Sun™ HPC Software, Linux Edition 2.0

Deployment and User Guide

Sun Microsystems, Inc. www.sun.com

Part No. 821-0037-10

June 2009

Table of Contents

Chapter 1:
Introduction

What is the Sun HPC Software, Linux Edition

Sun HPC Software, Linux Edition ("Sun HPC Software") is an integrated open-source software solution for Linux-based HPC clusters running on Sun hardware. It provides a framework of software components to simplify the process of deploying and managing large-scale Linux HPC clusters.

Who should use this document

This installation guide is written for administrators who are familiar with:

- Linux system administration
- High performance computing (HPC) concepts
- Cluster system configuration
- InfiniBand networks

What are the system requirements

The table below shows the Sun HPC Software system requirements.

Platforms	Sun x64 Servers
Operating Systems	Red Hat Enterprise Linux 5.3 (RHEL 5.3) CentOS 5.3 x86_64 (CentOS 5.3) SUSE Linux Enterprise Server 10 Service Pack 2 (SLES 10 SP2)
Networks	Ethernet, InfiniBand
Hard disk	Minimum 40 Gb disk space
RAM	Minimum 1 Gb

The Linux kernels supported by the Sun HPC Software installer are listed below. To determine the kernel version currently installed on the head node of your cluster, enter `uname -r`.

RHEL	RHEL 5.3 Release Kernel	2.6.18-128.el5
	RHEL 5.3 Errata Kernel	2.6.18-128.1.10.el5
SLES	SLES 10 SP2 Release Kernel	2.6.16.60-0.21-smp
	SLES 10 SP2 Errata Kernel	2.6.16.60-0.31-smp 2.6.16.60-0.37_f594963d-smp

How is this document organized

Chapter 1 provides useful information such as system requirements and sources for additional information. Chapter 2 describes how to install the Sun HPC Software on a head node and provision the client nodes in an HPC cluster. Chapters 3 and 4 describe tools for managing and monitoring an HPC cluster. Chapter 5 contains information about setting up a parallel computing environment to build and run Message Passing Interface (MPI)-based applications. Chapter 6 describes tools for managing compute resources. Appendix C provides descriptions of the Sun HPC Software components.

Where can I find additional information

Sun HPC Software, Linux Edition product page:
http://www.sun.com/software/products/hpcsoftware/

Lustre File System product page: http://www.sun.com/software/products/lustre/

Sun Grid Engine product page: http://www.sun.com/software/sge/

Sun ClusterTools product page: http://www.sun.com/software/products/clustertools/

Chapter 2:
Overview and Preparation

Installation Overview

The installation procedure described in this guide installs the Sun HPC Software on a cluster configured similar to that shown in Figure 1. This example cluster contains:

- **Head Node** - As part of the Sun HPC Software installation process, the Cobbler and oneSIS provisioning tools are installed on the head node. These tools are used for the provisioning of diskful and diskless cluster nodes. The head node must be connected to the cluster-wide provisioning network.
- **Client Nodes** - All nodes provisioned from the head node are referred to as clients of the head node. A client node may be provisioned in either a diskful or diskless configuration. Each client node must be connected to the cluster-wide provisioning network and are provisioned using the Cobbler/oneSIS-based provisioning system.

The Cobbler provisioning tool facilitates provisioning (via DHCP/PXE) of diskful or diskless node configurations. For diskless nodes, Cobbler uses the oneSIS system administration tool to provide NFS-mounted root file systems for each node class, such as a Lustre server, compute node, or login node.

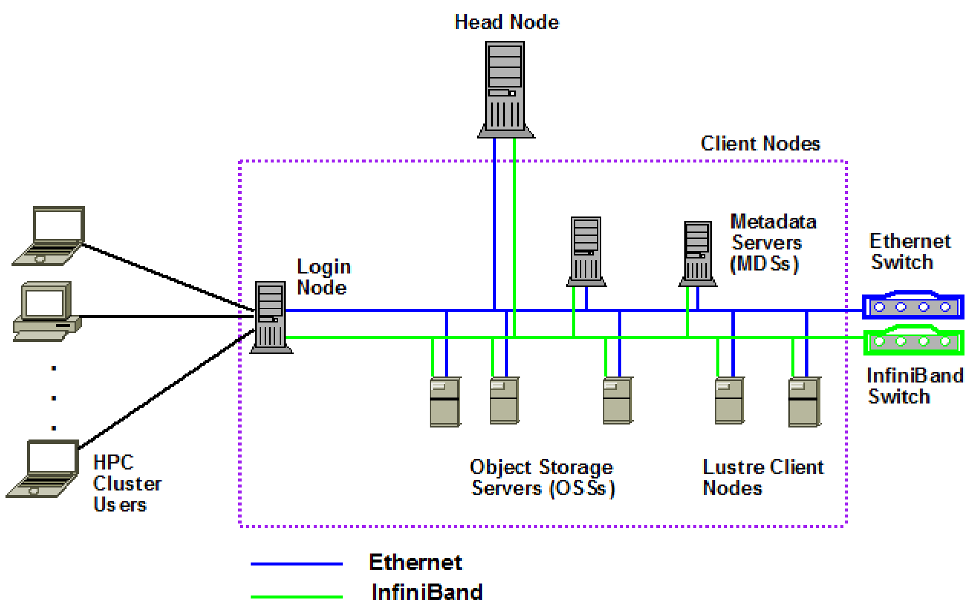

Figure 1. Example cluster configuration using an InfiniBand network as the compute network

Planning and Preparation

Before installing the Sun HPC Software, complete the preparations described in this section.

