpc.community/OpenHPC:/1.2/CentOS_7.2/x86_64/ohpc-release-1.2-1.x86_64.rpm
```

In addition to the OpenHPC package repository, the *master* host also requires access to the standard base OS distro repositories in order to resolve necessary dependencies. For CentOS7.2, the requirements are to have access to both the base OS and EPEL repositories for which mirrors are freely available online:

- **CentOS-7 - Base 7.2.1511** (e.g. [http://mirror.centos.org/centos-7/7.2.1511/os/x86_64](http://mirror.centos.org/centos-7/7.2.1511/os/x86_64))
- **EPEL 7** (e.g. [http://download.fedoraproject.org/pub/epel/7/x86_64](http://download.fedoraproject.org/pub/epel/7/x86_64))

The public EPEL repository will be enabled automatically upon installation of the `ohpc-release` package. Note that this requires the CentOS Extras repository, which is shipped with CentOS and is enabled by default.

### 3.2 Installation template

The collection of command-line instructions that follow in this guide, when combined with local site inputs, can be used to implement a bare-metal system installation and configuration. The format of these commands is intended to be usable via direct cut and paste (with variable substitution for site-specific settings). Alternatively, the OpenHPC documentation package (`docs-ohpc`) includes a template script which includes a summary of all of the commands used herein. This script can be used in conjunction with a simple text file to define the local site variables defined in the previous section (§1.3) and is provided as a convenience for administrators. For additional information on accessing this script, please see Appendix A.

### 3.3 Add provisioning services on *master* node

With the OpenHPC repository enabled, we can now begin adding desired components onto the *master* server. This repository provides a number of aliases that group logical components together in order to help aid in this process. For reference, a complete list of available group aliases and RPM packages available via OpenHPC are provided in Appendix C. To add support for provisioning services, the following commands illustrate addition of a common base package followed by the Warewulf provisioning system.

```bash
[sms]# yum -y groupinstall ohpc-base
[sms]# yum -y groupinstall ohpc-warewulf
```

Provisioning services with Warewulf rely on DHCP, TFTP, and HTTP network protocols. Depending on the local Base OS configuration on the SMS host, default firewall rules may prohibit these services. Consequently, this recipe assumes that the local firewall running on the SMS host is disabled. If installed, the default firewall service can be disabled as follows:

```bash
[sms]# systemctl disable firewalld
[sms]# systemctl stop firewalld
```
Tip

Many server BIOS configurations have PXE network booting configured as the primary option in the boot order by default. If your compute nodes have a different device as the first in the sequence, the ipmitool utility can be used to enable PXE.

```
[sms]# ipmitool -E -I lanplus -H ${bmc_ipaddr} -U root chassis bootdev pxe options=persistent
```

HPC systems typically rely on synchronized clocks throughout the system and the NTP protocol can be used to facilitate this synchronization. To enable NTP services on the SMS host with a specific server `${ntp_server}`, issue the following:

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

3.4 Add resource management services on master node

OpenHPC provides multiple options for distributed resource management. The following command adds the Slurm workload manager server components to the chosen master host. Note that client-side components will be added to the corresponding compute image in a subsequent step.

```
[sms]# yum -y groupinstall ohpc-slurm-server
```

Slurm requires the designation of a system user that runs the underlying resource management daemons. The default configuration file that is supplied with the OpenHPC build of Slurm identifies this SlurmUser to be a dedicated user named “slurm” and this user must exist. The following command can be used to add this user to the master server:

```
[sms]# useradd slurm
```

Other versions of this guide are available that describe installation of other resource management systems, and they can be found in the docs-ohpc package.

3.5 Add InfiniBand support services on master node

The following command adds OFED and PSM support using base distro-provided drivers to the chosen master host.

```
[sms]# yum -y groupinstall "InfiniBand Support"
[sms]# yum -y install infinipath-psm
```

Load IB drivers
```
[sms]# systemctl start rdma
```

With the InfiniBand drivers included, you can also enable (optional) IPoIB functionality which provides a mechanism to send IP packets over the IB network. If you plan to mount a Lustre file system over InfiniBand (see § 3.7.4.3 for additional details), then having IPoIB enabled is a requirement for the Lustre client. OpenHPC provides a template configuration file to aid in setting up an ib0 interface on the master host. To use, copy the template provided and update the `${sms_ipoib}` and `${ipoib_netmask}` entries to
match local desired settings (alter ib0 naming as appropriate if system contains dual-ported or multiple HCAs).

```bash
[sms]# cp /opt/ohpc/pub/examples/network/centos/ifcfg-ib0 /etc/sysconfig/network-scripts
# Define local IPoIB address and netmask
[sms]# perl -pi -e "s/master_ipoib/$(sms_ip)/" /etc/sysconfig/network-scripts/ifcfg-ib0
[sms]# perl -pi -e "s/ipoib_netmask/$(ipoib_netmask)/" /etc/sysconfig/network-scripts/ifcfg-ib0
# Initiate ib0
[sms]# ifup ib0
```

Tip

InfiniBand networks require a subnet management service that can typically be run on either an administrative node, or on the switch itself. The optimal placement and configuration of the subnet manager is beyond the scope of this document, but CentOS7.2 provides the opensm package should you choose to run it on the master node.

### 3.6 Complete basic Warewulf setup for master node

At this point, all of the packages necessary to use Warewulf on the master host should be installed. Next, we need to update several configuration files in order to allow Warewulf to work with CentOS7.2 and to support local provisioning using a second private interface (refer to Figure I).

Tip

By default, Warewulf is configured to provision over the eth1 interface and the steps below include updating this setting to override with a potentially alternatively-named interface specified by `${sms_eth_internal}`.

```bash
# Configure Warewulf provisioning to use desired internal interface
[sms]# perl -pi -e "s/device = eth1/device = $(sms_eth_internal)/" /etc/warewulf/provision.conf
# Enable tftp service for compute node image distribution
[sms]# perl -pi -e "s/\"s+disable\s+= yes\s+\s+disable = no/" /etc/xinetd.d/tftp
# Enable http access for Warewulf cgi-bin directory to support newer apache syntax
[sms]# export MODFILE=/etc/httpd/conf.d/warewulf-httpd.conf
[sms]# perl -pi -e "s/cgi-bin>/\$MODFILE/cgi-bin>\n\nRequire all granted/" $MODFILE
[sms]# perl -pi -e "s/\n\nRequire all granted/" $MODFILE
[sms]# perl -ni -e "print unless \n\nallow,deny/" $MODFILE
# Enable internal interface for provisioning
[sms]# ifconfig $(sms_ip) netmask $(internal_netmask) up
# Restart/enable relevant services to support provisioning
[sms]# systemctl restart xinetd
[sms]# systemctl enable mariadb.service
[sms]# systemctl restart mariadb
[sms]# systemctl enable httpd.service
[sms]# systemctl restart httpd
```
3.7 Define compute image for provisioning

With the provisioning services enabled, the next step is to define and customize a system image that can subsequently be used to provision one or more compute nodes. The following subsections highlight this process.

3.7.1 Build initial BOS image

The OpenHPC build of Warewulf includes specific enhancements enabling support for CentOS7.2. The following steps illustrate the process to build a minimal, default image for use with Warewulf. We begin by defining a directory structure on the master host that will represent the root filesystem of the compute node. The default location for this example is in `/opt/ohpc/admin/images/centos7.2`.

**Tip**

Note that Warewulf is configured by default to access an external repository (vault.centos.org) during the `wwmkchroot` process. If the master host cannot reach the public CentOS mirrors, or if you prefer to access a locally cached mirror, set the `${BOS_MIRROR}` environment variable to your desired repo location and update the template file prior to running the `wwmkchroot` command below. For example:

```bash
# Override default OS repository (optional) - set BOS_MIRROR variable to desired repo location
[sms]# perl -pi -e "s#^YUM_MIRROR=(\S+)#YUM_MIRROR=${BOS_MIRROR}#" /usr/libexec/warewulf/wwmkchroot/centos-7.tmpl
```

Now, we can include additional components to the compute instance using `yum` to install into the chroot location defined previously:

```bash
[sms]# cp -p /etc/resolv.conf $CHROOT/etc/resolv.conf
```

3.7.2 Add OpenHPC components

The `wwmkchroot` process used in the previous step is designed to provide a minimal CentOS7.2 configuration. Next, we add additional components to include resource management client services, InfiniBand drivers, and other additional packages to support the default OpenHPC environment. This process augments the chroot-based install performed by `wwmkchroot` to augment the base provisioning image and will access the BOS and OpenHPC repositories to resolve package install requests. To access the remote repositories by hostname (and not IP addresses), the chroot environment needs to be updated to enable DNS resolution. Assuming that the master host has a working DNS configuration in place, the chroot environment can be updated with a copy of the configuration as follows:

```bash
[sms]# cp -p /etc/resolv.conf $CHROOT/etc/resolv.conf
```
# Add Slurm client support
[sms]# yum -y --installroot=$CHROOT groupinstall ohpc-slurm-client

# Add IB support and enable
[sms]# yum -y --installroot=$CHROOT groupinstall "InfiniBand Support"
[sms]# yum -y --installroot=$CHROOT install infinipath-psm
[sms]# chroot $CHROOT systemctl enable rdma

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

# Add kernel drivers
[sms]# yum -y --installroot=$CHROOT install kernel

# Include modules user environment
[sms]# yum -y --installroot=$CHROOT install lmod-ohpc

3.7.3 Customize system configuration

Prior to assembling the image, it is advantageous to perform any additional customization within the chroot environment created for the desired compute instance. The following steps document the process to add a local ssh key created by Warewulf to support remote access, identify the resource manager server, configure NTP for compute resources, and enable NFS mounting of a $HOME file system and the public OpenHPC install path (/opt/ohpc/pub) that will be hosted by the master host in this example configuration.

# add new cluster key to base image
[sms]# wwinit ssh_keys
[sms]# cat ~/.ssh/cluster.pub >> $CHROOT/root/.ssh/authorized_keys

# add NFS client mounts of /home and /opt/ohpc/pub to base image
[sms]# echo "$sms_ip:/home /home nfs nfsvers=3,rsize=1024,wsize=1024,cto 0 0" >> $CHROOT/etc/fstab
[sms]# echo "$sms_ip:/opt/ohpc/pub /opt/ohpc/pub nfs nfsvers=3 0 0" >> $CHROOT/etc/fstab

# Identify resource manager host name on master host
[sms]# perl -pi -e "s/ControlMachine=\S+/ControlMachine=$sms_name/" /etc/slurm/slurm.conf

# Export /home and OpenHPC public packages from master server to cluster compute nodes
[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
[sms]# systemctl enable nfs-server

# Enable NTP time service on computes and identify master host as local NTP server
[sms]# chroot $CHROOT systemctl enable ntpd
[sms]# echo "server $sms_ip" >> $CHROOT/etc/ntp.conf

Tip

SLURM requires enumeration of the physical hardware characteristics for compute nodes under its control. In particular, three configuration parameters combine to define consumable compute resources: Sockets, CoresPerSocket, and ThreadsPerCore. The default configuration file provided via OpenHPC assumes dual-socket, 8 cores per socket, and two threads per core for this 4-node example. If this does not reflect your local hardware, please update the configuration file at /etc/slurm/slurm.conf accordingly to match your particular hardware.
3.7.4 Additional Customization (optional)

This section highlights common additional customizations that can optionally be applied to the local cluster environment. These customizations include:

- Increase memlock limits
- Restrict ssh access to compute resources
- Add Cluster Checker
- Add Lustre client
- Add Nagios Core monitoring
- Add Ganglia monitoring
- Add ClusterShell
- Add mrsh
- Add genders
- Add ConMan

Details on the steps required for each of these customizations are discussed further in the following sections.

3.7.4.1 Increase locked memory limits

In order to utilize InfiniBand as the underlying high speed interconnect, it is generally necessary to increase the locked memory settings for system users. This can be accomplished by updating the `/etc/security/limits.conf` file and this should be performed within the compute image and on all job submission hosts. In this recipe, jobs are submitted from the master host, and the following commands can be used to update the maximum locked memory settings on both the master host and the compute image:

```bash
# Update memlock settings on master
[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

# Update memlock settings within compute 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
```

3.7.4.2 Enable ssh control via resource manager

An additional optional customization that is recommended is to restrict ssh access on compute nodes to only allow access by users who have an active job associated with the node. This can be enabled via the use of a pluggable authentication module (PAM) provided as part of the Slurm package installs. To enable this feature within the compute image, issue the following:

```
[sms]# echo "account required pam_slurm.so" >> $CHROOT/etc/pam.d/sshd
```

3.7.4.3 Add Lustre client

To add Lustre client support on the cluster, it necessary to install the client and associated modules on each host needing to access a Lustre file system. In this recipe, it is assumed that the Lustre file system is hosted by servers that are pre-existing and are not part of the install process. Outlining the variety of Lustre client mounting options is beyond the scope of this document, but the general requirement is to add a mount entry for the desired file system that defines the management server (MGS) and underlying network transport protocol. To add client mounts on both the master server and compute image, the following commands can be used. Note that the Lustre file system to be mounted is identified by the `$mgs_fs_name` variable. In this example, the file system is configured to be mounted locally as `/mnt/lustre`.

```
# Add Lustre client software to master host
[sms]# yum -y install lustre-client-ohpc lustre-client-ohpc-modules
```
The default underlying network type used by Lustre is *tcp*. If your external Lustre file system is to be mounted using a network type other than *tcp*, additional configuration files are necessary to identify the desired network type. The example below illustrates creation of modprobe configuration files instructing Lustre to use an InfiniBand network with the *o2ib* LNET driver attached to *ib0*. Note that these modifications are made to both the *master* host and *compute* image.

With the Lustre configuration complete, the client can be mounted on the *master* host as follows:

```bash
[sms]# mkdir /mnt/lustre
[sms]# mount -t lustre -o localflock ${mgs_fs_name} /mnt/lustre
```

### 3.7.4.4 Add Nagios monitoring

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 *master* node, and add the facility to run tests and gather metrics from provisioned *compute* nodes.

```bash
# Install Nagios base, Remote Plugin Engine, and Plugins on master host
[sms]# yum -y groupinstall ohpc-nagios

# Also install in compute node image
[sms]# yum -y --installroot=$CHROOT groupinstall ohpc-nagios

# 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 : $sms_ip : ALLOW" >> $CHROOT/etc/nagios/nrpe.cfg
[sms]# chroot $CHROOT /usr/sbin/useradd -c "NRPE user for the NRPE service" -d /var/run/nrpe -r -g nrpe -s /sbin/nologin nrpe
[sms]# chroot $CHROOT /usr/sbin/groupadd -r 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]# for ((i=0; i<num_computes; i++)); do
  perl -pi -e "s/HOSTNAME=$((i+1))/${c_name[$i]}/g; s/HOST=$((i+1))/IP=${c_ip[$i]}/g" /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

# Enable Nagios on master, and configure
[sms]# chkconfig nagios on
[sms]# systemctl start nagios
[sms]# chmod u+s /grave.ts1

# Install Ganglia
[sms]# yum -y groupinstall ohpc-ganglia
[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

Once enabled and running, Ganglia should provide access to a web-based monitoring console on the master host. Read access to monitoring metrics will be enabled by default and can be accessed via a web browser. When running a web browser directly on the master host, the Ganglia top-level overview is available at http://localhost/ganglia. When accessing remotely, replace localhost with the chosen name of your master host ($\{sms_name\}).
3.7.4.6 Add ClusterShell  
ClusterShell is an event-based Python library to execute commands in parallel across cluster nodes. Installation and basic configuration defining three node groups (adm, compute, and all) is as follows:

```bash
# 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: c[1-$(num_computes)]" >> local.cfg
[sms]$ echo "all: @adm,@compute" >> local.cfg
```

3.7.4.7 Add mrsh  
mrsh is a secure remote shell utility, like ssh, which uses munge for authentication and encryption. By using the munge installation used by Slurm, mrsh provides shell access to systems using the same munge key without having to track ssh keys. Like ssh, mrsh provides a remote copy command, mrcp, and can be used as a rcmd by pdsh. Example installation and configuration is as follows:

```bash
# Install mrsh
[sms]$ yum -y install mrsh-ohpc mrsh-rsh-compat-ohpc
[sms]$ yum -y --installroot=$CHROOT install mrsh-ohpc mrsh-rsh-compat-ohpc mrsh-server-ohpc

# Identify mshell and mlogin in services file
[sms]$ echo "mshell 21212/tcp # mrshd" >> /etc/services
[sms]$ echo "mlogin 541/tcp # mrlogind" >> /etc/services

# Enable xinetd in compute node image
[sms]$ chroot $CHROOT systemctl enable xinetd
```

3.7.4.8 Add genders  
genders is a static cluster configuration database or node typing database used for cluster configuration management. Other tools and users can access the genders database in order to make decisions about where an action, or even what action, is appropriate based on associated types or "genders". Values may also be assigned to and retrieved from a gender to provide further granularity. The following example highlights installation and configuration of two genders: compute and bmc.

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

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

3.7.4.9 Add ConMan  
ConMan is a serial console management program designed to support a large number of console devices and simultaneous users. It supports logging console device output and connecting to compute node consoles via IPMI serial-over-lan. Installation and example configuration is outlined below.

```bash
# 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)
```
# 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
done

# Enable and start conman
wwsh systemctl enable conman
wwsh systemctl start conman

### Enable forwarding of system logs

It is often desirable to consolidate system logging information for the cluster in a central location, both to provide easy access to the data, and to reduce the impact of storing data inside the stateless compute node’s memory footprint. The following commands highlight the steps necessary to configure compute nodes to forward their logs to the SMS, and to allow the SMS to accept these log requests.

#### # Configure SMS to receive messages and reload rsyslog configuration

```bash
wwsh perl -pi -e "s/\#\ ModLoad imudp/\ ModLoad imudp/" /etc/rsyslog.conf
wwsh perl -pi -e "s/\#UDPServerRun 514/\ UDPServerRun 514/" /etc/rsyslog.conf
```

#### # Define compute node forwarding destination

```bash
wwsh echo "*.* $sms_ip:514" >> CHROOT/etc/rsyslog.conf
```

#### # Disable most local logging on computes. Emergency and boot logs will remain on the compute nodes

```bash
wwsh perl -pi -e "s/^*\.info/\#*\.info/" CHROOT/etc/rsyslog.conf
wwsh perl -pi -e "s/^authpriv/\#authpriv/" CHROOT/etc/rsyslog.conf
wwsh perl -pi -e "s/^mail/\#mail/" CHROOT/etc/rsyslog.conf
wwsh perl -pi -e "s/^cron/\#cron/" CHROOT/etc/rsyslog.conf
wwsh perl -pi -e "s/^uucp/\#uucp/" CHROOT/etc/rsyslog.conf
```

### Import files

The Warewulf system includes functionality to import arbitrary files from the provisioning server for distribution to managed hosts. This is one way to distribute user credentials to *compute* hosts. To import local file-based credentials, issue the following:

```bash
wwsh file import /etc/passwd
wwsh file import /etc/group
wwsh file import /etc/shadow
```

Similarly, to import the global Slurm configuration file and the cryptographic key that is required by the *munge* authentication library to be available on every host in the resource management pool, issue the following:

```bash
wwsh file import /etc/slurm/slurm.conf
wwsh file import /etc/munge/munge.key
```

Finally, to add *optional* support for controlling IPoIB interfaces (see §3.5), OpenHPC includes a template file for Warewulf that can optionally be imported and used later to provision *ib0* network settings. Finally, to add *optional* support for controlling IPoIB interfaces (see §3.5), OpenHPC includes a template file for Warewulf that can optionally be imported and used later to provision *ib0* network settings.
Install Guide (v1.2): CentOS7.2/x86_64 + Warewulf + SLURM

3.8 Finalizing provisioning configuration

Warewulf employs a two-stage boot process for provisioning nodes via creation of a bootstrap image that is used to initialize the process, and a virtual node file system capsule containing the full system image. This section highlights creation of the necessary provisioning images, followed by the registration of desired compute nodes.

3.8.1 Assemble bootstrap image

The bootstrap image includes the runtime kernel and associated modules, as well as some simple scripts to complete the provisioning process. The following commands highlight the inclusion of additional drivers and creation of the bootstrap image based on the running kernel.

```bash
# (Optional) Include Lustre drivers; needed if enabling Lustre client on computes
[smx]$ export WW_CONF=/etc/warewulf/bootstrap.conf
[smx]$ echo "drivers += updates/kernel/" >> $WW_CONF

# Build bootstrap image
[smx]$ wwbootstrap
```

3.8.2 Assemble Virtual Node File System (VNFS) image

With the local site customizations in place, the following step uses the `wwvnfs` command to assemble a VNFS capsule from the chroot environment defined for the `compute` instance.

```bash
[smx]$ wwvnfs -y --chroot $CHROOT
```

3.8.3 Register nodes for provisioning

In preparation for provisioning, we can now define the desired network settings for four example compute nodes with the underlying provisioning system and restart the `dhcp` service. Note the use of variable names for the desired compute hostnames, node IPs, and MAC addresses which should be modified to accommodate local settings and hardware. Included in these steps are commands to enable Warewulf to manage IPoIB settings and corresponding definitions of IPoIB addresses for the compute nodes. This is typically optional unless you are planning to include a Lustre client mount over InfiniBand. The final step in this process associates the VNFS image assembled in previous steps with the newly defined compute nodes, utilizing the user credential files and munge key that were imported in §3.7.5.

```bash
# Set provisioning interface as the default networking device
[smx]$ echo "GATEWAYDEV=$eth_provision" > /tmp/network.$$;
[smx]$ wsh -y file import /tmp/network.$$ --name network
[smx]$ wsh -y file set network --path /etc/sysconfig/network --mode=0644 --uid=0

# Add nodes to Warewulf data store
[smx]$ for ((i=0; i<$num_computes; i++)); do
  wsh -y node new ${c_name[i]} --ipaddr=${c_ip[i]} --hwaddr=${c_mac[i]} -D $eth_provision
done
```
# Define provisioning image for hosts
[smsh]# wwsh -y provision set "$(compute_regex)" --vnfs=centos7.2 --bootstrap="uname -r \ 
--files=dynamic_hosts,passwd,group,shadow,slurm.conf,munge.key,network"

# Optionally define IPoIB network settings (required if planning to mount Lustre over IB)
[smsh]# for ((i=0; i<num_computes; i++)); do
        wwsh -y node set ${c_name[$i]} -D ib0 --ipaddr=${c_ip[$i]} --netmask=${ipoib_netmask}
done
[smsh]# wwsh -y provision set "$(compute_regex)" --fileadd=ifcfg-ib0.ww

Tip

Warewulf includes a utility named `wwnodescan` to automatically register new compute nodes versus the outlined node-addition approach which requires hardware MAC addresses to be gathered in advance. With `wwnodescan`, nodes will be added to the Warewulf database in the order in which their DHCP requests are received by the master, so care must be taken to boot nodes in the order one wishes to see preserved in the Warewulf database. The IP address provided will be incremented after each node is found, and the utility will exit after all specified nodes have been found. Example usage is highlighted below:

[smsh]# wwnodescan --netdev=${eth_provision} --ipaddr=${c_ip[0]} --netmask=${internal_netmask} \ 
--vnfs=centos7.2 --bootstrap='uname -r' ${c_name[0]}-..${c_name[3]}

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

3.8.4 Optional kernel arguments

If you chose to enable ConMan in §3.7.4.9, additional boot-time kernel arguments are needed to enable serial console redirection. An example provisioning setting which adds to any other kernel arguments defined in `${kargs}` is as follows:

# Define node kernel arguments to support SOL console
[smsh]# wwsh -y provision set "$(compute_regex)" --kargs ${kargs} console=ttyS1,115200"

3.8.5 Optionally configure stateful provisioning

Warewulf normally defaults to running the assembled VNFS image out of system memory in a stateless configuration. Alternatively, Warewulf can also be used to partition and format persistent storage such that the VNFS image can be installed locally to disk in a stateful manner. This does, however, require that a boot loader (GRUB) be added to the image as follows:

# Add GRUB2 bootloader and re-assemble VNFS image
[smsh]# yum -y --installroot=$CHROOT install grub2
[smsh]# wvvnfs -y --chroot $CHROOT
Enabling stateful nodes also requires additional site-specific, disk-related parameters in the Warewulf configuration. In the example that follows, the compute node configuration is updated to define where to install the GRUB bootloader, which disk to partition, which partition to format, and what the filesystem layout will look like.

```bash
# Update node object parameters
[sms]# export sda1="mountpoint=/boot:dev=sda1:type=ext3:size=500"
[sms]# export sda2="dev=sda2:type=swap:size=32768"
[sms]# export sda3="mountpoint=/dev/sda3:type=ext3:size=f111"
[sms]# wwsh -y object modify -s bootloader=sda -t node "${compute_regex}"
[sms]# wwsh -y object modify -s diskpartition=sda -t node "${compute_regex}"
[sms]# wwsh -y object modify -s diskformat=sda1,sda2,sda3 -t node "${compute_regex}"
[sms]# wwsh -y object modify -s filesystems="sda1, sda2, sda3" -t node "${compute_regex}"
```

Upon subsequent reboot of the modified nodes, Warewulf will partition and format the disk to host the desired VNFS image. Once installed to disk, Warewulf can be instructed to subsequently boot from local storage (alternatively, the BIOS boot option order could be updated to reflect a desire to boot from disk):

```bash
# Update node object parameters
[sms]# wwsh -y object modify -s bootlocal=EXIT -t node "${compute_regex}"
```

Deleting the bootlocal object parameter will cause Warewulf to once again reformat and re-install to local storage upon a new PXE boot request.