Installation considerations

Answers to the following questions will help you determine which procedures to follow in the installation instructions.

- Will you install the Sun HPC Software from a Sun repository accessed over the Internet or from an ISO image downloaded from the Sun website?
- Will you be running diskful clients or diskless clients?
- Will you be installing and configuring the Lustre file system?

Preparing to install the software

Before starting the installation procedure, prepare the items below:

- If you will be downloading the base operating system (base OS) or the Sun HPC Software from a remote repository, ensure the head node has access to the Internet (see Obtaining the Software).
- If you will be installing software from an ISO image, obtain the appropriate ISO images (see Obtaining the Software).
- Create an inventory of your cluster including the network configuration (for an example HPC cluster inventory, see Appendix A).

Obtaining the software

You will need access to a supported base operating system (OS):

- RHEL 5.3:
  Sign up for an account and download at https://www.redhat.com/apps/download/
- CentOS 5.3:
  Download at http://isoredirect.centos.org/centos/5/isos/x86_64/
- SLES 10 SP2:
  Download at http://download.novell.com/Download?buildid=xWohTS2zkSs~

  To obtain online updates for the SLES OS, you will need a license from Novell.
- SLE 10 SP2 SDK (required for provisioning diskful clients):
  Download at http://download.novell.com/Download?buildid=eRAdQttrkeA~

You will also need access to the Sun HPC Software, Linux Edition repository located at: http://www.sun.com/software/products/hpcsoftware/getit.jsp.

Note:

- RHEL 5.3 includes OFED 1.3.2, which is replaced by OFED 1.3.1 when the Sun HPC Software is installed on the head node.
- SLES 10 SP2 includes OFED 1.3, which is replaced by OFED 1.3.1 when the Sun HPC Software is installed on the head node.

Chapter 3:

Installing the Software and Provisioning the Cluster

The workflow for installing the Sun HPC Software on the head node and provisioning the clients is illustrated in Figure 2.

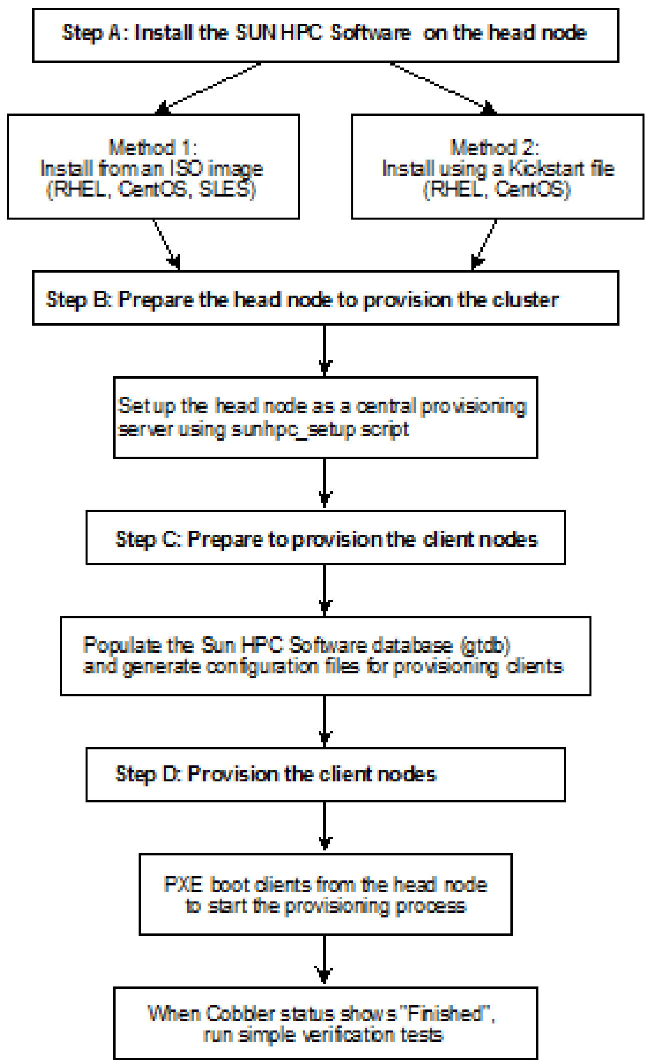

Figure 2. Installing and provisioning workflow

Step A. Install the Sun HPC Software on the Head Node

Overview

The Sun HPC Software installation process is designed to accommodate a variety of customer environments. Two recommended methods for installing the Sun HPC Software on the head node are:

- **Method 1 : Installing from an ISO image (RHEL, CentOS, or SLES).** An ISO image of the Sun HPC Software is downloaded from the Sun web site and optionally burned to a DVD. The ISO image contains all the software packages needed to deploy the Sun HPC Software on an existing Red Hat or SLES installation. It is assumed that the base Red Hat or SLES distribution is already installed on the head node.
- **Method 2: Using Kickstart (RHEL or CentOS only).** You can use this method if the head node has Internet access to the Sun HPC Software repository on a Sun-hosted server. The Kickstart automated installation tool allows a system administrator to perform a semi- or fully-automated installation of an RPM-based Linux system. A Kickstart-based installation of the Sun HPC Software results in a head node that is installed with the base Red Hat distribution and the Sun HPC Software and ready to configure. Using the Kickstart method ensures that the Linux distribution packages needed by the Sun HPC Software will be correctly installed.

To install the Sun HPC Software on the head node, choose Method 1 or Method 2.

Method 1: Install the Sun HPC Software from an ISO image

To install the Sun HPC Software from an ISO image on a RHEL or SLES system, complete the steps below.

1. Install the base operating system (RHEL 5.3 or SLES 10 SP2) on the head node. Refer to the appropriate vendor documentation for a detailed procedure:
   - RHEL 5.3: http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Installation_Guide/index.html
   - SLES 10 SP 2: http://www.novell.com/documentation/sles10/index.html
2. Check that the head node has access to a RHEL 5.3 or SLES 10 SP2 repository on the official vendor site or elsewhere. Some dependent packages will be installed from the OS repository. If you are unable to configure this access, you may need to install updated packages, such as updated kernels, manually on the head node.

3. Create a Linux distribution software repository to be used by Cobbler and oneSIS when provisioning the client nodes in the cluster.

   - ***For RHEL 5.3***, create a software repository by completing these steps:

     a. Download the RHEL 5.3 ISO image to the head node. (For this example, it is placed in `/root/iso/`.)

     b. Add the following line to `/etc/fstab` (enter on one line):

```
/root/iso/rhel-server-5.3-x86_64-dvd.iso /mnt/rhel5.3 \
  iso9660 ro,loop 0 0
```

     c. Mount the file containing the RHEL 5.3 ISO image to the mount point `/mnt/rhel5.3` by entering:

```
# mkdir -p /mnt/rhel5.3
# mount -a
```

     d. Create a configuration file for the RHEL repository:

```
# cat /etc/yum.repos.d/rhel.repo
[rhel]
name=Red Hat Enterprise Linux DVD
baseurl=file:///mnt/rhel5.3/Server
enabled=1
gpgcheck=0
```

   - ***For CentOS 5.3***, create a software repository by completing these steps:

     a. Download the CentOS 5.3 ISO image to the head node. (For this example, it is placed in `/root/iso/`.)

     b. Add the following line to `/etc/fstab` (enter on one line):

```
/root/iso/CentOS5.3-x86_64-bin-DVD.iso /mnt/centos5.3 \
  iso9660 ro,loop 0 0
```

     c. Mount the file containing the CentOS 5.3 ISO image to the mount point`/mnt/centos5.3`.

```
# mkdir -p /mnt/centos5.3
# mount -a
```

     d. Create a configuration file for the RHEL repository:

```
# cat /etc/yum.repos.d/centos.repo
[centos]
name=CentOS DVD
baseurl=file:///mnt/centos5.3
enabled=1
gpgcheck=0
```

- ***For SLES 10 SP2***, create a software repository by completing these steps:

  a. Download the two ISO images `SLES-10-SP2-DVD-x86_64-GM-DVD1.iso` and `SLE-10-SP2-SDK-DVD-x86_64-GM-DVD1.iso` to the head node. (For this example, they are placed in `/root/iso/`.)

  b. Add the following two lines to `/etc/fstab` (include each complete entry on one line):

```
/root/iso/SLE-10-SP2-SDK-DVD-x86_64-GM-DVD1.iso \
  /media/sles10sdk iso9660 ro,loop  0 0
/root/iso/SLES-10-SP2-DVD-x86_64-GM-DVD1.iso \
  /media/sles10sp2 iso9660 ro,loop 0 0
```

  c. Mount the files containing the SLES ISO images.

```
# mkdir -p /media/sles10sdk
# mkdir -p /media/sles10sp2
# mount -a
```

  d. Add both mount points as software sources:

```
# zypper sa file:///media/sles10sp2/
# zypper sa file:///media/sles10sdk/
```

4. Check if `dialog` is installed by entering:

```
# rpm -qa |grep dialog
```

5. If dialog is not installed, use the appropriate command below to install it.

   - ***For RHEL 5.3***, enter:

```
# yum install dialog
```

   - ***For SLES 10 SP2***, enter:

```
# zypper install dialog
```

6. Mount the Sun HPC Software ISO file and install the installation script.

   a. Download the Sun HPC Software, Linux Edition 2.0 ISO from the Sun website: http://www.sun.com/software/products/hpcsoftware/getit.jsp

   b. Choose one of these two options:

      - Burn the ISO image to a DVD disk and insert the disk into the head node DVD disk drive.

```
# mkdir -p /media/sun_hpc_linux
# mount -o ro /dev/dvd /media/sun_hpc_linux
```

- Mount the ISO image to `/media/sun_hpc_linux` on the head node.