### 3.9 Boot compute nodes

At this point, the master server should be able to boot the newly defined compute nodes. Assuming that the compute node BIOS settings are configured to boot over PXE, all that is required to initiate the provisioning process is to power cycle each of the desired hosts using IPMI access. The following commands use the `ipmitool` utility to initiate power resets on each of the four compute hosts. Note that the utility requires that the `IPMI_PASSWORD` environment variable be set with the local BMC password in order to work interactively.

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

Once kicked off, the boot process should take less than 5 minutes (depending on BIOS post times) and you can verify that the compute hosts are available via ssh, or via parallel ssh tools to multiple hosts. For example, to run a command on the newly imaged compute hosts using `pdsh`, execute the following:

```bash
[sms]# pdsh -w c[1-4] uptime
c1 05:03am up 02:02, 0 users, load average: 0.20, 0.13, 0.05
c2 05:03am up 02:02, 0 users, load average: 0.20, 0.14, 0.06
c3 05:03am up 02:02, 0 users, load average: 0.19, 0.15, 0.06
c4 05:03am up 02:02, 0 users, load average: 0.15, 0.12, 0.05
```

**Tip**

While the `pxelinux.0` and `lpxelinux.0` files that ship with Warewulf to enable network boot support a wide range of hardware, some hosts may boot more reliably or faster using the BOS versions provided via the `syslinux-tftpboot` package. If you encounter PXE issues, consider replacing the `pxelinux.0` and `lpxelinux.0` files supplied with `warewulf-provision-ohpc` with versions from `syslinux-tftpboot`. 
4 Install OpenHPC Development Components

The install procedure outlined in §3 highlighted the steps necessary to install a master host, assemble and customize a compute image, and provision several compute hosts from bare-metal. With these steps completed, additional OpenHPC-provided packages can now be added to support a flexible HPC development environment including development tools, C/C++/Fortran compilers, MPI stacks, and a variety of 3rd party libraries. The following subsections highlight the additional software installation procedures.

4.1 Development Tools

To aid in general development efforts, OpenHPC provides recent versions of the GNU autotools collection, the Valgrind memory debugger, EasyBuild, Spack, and R. These can be installed as follows:

```
[sms]$ yum -y groupinstall ohpc-autotools
[sms]$ yum -y install valgrind-ohpc
[sms]$ yum -y install EasyBuild-ohpc
[sms]$ yum -y install spack-ohpc
[sms]$ yum -y install R_base-ohpc
```

4.2 Compilers

OpenHPC presently packages the GNU compiler toolchain integrated with the underlying modules-environment system in a hierarchical fashion. The modules system will conditionally present compiler-dependent software based on the toolchain currently loaded.

```
[sms]$ yum -y install gnu-compilers-ohpc
```

4.3 MPI Stacks

For MPI development support, OpenHPC provides pre-packaged builds for a variety of MPI families and transport layers. Currently available options and their applicability to various network transports are summarized in Table 1. The command that follows installs a default set of MPI families that provide support for ethernet and InfiniBand environments. If installing on a system with an Intel® Omni-Path fabric, use the psm2 variants for OpenMPI and MVAPICH instead (e.g. openmpi-psm2-gnu-ohpc and mvapich2-psm2-gnu-ohpc, respectively).

```
```

Table 1: Available MPI builds

<table>
<thead>
<tr>
<th></th>
<th>Ethernet (TCP)</th>
<th>InfiniBand</th>
<th>Intel® Omni-Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPICH</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MVAPICH2</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MVAPICH2 (psm2)</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>OpenMPI</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>OpenMPI (psm2)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

```
```
4.4 Performance Tools

OpenHPC provides a variety of open-source tools to aid in application performance analysis (refer to Appendix C for a listing of available packages). This group of tools can be installed as follows:

```
[sms]# yum -y groupinstall ohpc-perf-tools-gnu
```

4.5 Setup default development environment

System users often find it convenient to have a default development environment in place so that compilation can be performed directly for parallel programs requiring MPI. This setup can be conveniently enabled via modules and the OpenHPC modules environment is pre-configured to load an ohpc module on login (if present). The following package install provides a default environment that enables autotools, the GNU compiler toolchain, and the MVAPICH2 MPI stack.

```
[sms]# yum -y install lmod-defaults-gnu-mvapich2-ohpc
```

Tip

If you want to change the default environment from the suggestion above, OpenHPC also provides the GNU compiler toolchain with the OpenMPI and MPICH stacks:

- lmod-defaults-gnu-openmpi-ohpc
- lmod-defaults-gnu-mpich-ohpc

4.6 3rd Party Libraries and Tools

OpenHPC provides pre-packaged builds for a number of popular open-source tools and libraries used by HPC applications and developers. For example, OpenHPC provides builds for FFTW and HDF5 (including serial and parallel I/O support), and the GNU Scientific Library (GSL). Again, multiple builds of each package are available in the OpenHPC repository to support multiple compiler and MPI family combinations where appropriate. Note, however, that not all combinatorial permutations may be available for components where there are known license incompatibilities. The general naming convention for builds provided by OpenHPC is to append the compiler and MPI family name that the library was built against directly into the package name. For example, libraries that do not require MPI as part of the build process adopt the following RPM name:

```
package-<compiler_family>-ohpc-<package_version>-<release>.rpm
```

Packages that do require MPI as part of the build expand upon this convention to additionally include the MPI family name as follows:

```
package-<compiler_family>-<mpi_family>-ohpc-<package_version>-<release>.rpm
```

To illustrate this further, the command below queries the locally configured repositories to identify all of the available PETSc packages that were built with the GNU toolchain. The resulting output that is included shows that pre-built versions are available for each of the supported MPI families presented in §4.3.
Tip

OpenHPC-provided 3rd party builds are configured to be installed into a common top-level repository so that they can be easily exported to desired hosts within the cluster. This common top-level path (/opt/ohpc/pub) was previously configured to be mounted on compute nodes in §3.7.3, so the packages will be immediately available for use on the cluster after installation on the master host.

For convenience, OpenHPC provides package aliases for these 3rd party libraries and utilities that can be used to install available libraries for use with the GNU compiler family toolchain. For parallel libraries, aliases are grouped by MPI family toolchain so that administrators can choose a subset should they favor a particular MPI stack. Please refer to Appendix C for a more detailed listing of all available packages in each of these functional areas. To install all available package offerings within OpenHPC, issue the following:

```plaintext
[sms]# yum -y groupinstall ohpc-serial-libs-gnu
[sms]# yum -y groupinstall ohpc-io-libs-gnu
[sms]# yum -y groupinstall ohpc-python-libs-gnu
[sms]# yum -y groupinstall ohpc-runtimes-gnu
```

# Install parallel libs for all available MPI toolchains

```plaintext
[sms]# yum -y groupinstall ohpc-parallel-libs-gnu-mpich
[sms]# yum -y groupinstall ohpc-parallel-libs-gnu-mvapich2
[sms]# yum -y groupinstall ohpc-parallel-libs-gnu-openmpi
```

## 4.7 Optional Development Tool Builds

In addition to the 3rd party development libraries built using the open source toolchains mentioned in §4.6, OpenHPC also provides optional compatible builds for use with the compilers and MPI stack included in newer versions of the Intel® Parallel Studio XE software suite. These packages provide a similar hierarchical user environment experience as other compiler and MPI families present in OpenHPC.