  For RHEL 5.3 or CentOS 5.3:

  i. Add the following line to `/etc/fstab`:

```
/root/iso/sun-hpc-linux-rhel-2.0.iso /media/sun_hpc_linux \
iso9660 ro,loop 0 0
```

  ii. Mount the file containing the ISO image by entering:

```
#  mkdir -p /media/sun_hpc_linux
# mount -a
```

  For SLES 10 SP2:

  i. Add the following line to /etc/fstab:

```
/root/iso/sun-hpc-linux-sles-2.0.iso /media/sun_hpc_linux \
iso9660 ro,loop 0 0
```

  ii. Mount the file containing the ISO image by entering:

```
# mkdir -p /media/sun_hpc_linux
# mount -a
```

Note: The mount point must be `/media/sun_hpc_linux`. If you are using a Gnome desktop environment, the Gnome automount utility will automatically mount the ISO to `/media/sun_hpc_linux`.

c. Install the Sun HPC Software configuration files and installer scripts by entering:

```
# rpm -ivh /media/sun_hpc_linux/SunHPC/sunhpc-release.rpm
```

7. Install the Sun HPC Software RPMs on the head node. Run the software installer script `sunhpc_install` by entering:

```
# sunhpc_install
```

During installation, the Sun HPC Software installer may display messages similar to the following :

```
Welcome to the SunHPC Stack Linux Edition 2.0
 This installer will prepare this node to be
 the head node of a SunHPC Linux cluster
 running a Linux OS.
```

```
The kernel version 2.6.18-128.el5 is supported
 by Sun HPC Software, Linux Edition 2.0.
```

```
Checking OS repositories.
 Please wait.
```

```
 The installer has detected a SLES SDK source
 and an activated SLES Update repository.
```

```
Checking access to SunHPC repositories.
 Please wait.
Install logs are in /var/tmp/sunhpc_install.29917.log
```

```
Installation of the SunHPC head node
 is complete.
Install logs are in /var/tmp/sunhpc_install.29917.log
```

The Sun HPC Software installer may display warning messages similar to the following to indicate a problem that will prevent successful installation of the Sun HPC Software:

```
The currently installed kernel version is not supported.
Please use yum to install the kernel-2.6.18-128.el5
and reboot the head node. Then run sunhpc_install again.
```

```
The installer could NOT detect a SLES base install source. Access
 to the SLES base install source is required to complete this
installation.
 Please add a SLES base install source and run sunhpc_install
 again. A zypper search for [certain packages] failed.
```

```
The installer could NOT detect a SLES SDK install source. Access
 to the SLES SDK install source is required to complete this
installation.
 Please add an SLE SDK install source and run sunhpc_install
 again. A zypper search for [certain packages] failed.
```

```
The installer could NOT detect a RHEL/CentOS base install source. Access
 to the RHEL/CentOS base install source is required to complete this
installation.
 Please add a RHEL/CentOS base install source and run sunhpc_install
again.
```

Method 2. Use Kickstart to install RHEL and the Sun HPC Software

To install RHEL 5.3 and the Sun HPC Software to the head node using Kickstart, follow the steps below:

1. Check that the Sun HPC Software 2.0 repository is accessible from the network to which your head node is connected. The Sun HPC Software repository is located at: http://www.sun.com/software/products/hpcsoftware/getit.jsp
2. Insert the RHEL 5.3 DVD in the head node DVD drive and power on or reboot the node. If the BIOS has been configured to boot from the DVD device, a `boot:` prompt will appear.
3. At the `boot:` prompt, enter the following to configure the boot parameters:

```
boot: linux ks=http://dlc.sun.com/linux_hpc/ks/rhel5-2.0.cfg ip=dhcp
```

The wizard initial screen shown in Figure 3 will appear.