To take advantage of the available builds, the Parallel Studio software suite must be obtained and installed separately. Once installed locally, the OpenHPC compatible packages can be installed using standard package manager semantics. Note that licenses are provided free of charge for many categories of use. In particular, licenses for compilers and developments tools are provided at no cost to academic researchers or developers contributing to open-source software projects. More information on this program can be found at:

https://software.intel.com/en-us/qualify-for-free-software
To enable all 3rd party builds available in OpenHPC that are compatible with Intel® Parallel Studio, issue the following:

```bash
# Install OpenHPC compatibility packages (requires prior installation of Parallel Studio)
[sms]# yum -y install intel-compilers-devel-ohpc
[sms]# yum -y install intel-mpi-devel-ohpc

# Optionally, choose Omni-Path enabled MPI builds. Otherwise, skip to retain default MPI stacks
[sms]# yum -y install openmpi-psm2-intel-ohpc mvapich2-psm2-intel-ohpc

# Install 3rd party libraries/tools built with Intel toolchain
[sms]# yum -y groupinstall ohpc-serial-libs-intel
[sms]# yum -y groupinstall ohpc-serial-libs-intel
[sms]# yum -y groupinstall ohpc-perf-tools-intel
[sms]# yum -y groupinstall ohpc-python-libs-intel
[sms]# yum -y groupinstall ohpc-runtimes-intel
[sms]# yum -y groupinstall ohpc-parallel-libs-intel-mpich
[sms]# yum -y groupinstall ohpc-parallel-libs-intel-mvapich2
[sms]# yum -y groupinstall ohpc-parallel-libs-intel-openmpi
[sms]# yum -y groupinstall ohpc-parallel-libs-intel-mpi
```

## 5 Resource Manager Startup

In section §3, the Slurm resource manager was installed and configured for use on both the master host and compute node instances. With the cluster nodes up and functional, we can now startup the resource manager services in preparation for running user jobs. Generally, this is a two-step process that requires starting up the controller daemons on the master host and the client daemons on each of the compute hosts. Note that Slurm leverages the use of the munge library to provide authentication services and this daemon also needs to be running on all hosts within the resource management pool. The following commands can be used to startup the necessary services to support resource management under Slurm.

```bash
# start munge and slurm controller on master host
[sms]# systemctl enable munge
[sms]# systemctl enable slurmctld
[sms]# systemctl start munge
[sms]# systemctl start slurmctld

# start slurm clients on compute hosts
[sms]# pssh -w c[1-4] systemctl start slurmd
```
6 Run a Test Job

With the resource manager enabled for production usage, users should now be able to run jobs. To demonstrate this, we will add a “test” user on the master host that can be used to run an example job.

```bash
[sms]# useradd -m test
```

Warewulf installs a utility on the compute nodes to automatically synchronize known files from the provisioning server at five minute intervals. In this recipe, recall that we previously registered credential files with Warewulf (e.g. passwd, group, and shadow) so that these files would be propagated during compute node imaging. However, with the addition of a new “test” user above, the files have been outdated and we need to update the Warewulf database to incorporate the additions. This re-sync process can be accomplished as follows:

```bash
[sms]# wwsh file resync passwd shadow group
```

**Tip**

After re-syncing to notify Warewulf of file modifications made on the master host, it should take approximately 5 minutes for the changes to propagate. However, you can also manually pull the changes from compute nodes via the following:

```bash
[sms]# pdsh -w c[1-4] /warewulf/bin/wwgetfiles
```

OpenHPC includes a simple “hello-world” MPI application in the `/opt/ohpc/pub/examples` directory that can be used for this quick compilation and execution. OpenHPC also provides a companion job-launch script named `prun` that is installed in concert with the pre-packaged MPI toolchains. At present, OpenHPC is unable to include the PMI process management server normally included within Slurm which implies that `srun` cannot be use for MPI job launch. Instead, native job launch mechanisms provided by the MPI stacks are utilized and `prun` abstracts this process for the various stacks to retain a single launch command.
6.1 Interactive execution

To use the newly created “test” account to compile and execute the application *interactively* through the resource manager, execute the following (note the use of `prun` for parallel job launch which summarizes the underlying native job launch mechanism being used):

```bash
# 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 and use prun to launch executable
[test@sms ~]$ srun -n 8 -N 2 --pty /bin/bash

[test@c1 ~]$ prun ./a.out

[prun] Master compute host = c1
[prun] Resource manager = slurm
[prun] Launch cmd = mpiexec.hydra -bootstrap slurm ./a.out

Hello, world (8 procs total)
--> Process # 0 of 8 is alive. -> c1
--> Process # 4 of 8 is alive. -> c2
--> Process # 1 of 8 is alive. -> c1
--> Process # 5 of 8 is alive. -> c2
--> Process # 2 of 8 is alive. -> c1
--> Process # 6 of 8 is alive. -> c2
--> Process # 3 of 8 is alive. -> c1
--> Process # 7 of 8 is alive. -> c2
```

### Tip

The following table provides approximate command equivalences between SLURM and PBS Pro:

<table>
<thead>
<tr>
<th>Command</th>
<th>PBS Pro</th>
<th>SLURM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit <em>batch</em> job</td>
<td><code>qsub [job script]</code></td>
<td><code>sbatch [job script]</code></td>
</tr>
<tr>
<td>Request <em>interactive</em> shell</td>
<td><code>qsub -I /bin/bash</code></td>
<td><code>srun --pty /bin/bash</code></td>
</tr>
<tr>
<td>Delete job</td>
<td><code>qdel [job id]</code></td>
<td><code>scancel [job id]</code></td>
</tr>
<tr>
<td>Queue status</td>
<td><code>qstat -q</code></td>
<td><code>sinfo</code></td>
</tr>
<tr>
<td>Job status</td>
<td><code>qstat -f [job id]</code></td>
<td><code>scontrol show job [job id]</code></td>
</tr>
<tr>
<td>Node status</td>
<td><code>pbsnodes [node name]</code></td>
<td><code>scontrol show node [node id]</code></td>
</tr>
</tbody>
</table>
6.2 Batch execution

For batch execution, OpenHPC provides a simple job script for reference (also housed in the `/opt/ohpc/pub/examples` directory). This example script can be used as a starting point for submitting batch jobs to the resource manager and the example below illustrates use of the script to submit a batch job for execution using the same executable referenced in the previous interactive example.

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

# examine contents (and edit to set desired job sizing characteristics)
[test@sms ~]$ cat job.mpi
#!/bin/bash

#SBATCH -J test    # Job name
#SBATCH -o job.%j.out # Name of stdout output file (%j expands to %jobId)
#SBATCH -N 2          # Total number of nodes requested
#SBATCH -n 16    # Total number of mpi tasks #requested
#SBATCH -t 01:30:00    # Run time (hh:mm:ss) - 1.5 hours

# Launch MPI-based executable
prun ./a.out

# Submit job for batch execution
[test@sms ~]$ sbatch job.mpi
Submitted batch job 339
```

**Tip**

The use of the `%j` option in the example batch job script shown is a convenient way to track application output on an individual job basis. The `%j` token is replaced with the Slurm job allocation number once assigned (job #339 in this example).
Appendices

A Installation Template

This appendix highlights the availability of a companion installation script that is included with OpenHPC documentation. This script, when combined with local site inputs, can be used to implement a starting recipe for bare-metal system installation and configuration. This template script is used during validation efforts to test cluster installations and is provided as a convenience for administrators as a starting point for potential site customization.

Tip

Note that the template script provided is intended for use during initial installation and is not designed for repeated execution. If modifications are required after using the script initially, we recommend running the relevant subset of commands interactively.

The template script relies on the use of a simple text file to define local site variables that were outlined in §1.3. By default, the template installation script attempts to use local variable settings sourced from the /opt/ohpc/pub/doc/recipes/vanilla/input.local file, however, this choice can be overridden by the use of the $({OHPC\_INPUT\_LOCAL}) environment variable. The template install script is intended for execution on the SMS master host and is installed as part of the docs-ohpc package into /opt/ohpc/pub/doc/recipes/vanilla/recipe.sh. After enabling the OpenHPC repository and reviewing the guide for additional information on the intent of the commands, the general starting approach for using this template is as follows:

1. Install the docs-ohpc package

   [sms]\# yum -y install docs-ohpc

2. Copy the provided template input file to use as a starting point to define local site settings:

   [sms]\# cp /opt/ohpc/pub/doc/recipes/vanilla/input.local input.local

3. Update input.local with desired settings

4. Copy the template installation script which contains command-line instructions culled from this guide.


5. Review and edit recipe.sh to suite.

6. Use environment variable to define local input file and execute recipe.sh to perform a local installation.

   [sms]\# export OHPC\_INPUT\_LOCAL=./input.local
   [sms]\# ./recipe.sh
B Rebuilding Packages from Source

Users of OpenHPC may find it desirable to rebuild one of the supplied packages to apply build customizations or satisfy local requirements. One way to accomplish this is to install the appropriate source RPM, modify the specfile as needed, and rebuild to obtain an updated binary RPM. A brief example using the FFTW library is highlighted below. Note that the source RPMs can be downloaded from the community build server at [https://build.openhpc.community](https://build.openhpc.community) via a web browser or directly via rpm as highlighted below. The OpenHPC build system design leverages several keywords to control the choice of compiler and MPI families for relevant development libraries and the rpmbuild example illustrates how to override the default mpi_family.