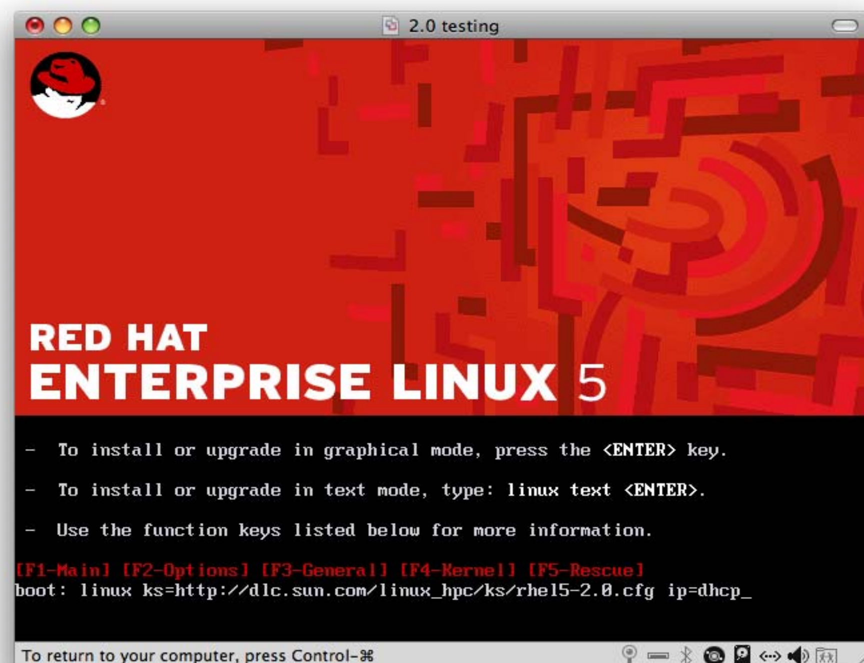

Figure 3. Kickstart wizard screen

4. Follow the wizard instructions to set the time zone, network configuration, hostname and root password.

Note: For IP and hostname, chose either “manual” or “automatic” for both. Mixing manual and automatic configurations is known to cause installation failure. Refer to known issues in the *Release Notes* for more information.

The head node will reboot when the installation process completes. RHEL 5.3 and the Sun HPC Software are now installed.

5. Create a Linux distribution software repository to be used by Cobbler and oneSIS when provisioning the client nodes in the cluster.

   - ***For RHEL 5.3***, create a software repository by completing these steps:

     a. Download the RHEL 5.3 ISO image to the head node. (For this example, it is placed in `/root/iso/`.)

     b. Mount the file containing the RHEL 5.3 ISO image to the mount point `/mnt/rhel5.3`.

     ```
     # mkdir -p /mnt/rhel5.3
     # mount -t iso9660 -o loop \
     /root/iso/rhel-server-5.3-x86_64-dvd.iso /mnt/rhel5.3
     ```

     c. Create a configuration file for the RHEL repository:

     ```
     # cat /etc/yum.repos.d/rhel.repo
     [rhel]
     name=Red Hat Enterprise Linux DVD
     baseurl=file:///mnt/rhel5.3/Server
     enabled=1
     gpgcheck=0
     ```

   - ***For CentOS 5.3***, create a software repository by completing these steps:

     a. Download the CentOS 5.3 ISO image to the head node. (For this example, it is placed in `/root/iso/`.)

     b. Mount the file containing the CentOS 5.3 ISO image to the mount point`/mnt/centos5.3`.

     ```
     # mkdir -p /mnt/centos5.3
     # mount -t iso9660 -o loop \
       /root/iso/CentOS-5.3-x86_64-bin-DVD.iso /mnt/centos5.3
     ```

c. Create a configuration file for the RHEL repository:

```
# cat /etc/yum.repos.d/centos.repo
[centos]
name=CentOS DVD
baseurl=file:///mnt/centos5.3
enabled=1
gpgcheck=0
```

Step B. Prepare the Head Node to Provision the Cluster

Overview

The software configuration script `sunhpc_setup` sets up a central provisioning server for deploying Sun HPC Software on a compute cluster. When this script is run, all the steps needed to provision client images are carried out. The script runs the necessary Cobbler commands to set up diskful Kickstart files for RHEL or CentOS or AutoYaST files for SLES to install the operating system and the Sun HPC Software on diskful nodes. It also runs the necessary oneSIS commands to create diskless images.

The `sunhpc_setup` script has several key functions:

- Builds provision software sources from multiple sources such as local repositories, remote repositories accessed over the Internet, or downloaded ISO images.
- Configures a DHCP service, Cobbler service and Kickstart service on the head node.
- Supports both diskless configuration and diskful configurations.
- Generates root `.ssh` keys, e.g. `id_rsa`, `id_rsa.pub`, if they were not created earlier.
- Configures password-less `ssh` between the head node and the provisioned diskful and diskless clients.
- Configures Cfengine for the head node and diskful clients. See Setting up Cfengine to Manage Configuration Files on Clients for more information about updating diskless clients.

By default, oneSIS images are stored in `/var/lib/oneSis/image`. If you need information for debugging, the log file for `sunhpc_setup` is located at `/var/tmp/sunhpc_setup.log` and the oneSIS log can be found at `/tmp/onesis_lustre_rootfs.log`.