```
# Install rpm-build package from base OS distro
[test@sms ~]$ yum -y install rpm-build

# Download SRPM from OpenHPC repository and install locally
[test@sms ~]$ rpm -i http://build.openhpc.community/OpenHPC:/1.2/CentOS_7.2/src/fftw-gnu-openmpi-ohpc-3.3.4-4.1.src.rpm

# Modify spec file as desired
[test@sms ~]$ cd ~/rpmbuild/SPECS
[test@sms ~/rpmbuild/SPECS]$ perl -pi -e "s/enable-static=no/enable-static=yes/" fftw.spec

# Increment RPM release so package manager will see an update
[test@sms ~/rpmbuild/SPECS]$ perl -pi -e "s/Release: 4.1/Release: 5.1/" fftw.spec

# Rebuild binary RPM. Note that additional directives can be specified to modify build
[test@sms ~/rpmbuild/SPECS]$ rpmbuild -bb --define "mpi_family mvapich2" fftw.spec

# As privileged user, install the new package
[smu]$ yum -y install ~/rpmbuild/RPMS/x86_64/fftw-gnu-mvapich2-ohpc-3.3.4-5.1.rpm
```
C Package Manifest

This appendix provides a summary of available convenience groups and all of the underlying RPM packages that are available as part of this OpenHPC release. The convenience groups are aliases that the underlying package manager supports in order to provide a mechanism to group related collections of packages together. The collection of packages (along with any additional dependencies) can be installed by using the group name. A list of the available convenience groups and a brief description are presented in Table 2.

Table 2: Available OpenHPC Convenience Groups

<table>
<thead>
<tr>
<th>Group Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ohpc-autotools</td>
<td>Collection of GNU autotools packages.</td>
</tr>
<tr>
<td>ohpc-base</td>
<td>OpenHPC base packages.</td>
</tr>
<tr>
<td>ohpc-ganglia</td>
<td>Collection of Ganglia monitoring and metrics packages.</td>
</tr>
<tr>
<td>ohpc-io-libs-gnu</td>
<td>OpenHPC IO library builds for use with GNU compiler toolchain.</td>
</tr>
<tr>
<td>ohpc-io-libs-intel</td>
<td>OpenHPC IO library builds for use with Intel parallel studio toolchain.</td>
</tr>
<tr>
<td>ohpc-nagios</td>
<td>Collection of Nagios monitoring and metrics packages.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-gnu-mpich</td>
<td>OpenHPC parallel library builds for use with GNU compiler toolchain and the MPICH runtime.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-gnu-mvapich2</td>
<td>OpenHPC parallel library builds for use with GNU compiler toolchain and the MVAPICH2 runtime.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-gnu-openmpi</td>
<td>OpenHPC parallel library builds for use with GNU compiler toolchain and the OpenMPI runtime.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-intel-impi</td>
<td>OpenHPC parallel library builds for use with Intel Parallel Studio toolchain and the Intel MPI runtime.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-intel-mpich</td>
<td>OpenHPC parallel library builds for use with Intel Parallel Studio toolchain and the MPICH runtime.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-intel-mvapich2</td>
<td>OpenHPC parallel library builds for use with Intel Parallel Studio toolchain and the MVAPICH2 runtime.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-intel-openmpi</td>
<td>OpenHPC parallel library builds for use with Intel Parallel Studio toolchain and the OpenMPI runtime.</td>
</tr>
<tr>
<td>ohpc-perf-tools-gnu</td>
<td>OpenHPC performance tool builds for use with GNU compiler toolchain.</td>
</tr>
<tr>
<td>ohpc-perf-tools-intel</td>
<td>OpenHPC performance tool builds for use with Intel Parallel Studio toolchain.</td>
</tr>
<tr>
<td>ohpc-python-libs-gnu</td>
<td>OpenHPC python related library builds for use with GNU compiler toolchain.</td>
</tr>
<tr>
<td>ohpc-python-libs-intel</td>
<td>OpenHPC python related library builds for use with Intel Parallel Studio toolchain.</td>
</tr>
<tr>
<td>ohpc-runtimes-gnu</td>
<td>OpenHPC runtimes for use with GNU compiler toolchain.</td>
</tr>
<tr>
<td>ohpc-runtimes-intel</td>
<td>OpenHPC runtimes for use with Intel Parallel Studio toolchain.</td>
</tr>
<tr>
<td>ohpc-serial-libs-gnu</td>
<td>OpenHPC serial library builds for use with GNU compiler toolchain.</td>
</tr>
<tr>
<td>ohpc-serial-libs-intel</td>
<td>OpenHPC serial library builds for use with Intel Parallel Studio toolchain.</td>
</tr>
<tr>
<td>ohpc-slurm-server</td>
<td>OpenHPC server packages for SLURM.</td>
</tr>
<tr>
<td>ohpc-warewulf</td>
<td>Collection of base packages for Warewulf provisioning.</td>
</tr>
<tr>
<td>ohpc-parallel-libs-gnu</td>
<td>OpenHPC parallel library builds for use with GNU compiler toolchain.</td>
</tr>
<tr>
<td>ohpc-slurm-client</td>
<td>OpenHPC client packages for SLURM.</td>
</tr>
</tbody>
</table>
What follows next in this Appendix are a series of tables that summarize the underlying RPM packages available in this OpenHPC release. These packages are organized by groupings based on their general functionality and each table provides information for the specific RPM name, version, brief summary, and the web URL where additional information can be contained for the component. Note that many of the 3rd party community libraries that are pre-packaged with OpenHPC are built using multiple compiler and MPI families. In these cases, the RPM package name includes delimiters identifying the development environment for which each package build is targeted. Additional information on the OpenHPC package naming scheme is presented in §4.6 The relevant package groupings and associated Table references are as follows:

- Administrative tools (Table 3)
- Provisioning (Table 4)
- Resource management (Table 5)
- Compiler families (Table 6)
- MPI families (Table 7)
- Development tools (Table 8)
- Performance analysis tools (Table 9)
- Distro support packages and dependencies (Table 10)
- IO Libraries (Table 12)
- Runtimes (Table 13)
- Serial Libraries (Table 14)
- Parallel Libraries (Table 15)
<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>conman-ohpc</td>
<td>0.2.7</td>
<td>ConMan: The Console Manager. <a href="http://dun.github.io/conman">http://dun.github.io/conman</a></td>
</tr>
<tr>
<td>docs-ohpc</td>
<td>1.2</td>
<td>OpenHPC documentation. <a href="https://github.com/openhpc/ohpc">https://github.com/openhpc/ohpc</a></td>
</tr>
<tr>
<td>examples-ohpc</td>
<td>1.4</td>
<td>Example source code and templates for use within OpenHPC environment. <a href="https://github.com/openhpc/ohpc">https://github.com/openhpc/ohpc</a></td>
</tr>
<tr>
<td>ganglia-ohpc</td>
<td>3.7.2</td>
<td>Distributed Monitoring System. <a href="http://ganglia.sourceforge.net">http://ganglia.sourceforge.net</a></td>
</tr>
<tr>
<td>genders-ohpc</td>
<td>1.22</td>
<td>Static cluster configuration database. <a href="https://github.com/chaos/genders">https://github.com/chaos/genders</a></td>
</tr>
<tr>
<td>lmod-defaults-gnu-impi-ohpc</td>
<td>1.2</td>
<td>OpenHPC default login environments. <a href="https://github.com/openhpc/ohpc">https://github.com/openhpc/ohpc</a></td>
</tr>
<tr>
<td>lmod-defaults-gnu-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lmod-defaults-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lmod-defaults-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lmod-defaults-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lmod-defaults-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lmod-defaults-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lmod-defaults-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lmod-ohpc</td>
<td>6.5.11</td>
<td>Lua based Modules (lmod). <a href="https://github.com/TACC/Lmod">https://github.com/TACC/Lmod</a></td>
</tr>
<tr>
<td>losf-ohpc</td>
<td>0.53.0</td>
<td>A Linux operating system framework for managing HPC clusters. <a href="https://github.com/hpcsi/losf">https://github.com/hpcsi/losf</a></td>
</tr>
<tr>
<td>mrsf-ohpc</td>
<td>2.12</td>
<td>Remote shell program that uses munge authentication. <a href="https://github.com/chaos/genders">https://github.com/chaos/genders</a></td>
</tr>
<tr>
<td>nagios-ohpc</td>
<td>4.1.1</td>
<td>Nagios monitors hosts and services and yells if something breaks. <a href="http://www.nagios.org">http://www.nagios.org</a></td>
</tr>
<tr>
<td>nagios-plugins-ohpc</td>
<td>2.1.1</td>
<td>Host/service/network monitoring program plugins for Nagios. <a href="https://www.nagios-plugins.org">https://www.nagios-plugins.org</a></td>
</tr>
<tr>
<td>ohpc-release</td>
<td>1.2</td>
<td>OpenHPC release files. <a href="https://github.com/openhpc/ohpc">https://github.com/openhpc/ohpc</a></td>
</tr>
<tr>
<td>pdsh-ohpc</td>
<td>2.31</td>
<td>Parallel remote shell program. <a href="http://sourceforge.net/projects/pdsh">http://sourceforge.net/projects/pdsh</a></td>
</tr>
<tr>
<td>prun-ohpc</td>
<td>1.1</td>
<td>Convenience utility for parallel job launch. <a href="https://github.com/openhpc/ohpc">https://github.com/openhpc/ohpc</a></td>
</tr>
</tbody>
</table>
Table 4: Provisioning

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>warewulf-provision-ohpc</td>
<td>3.6</td>
<td>Warewulf - Provisioning Module. <a href="http://warewulf.lbl.gov">http://warewulf.lbl.gov</a></td>
</tr>
<tr>
<td>warewulf-provision-server-ohpc</td>
<td>3.6</td>