You will run `sunhpc_setup` after the head node has been installed to set up the head node to provision clients. The `sunhpc_setup` script can be run multiple times to set up multiple profiles. However, cross-OS support is not provided, so you cannot create SLES clients on a RHEL head node. If you need to set up both diskless and diskful clients, you can run two sunhpc_setup commands back to back as shown in the example below:

```
# Configure centos 5.3 cobbler profile for diskful nodes with remote repo
sunhpc_setup --profile=centos5.3 --distro-image=/mnt/centos5.3 \
  --sunhpc-repo=http://giraffe.lustre.sun.com/dlc_stage/yum/sunhpc/\
  trunk/rhel --netif=eth1 --bootdisk=hda

# Configure centos 5.3 cobbler profile for diskless nodes
sunhpc_setup --profile=centos5.3-onesis --diskless --netif=eth1
```

The above commands will create four profiles to use when provisioning clients:

```
# cobbler list |grep profile
   profile centos5.3
   profile centos5.3-lustre
   profile centos5.3-onesis
   profile centos5.3-onesis-lustre
```

The sunhpc_setup script also builds repositories for provisioned clients. In the above example, the following repositories were created:

```
repo sunhpc_base_centos5.3
repo sunhpc_extras_centos5.3
repo sunhpc_lustre_centos5.3
repo sunhpc_updates_centos5.3
```

At a minimum, `sunhpc_setup` must be supplied with:

- The name of a provisioning configuration, using the `--profile` option.
- The location of the Linux base installation media, using the `--distro-image` option, which supports SLES 10.2, RHEL 5.3 and CentOS 5.3.
- The location of the Sun HPC Software, Linux Edition repositories, using the `--sunhpc-repo` option.
- The network interface connecting the HPC cluster nodes to be provisioned, using the `--netif` option. This is the network interface the head node will use to communicate with the provisioned client nodes.

Note: When `sunhpc_setup` is run on the head node, `iptables` are disabled. The `sunhpc_setup` script includes the following steps:

```
     iptables stop
     chkconfig –del iptables
```

If clients are provisioned with `iptables`, `iptables` will no longer be running after `sunhpc_setup` is run. Please secure your system as necessary before running `sunhpc_setup` to provision clients.

Enter `sunhpc_setup -h` to view the `sunhpc_setup` command options:

```
# sunhpc_setup -h

usage: /usr/sbin/sunhpc_setup options

OPTIONS:
-h, --help                      show this message
--profile=NAME                  profile name
--diskless                      diskless cluster configuration with oneSIS
--release=RELEASE               use different release name
                                (e.g. --release=2.0)
--netif=NETWORK_INTERFACE       provisioning network interface
                                (default: eth0)
--distro-image=PATH             top of distribution's image tree
--bootparams=PARAMS             additional boot parameters
--sunhpc-repo=URL|DIR           custom SunHPC repository location
--sdk-repo=DIR                  SLES SDK repository location
--bootdisk=diskname             boot disk device (default: sda)
--remove-repo=<all|repo name>   remove repo configurations in cobbler
--remove-profile=<all|profile name>
                                remove profile configurations in cobbler
--remove-all                    remove all cobbler configuration data
                                 ('distro','profile', 'system', 'repo')
--extras-repo                   use SunHPC 'extras' repo
--onesis-rootfs=DIR             root image directory for oneSIS client
--onesis-config=CONFIG_PATH     oneSIS's configuration path (default: auto
                                 detect)
--onesis-exclude=DIR            directories to be excluded from copy-rootfs
--skip-onesis-rootfs            do not create new oneSIS image, use existing
--onesis-no-lustre              do not configure diskless lustre server
                                 components
--gateway=IP address            dhcp client gateway IP address
```

Additional notes:

`--profile` may have any name. However in this document the following conventions are used:

- `rhel5.3` specifies a RHEL 5.3 profile.
- `centos5.3` specifies a CentOS 5.3 profile.
- `sles10.2` specifies a SLES 10.2 profile.
- `-lustre` specifies a Lustre server node, such as an object storage server or metadata server.
- `-onesis` specifies a diskless profile created by oneSIS.

After `sunhpc_setup` has been run, running the command `cobbler list |grep` profile will show which profiles have been created.

`--onesis-exclude=/root` excludes the specified directory (in this example, the `root` home directory `/root`). This option is usually used to exclude a very large directory. Be careful when using this option to make sure the directory *does not* contain required files (such as login files for `root`).

`--diskless` sets up a oneSIS image for diskless clients instead of creating Cobbler repositories.

`--sunhpc-repo=` should be set to point to the Sun HPC Software repository at either a local ISO mount point (for example, `/media/sun_hpc_linux`) or the Sun online repository (`dlc.sun.com`).

`--bootparams=` can be used to add more customized boot parameters for client nodes. For example, the serial console on some types of servers might be attached to `com2` instead of `com1`, which would appear to be `ttyS1` instead of `ttyS0`. For example, to change the default console setting, include the option `--bootparams="console=ttyS1,9600"` at the end of the `sunhpc_setup` command.

`--bootdisk=` can be used to support disk types other than a SATA hard disk (`sda`). For example, the flash drive in a Sun Fire x4600 Server would be specified as `bootdisk =hda`.

`--extras-repo` can be used to enable the repository to install a `perfctr` patched kernel and `perfctr` user library.

Note: If your cluster will boot the client nodes over an InfiniBand network, refer to *Appendix B: Using Boot Over IB (BoIB) to Deploy Diskless Clients*. The procedure in Appendix B assumes Ethernet access is available to all clients to perform the initial provisioning.

Most newer Sun systems ship with firmware already enabled for BoIB. Check with your Sun customer service rep for more information.

To prepare the head node running RHEL 5.3 to serve as the central provisioning server, complete the procedure Preparing a head node running RHEL 5.3. To prepare the head node running SLES 10 SP2 to serve as the central provisioning server, complete the procedure Preparing a head node running SLES 10 SP2.

Preparing a head node running RHEL 5.3

To prepare a head node running RHEL 5.3 to serve as the central provisioning server for the client nodes, follow the procedure in this section. For CentOS 5.3, change “`rhel5.3`” to “`centos5.3`” each time it occurs.

A Cobbler profile will be set up to be used to provision the compute cluster. The examples shown assume that the head node has two network interfaces: `eth0` connects to the Internet or public network; `eth1` connects to the rest of the HPC cluster nodes and serves as a DHCP interface.

Complete the steps below:

1. Check that both the RHEL 5.3 ISO image and the Sun HPC Software ISO image are mounted on the RHEL head node. The output of the `mount` command should contain the snippets below:

```
# mount
--snip--
/root/iso/rhel-server-5.3-x86_64-dvd.iso on /mnt/rhel5.3 \
  type iso9660 (rw,loop=/dev/loop0)
/root/sun-hpc-linux-rhel-trunk-beta2.iso on /media/sun_hpc_linux \
  type iso9660 (rw,loop=/dev/loop1)
--snip--
```

2. To provision the Cobbler repository, complete one or more of the options below for diskful, diskless, and perfctr clients, as appropriate for your HPC cluster.

   - ***If diskful clients using a RHEL operating system** are to be provisioned:*

     a. Enter a command similar to the following, where the head node connects to the client nodes on Ethernet interface `eth1`:

```
# sunhpc_setup --profile=rhel5.3 --distro-image=/mnt/rhel5.3 \
  --sunhpc-repo=/media/sun_hpc_linux --netif=eth1

Initializing Cobbler configuration... Done
Disabling the iptables... Done
Restarting dhcpd/cobblerd/httpd... Done
Copying /mnt/rhel5.3 to /var/www/cobbler/ks_mirror/rhel5.3... Done
Created 'sunhpc_base_rhel5.3' repo and copying... Done
Created 'sunhpc_extras_rhel5.3' repo and copying... Done
Created 'sunhpc_lustre_rhel5.3' repo and copying... Done
Created 'sunhpc_updates_rhel5.3' repo and copying... Done
Creating distro 'rhel5.3' in cobbler... Done
Creating profile 'rhel5.3' in cobbler... Done
Creating profile 'rhel5.3-lustre' in cobbler... Done
```

     b. Generate a Cobbler profile report to check that the Cobbler profiles `rhel5.3` and `rhel5.3-lustre` have been created. `rhel5.3` is the profile for diskful Lustre client nodes. `rhel5.3-lustre` is the profile for diskful Lustre server nodes, which will run on a Lustre patched kernel.

```
# cobbler profile list
rhel5.3
rhel5.3-lustre
```

- ***If diskless clients using a RHEL operating system*** *are to be provisioned:*

a. Enter the following, where the head node connects to the client nodes on Ethernet interface `eth1`:

```
# sunhpc_setup --profile=rhel5.3-onesis --diskless --netif=eth1
```

Output similar to the following will be displayed

```
Initializing Cobbler configuration... Done
Disabling the iptables... Done
Restarting dhcpd/cobblerd/httpd... Done
Copying / to /var/lib/oneSIS/image/rhel5.3-onesis... Done
Creating initrd... Done
Applying OneSIS configuration... Done
Updated /etc/exports and restarting NFS... Done
Copying /var/lib/oneSIS/image/rhel5.3-onesis to
/var/lib/oneSIS/image/rhel5.3-onesis-lustre ... Done.
Un-specializing rhel5.3-onesis-lustre ... Done.
Removing SunHPC Lustre Client group from rhel5.3-onesis-lustre ...
Done.
Installing perl-TimeDate from distro... Done.
Installing compat-libcom_err from distro... Done.
Installing uuidd from distro... Done.
Installing libnet from distro... Done.
Installing python-xml from distro... Done.
Upgrading e2fsprogs for ldiskfs support... Done.
Removing base kernel from rhel5.3-onesis-lustre ... Done.
Installing SunHPC Lustre Server group to rhel5.3-onesis-lustre ...
Done.
Creating oneSIS initrd for rhel5.3-onesis-lustre ... Done.
Converting rhel5.3-onesis-lustre to oneSIS rootfs image ... Done.
/var/lib/oneSIS/image/rhel5.3-onesis-lustre is already in
/etc/exports
Now (re)starting NFS... Done.
Creating distro 'rhel5.3-onesis' in cobbler... Done
Creating distro 'rhel5.3-onesis-lustre' in cobbler... Done
Creating profile 'rhel5.3-onesis' in cobbler... Done
Creating profile 'rhel5.3-onesis-lustre' in cobbler... Done
```

This command creates two oneSIS system images, one for diskless Lustre client nodes and one for diskless Lustre server nodes, in the directory `/var/lib/oneSIS/image` on the head node.

```
# ls /var/lib/oneSIS/image
rhel5.3-onesis rhel5.3-onesis-lustre
```

b. Generate a Cobbler profile report to check that the Cobbler profiles `rhel5.3-onesis` and `rhel5.3-onesis-lustre` have been created. `rhel5.3-onesis` is the profile for diskless Lustre client nodes. `rhel5.3-lustre-onesis` is the profile for diskless Lustre server nodes, which will run on a Lustre patched kernel.

```
# cobbler profile list
rhel5.3-onesis
rhel5.3-onesis-lustre
```

Note: The procedure of creating a diskless image can be broken into several manual steps. This may be useful when you need to preserve an existing Lustre client or Lustre server configuration while creating another.