<td>Warewulf - Provisioning Module - Server. <a href="http://warewulf.lbl.gov">http://warewulf.lbl.gov</a></td>
</tr>
<tr>
<td>warewulf-vnfs-ohpc</td>
<td>3.6</td>
<td>Warewulf VNFS Module. <a href="http://warewulf.lbl.gov">http://warewulf.lbl.gov</a></td>
</tr>
</tbody>
</table>
Table 5: **Resource Management**

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>munge-ohpc</td>
<td>0.5.12</td>
<td>MUNGE authentication service. <a href="http://dun.github.io/munge">http://dun.github.io/munge</a></td>
</tr>
<tr>
<td>pbspro-client-ohpc</td>
<td>14.1.0</td>
<td>PBS Professional for a client host. <a href="https://github.com/pbspro/pbspro">https://github.com/pbspro/pbspro</a></td>
</tr>
<tr>
<td>pbspro-execution-ohpc</td>
<td>14.1.0</td>
<td>PBS Professional for an execution host. <a href="https://github.com/pbspro/pbspro">https://github.com/pbspro/pbspro</a></td>
</tr>
<tr>
<td>pbspro-server-ohpc</td>
<td>14.1.0</td>
<td>PBS Professional for a server host. <a href="https://github.com/pbspro/pbspro">https://github.com/pbspro/pbspro</a></td>
</tr>
<tr>
<td>slurm-devel-ohpc</td>
<td>16.05.5</td>
<td>Development package for Slurm. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-munge-ohpc</td>
<td>16.05.5</td>
<td>Slurm authentication and crypto implementation using Munge. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-ohpc</td>
<td>16.05.5</td>
<td>Slurm Workload Manager. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-pam_slurm-ohpc</td>
<td>16.05.5</td>
<td>PAM module for restricting access to compute nodes via Slurm. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-perlapi-ohpc</td>
<td>16.05.5</td>
<td>Perl API to Slurm. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-plugins-ohpc</td>
<td>16.05.5</td>
<td>Slurm plugins (loadable shared objects). <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-sjobexit-ohpc</td>
<td>16.05.5</td>
<td>Slurm job exit code management tools. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-sjstat-ohpc</td>
<td>16.05.5</td>
<td>Perl tool to print Slurm job state information. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-slurmd-direct-ohpc</td>
<td>16.05.5</td>
<td>Wrappers to write directly to the slurmdb. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-slurmdbd-ohpc</td>
<td>16.05.5</td>
<td>Slurm database daemon. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-sql-ohpc</td>
<td>16.05.5</td>
<td>Slurm SQL support. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
<tr>
<td>slurm-torque-ohpc</td>
<td>16.05.5</td>
<td>Torque/PBS wrappers for transition from Torque/PBS to Slurm. <a href="http://slurm.schedmd.com">http://slurm.schedmd.com</a></td>
</tr>
</tbody>
</table>

Table 6: **Compiler Families**

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnu-compilers-ohpc</td>
<td>5.4.0</td>
<td>The GNU C Compiler and Support Files. <a href="http://gcc.gnu.org">http://gcc.gnu.org</a></td>
</tr>
</tbody>
</table>
### Table 7: MPI Families

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>intel-mpi-devel-ohpc</td>
<td>2017</td>
<td>OpenHPC compatibility package for Intel(R) MPI Library. <a href="https://github.com/openhpc/ohpc">https://github.com/openhpc/ohpc</a></td>
</tr>
<tr>
<td>mpich-gnu-ohpc, mpich-intel-ohpc</td>
<td>3.2</td>
<td>MPICH MPI implementation. <a href="http://www.mpich.org">http://www.mpich.org</a></td>
</tr>
<tr>
<td>mvapich2-gnu-ohpc, mvapich2-intel-ohpc, mvapich2-psm2-gnu-ohpc, mvapich2-psm2-intel-ohpc</td>
<td>2.2</td>
<td>OSU MVAPICH2 MPI implementation. <a href="http://mvapich.cse.ohio-state.edu/overview/mvapich2">http://mvapich.cse.ohio-state.edu/overview/mvapich2</a></td>
</tr>
</tbody>
</table>

### Table 8: Development Tools

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_base-ohpc</td>
<td>3.3.1</td>
<td>R is a language and environment for statistical computing and graphics (S-Plus like). <a href="http://www.r-project.org">http://www.r-project.org</a></td>
</tr>
<tr>
<td>autoconf-ohpc</td>
<td>2.69</td>
<td>A GNU tool for automatically configuring source code. <a href="http://www.gnu.org/software/autoconf">http://www.gnu.org/software/autoconf</a></td>
</tr>
<tr>
<td>libtool-ohpc</td>
<td>2.4.6</td>
<td>The GNU Portable Library Tool. <a href="http://www.gnu.org/software/libtool">http://www.gnu.org/software/libtool</a></td>
</tr>
<tr>
<td>valgrind-ohpc</td>
<td>3.11.0</td>
<td>Valgrind Memory Debugger. <a href="http://www.valgrind.org">http://www.valgrind.org</a></td>
</tr>
</tbody>
</table>
## Table 9: Performance Analysis Tools

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>imb-gnu-impi-ohpc</td>
<td>4.1</td>
<td>Intel MPI Benchmarks (IMB).</td>
</tr>
<tr>
<td>imb-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>imb-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>imb-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>imb-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>imb-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>imb-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiP-gnu-impi-ohpc</td>
<td>3.4.1</td>
<td>mpiP: a lightweight profiling library for MPI applications.</td>
</tr>
<tr>
<td>mpiP-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiP-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiP-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiP-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiP-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiP-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>papi-ohpc</td>
<td>5.4.3</td>
<td>Performance Application Programming Interface.</td>
</tr>
<tr>
<td><a href="http://icl.cs.utk.edu/papi">http://icl.cs.utk.edu/papi</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdtoolkit-gnu-ohpc</td>
<td>3.22</td>
<td>PDT is a framework for analyzing source code.</td>
</tr>
<tr>
<td><a href="http://www.cs.uoregon.edu/Research/pdt">http://www.cs.uoregon.edu/Research/pdt</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdtoolkit-intel-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scalasca-gnu-impi-ohpc</td>
<td>2.3.1</td>
<td>Toolset for performance analysis of large-scale parallel applications.</td>
</tr>
<tr>
<td>scalasca-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.scalasca.org">http://www.scalasca.org</a></td>
</tr>
<tr>
<td>scalasca-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scalasca-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scalasca-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scalasca-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scalasca-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scalasca-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scorep-gnu-impi-ohpc</td>
<td>3.0</td>
<td>Scalable Performance Measurement Infrastructure for Parallel Codes.</td>
</tr>
<tr>
<td>scorep-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scorep-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scorep-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scorep-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scorep-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scorep-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sionlib-gnu-impi-ohpc</td>
<td>1.7.0</td>
<td>Scalable Performance Measurement Infrastructure for Parallel Codes.</td>
</tr>
<tr>
<td>sionlib-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sionlib-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sionlib-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sionlib-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sionlib-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sionlib-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tau-gnu-impi-ohpc</td>
<td>2.25.2</td>
<td>Tuning and Analysis Utilities Profiling Package.</td>
</tr>
<tr>
<td>tau-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.cs.uoregon.edu/research/tau/home.php">http://www.cs.uoregon.edu/research/tau/home.php</a></td>
</tr>
<tr>
<td>tau-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tau-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tau-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tau-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tau-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tau-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

36 Rev: 4f4aa01
Table 10: **Distro Support Packages/Dependencies**

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>lua-bit-ohpc</td>
<td>1.0.2</td>
<td>Module for Lua which adds bitwise operations on numbers. <a href="http://bitop.luajit.org">http://bitop.luajit.org</a></td>
</tr>
<tr>
<td>lua-filesystem-ohpc</td>
<td>1.6.3</td>
<td>Lua library to Access Directories and Files. <a href="http://keplerproject.github.com/luafilesystem">http://keplerproject.github.com/luafilesystem</a></td>
</tr>
<tr>
<td>lua-posix-ohpc</td>
<td>33.2.1</td>
<td>POSIX library for Lua. <a href="https://github.com/luaposix/luaposix">https://github.com/luaposix/luaposix</a></td>
</tr>
</tbody>
</table>

Table 11: **Lustre**

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>lustre-client-ohpc</td>
<td>2.8.0</td>
<td>Lustre File System.</td>
</tr>
</tbody>
</table>
## Table 12: IO Libraries

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>adios-gnu-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adios-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adios-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adios-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adios-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adios-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adios-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hdf5-gnu-ohpc</td>
<td>1.8.17</td>
<td>A general purpose library and file format for storing scientific data.</td>
</tr>
<tr>
<td>hdf5-intel-ohpc</td>
<td></td>
<td><a href="http://www.hdfgroup.org/HDF5">http://www.hdfgroup.org/HDF5</a></td>
</tr>
<tr>
<td>netcdf-cxx-gnu-impi-ohpc</td>
<td>4.2.1</td>
<td>C++ Libraries for the Unidata network Common Data Form.</td>
</tr>
<tr>
<td>netcdf-cxx-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.unidata.ucar.edu/software/netcdf">http://www.unidata.ucar.edu/software/netcdf</a></td>
</tr>
<tr>
<td>netcdf-cxx-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-cxx-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-cxx-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-cxx-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-cxx-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-cxx-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-fortran-gnu-impi-ohpc</td>
<td>4.4.4</td>
<td>Fortran Libraries for the Unidata network Common Data Form.</td>
</tr>
<tr>
<td>netcdf-fortran-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.unidata.ucar.edu/software/netcdf">http://www.unidata.ucar.edu/software/netcdf</a></td>
</tr>
<tr>
<td>netcdf-fortran-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-fortran-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-fortran-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-fortran-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-fortran-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-fortran-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-gnu-impi-ohpc</td>
<td>4.4.1</td>
<td>C Libraries for the Unidata network Common Data Form.</td>
</tr>
<tr>
<td>netcdf-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.unidata.ucar.edu/software/netcdf">http://www.unidata.ucar.edu/software/netcdf</a></td>
</tr>
<tr>
<td>netcdf-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netcdf-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phdf5-gnu-impi-ohpc</td>
<td>1.8.17</td>
<td>A general purpose library and file format for storing scientific data.</td>
</tr>
<tr>
<td>phdf5-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.hdfgroup.org/HDF5">http://www.hdfgroup.org/HDF5</a></td>
</tr>
<tr>
<td>phdf5-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phdf5-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phdf5-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phdf5-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phdf5-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phdf5-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 13: Runtimes

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocr-gnu-ohpc</td>
<td>1.0.1</td>
<td>Open Community Runtime (OCR) for shared memory. <a href="https://xstack.exascale-tech.com/wiki">https://xstack.exascale-tech.com/wiki</a></td>
</tr>
<tr>
<td>ocr-intel-ohpc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 14: Serial Libraries

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>metis-gnu-ohpc</td>
<td>5.1.0</td>
<td>Serial Graph Partitioning and Fill-reducing Matrix Ordering. <a href="http://glaros.dtc.umn.edu/gkhome/metis/metis/overview">http://glaros.dtc.umn.edu/gkhome/metis/metis/overview</a></td>
</tr>
<tr>
<td>metis-intel-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>openblas-gnu-ohpc</td>
<td>0.2.19</td>
<td>An optimized BLAS library based on GotoBLAS2. <a href="http://www.openblas.net">http://www.openblas.net</a></td>
</tr>
<tr>
<td>superlu-gnu-ohpc</td>
<td>5.2.1</td>
<td>A general purpose library for the direct solution of linear equations. <a href="http://crd.lbl.gov/~xiaoye/SuperLU">http://crd.lbl.gov/~xiaoye/SuperLU</a></td>
</tr>
<tr>
<td>superlu-intel-ohpc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Table 15: Parallel Libraries**

<table>
<thead>
<tr>
<th>RPM Package Name</th>
<th>Version</th>
<th>Info/URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>boost-gnu-impi-ohpc</td>
<td>1.61.0</td>
<td>Boost free peer-reviewed portable C++ source libraries.</td>
</tr>
<tr>
<td>boost-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.boost.org">http://www.boost.org</a></td>
</tr>
<tr>
<td>boost-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boost-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boost-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boost-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boost-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boost-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fftw-gnu-mpich-ohpc</td>
<td>3.3.4</td>
<td>A Fast Fourier Transform library.</td>
</tr>
<tr>
<td>fftw-gnu-mvapich2-ohpc</td>
<td></td>
<td><a href="http://www.fftw.org">http://www.fftw.org</a></td>
</tr>
<tr>
<td>fftw-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hypre-gnu-impi-ohpc</td>
<td>2.10.1</td>
<td>Scalable algorithms for solving linear systems of equations.</td>
</tr>
<tr>
<td>hypre-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.llnl.gov/casc/hypre">http://www.llnl.gov/casc/hypre</a></td>
</tr>
<tr>
<td>hypre-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hypre-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hypre-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hypre-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hypre-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hypre-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mumps-gnu-impi-ohpc</td>
<td>5.0.2</td>
<td>A MUltifrontal Massively Parallel Sparse direct Solver.</td>
</tr>
<tr>
<td>mumps-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mumps-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mumps-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mumps-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mumps-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mumps-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>petsc-gnu-impi-ohpc</td>
<td>3.7.0</td>
<td>Portable Extensible Toolkit for Scientific Computation.</td>
</tr>
<tr>
<td>petsc-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://www.mcs.anl.gov/petsc">http://www.mcs.anl.gov/petsc</a></td>
</tr>
<tr>
<td>petsc-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>petsc-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>petsc-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>petsc-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>petsc-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>petsc-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>superlu_dist-gnu-impi-ohpc</td>
<td>4.2</td>
<td>A general purpose library for the direct solution of linear equations.</td>
</tr>
<tr>
<td>superlu_dist-gnu-mpich-ohpc</td>
<td></td>
<td><a href="http://crd-legacy.lbl.gov/~xiaoye/SuperLU">http://crd-legacy.lbl.gov/~xiaoye/SuperLU</a></td>
</tr>
<tr>
<td>superlu_dist-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>superlu_dist-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>superlu_dist-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>superlu_dist-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>superlu_dist-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>superlu_dist-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinos-gnu-impi-ohpc</td>
<td>12.6.4</td>
<td>A collection of libraries of numerical algorithms.</td>
</tr>
<tr>
<td>trilinos-gnu-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinos-gnu-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinos-intel-impi-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinos-intel-mpich-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinos-intel-mvapich2-ohpc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trilinos-intel-openmpi-ohpc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D Package Signatures

All of the RPMs provided via the OpenHPC repository are signed with a GPG signature. By default, the underlying package managers will verify these signatures during installation to ensure that packages have not been altered. The RPMs can also be manually verified and the public signing key fingerprint for the latest repository is shown below:

Fingerprint: DD5D 8CAA CB57 364F FCC2 D3AE C468 07FF 26CE 6884

The following command can be used to verify an RPM once it has been downloaded locally by confirming if the package is signed, and if so, indicating which key was used to sign it. The example below highlights usage for a local copy of the docs-ohpc package and illustrates how the key ID matches the fingerprint shown above.

```
[sms]# rpm --checksig -v docs-ohpc-*.rpm
docs-ohpc-1.0-1.1.x86_64.rpm:
  Header V3 RSA/SHA256 Signature, key ID 26ce6884: OK
  Header SHA1 digest: OK (c3873bf495c51d2ea6d3ef23ab88be105983c72c)
  V3 RSA/SHA256 Signature, key ID 26ce6884: OK
  MD5 digest: OK (43d067f33fb370e30a39789439ead238)
```