- To create only a oneSIS image for a Lustre client:

```
# onesis_setup --rootfs=/var/lib/oneSIS/image/rhel5.3-onesis \
  -c /usr/share/oneSIS/includes/sysimage.conf.rhel5.3

# sunhpc_setup --diskless --netif=eth0 \
  --profile=rhel5.3-onesis  --skip-onesis-rootfs \
  --onesis-no-lustre
```

- To keep an existing oneSIS image for a Lustre client and create a oneSIS image for a Lustre server:

```
# onesis_lustre_rootfs \
  /var/lib/oneSIS/image/centos5.3-onesis \
  /var/lib/oneSIS/image/centos5.3-onesis-lustre

# sunhpc_setup --diskless --netif=eth0 \
  --profile=centos5.3-onesis --distro=centos5.3-onesis \
  --skip-onesis-rootfs
```

- *If* ***`perfctr`*** ***clients using a RHEL operating system*** are to be provisioned:

  a. Enter a command similar to the following, where the head node connects to the client nodes on Ethernet interface `eth1`:

```
# sunhpc_setup --profile=rhel5.3-perfctr \
  --distro-image=/mnt/rhel5.3 --sunhpc-repo=/media/sun_hpc_linux \
  --netif=eth1 --extras-repo
```

  The `--extras-repo` option enables the repository to install the `perfctr` patched kernel and `perfctr` user library.

  b. Generate a cobbler profile report to check that the cobbler profile `rhel5.3-perfctr` has been created:

```
# cobbler profile list
rhel5.3-perfctr
```

Preparing a head node running SLES 10 SP2

To set up a Cobbler profile on a head node running SLES 10 SP2, follow the procedure below. The examples assume that the head node has two network interfaces: `eth0` connects to the Internet or public network; `eth1` connects to the rest of the HPC cluster nodes and serves as a DHCP interface. The Cobbler profile is used to provision the compute cluster.

1. Check that the SLES 10 SP2 ISO, SLES 10 SP2 SDK ISO, and Sun HPC Software ISO are all mounted on the head node. The output of the `mount` command should contain the snippets below:

```
# mount
--snip--
/root/2.0/iso/SLE-10-SP2-SDK-DVD-x86_64-GM-DVD2.iso on \
   /media/sles10sdk type iso9660 (rw,loop=/dev/loop1)
/root/iso/SLES-10-SP2-DVD-x86_64-GM-DVD1.iso on /media/sles10sp2 \
   type iso9660 (rw,loop=/dev/loop0)
/root/iso/sun-hpc-linux-sles-trunk-beta2.iso on \
   /media/sun_hpc_linux  type iso9660 (rw,loop=/dev/loop2)
--snip--
```

2. To provision the Cobbler repository, complete one or both of the options below for diskful and diskless clients, as appropriate for your HPC cluster.

   - ***If diskful clients using a SLES operating system*** are to be provisioned:

     a. Enter a command similar to the following, where the head node connects to the client nodes on Ethernet interface `eth1`:

```
# sunhpc_setup --profile=sles10sp2 --distro-image=/mnt/sles10 \
  --sdk-repo=/mnt/sles10_sdk \
  --sunhpc-repo=/media/sun_hpc_linux_sles --netif=eth1
```

Output similar to the following will be displayed.

```
Initializing Cobbler configuration... Done
Restarting dhcpd/cobblerd/httpd... Done
Copying /mnt/sles to /var/www/cobbler/ks_mirror/sles10.2... Done
Created 'sunhpc_base_sles10.2' repo and copying... Done
Created 'sunhpc_lustre_sles10.2' repo and copying... Done
Created 'sunhpc_updates_sles10.2' repo and copying... Done
Copying repo sunhpc_base_sles10.2 to sunhpc_base_sles10.2_yast...
Done
Converting comps.xml to pattern... Done
Copying repo sunhpc_lustre_sles10.2 to
sunhpc_lustre_sles10.2_yast... Done
Converting comps.xml to pattern... Done
Copying repo sunhpc_updates_sles10.2 to
sunhpc_updates_sles10.2_yast... Done
Converting comps.xml to pattern... Done
Creating distro 'sles10.2' in cobbler... Done
Creating profile 'sles10.2' in cobbler... Done
Creating profile 'sles10.2-lustre' in cobbler... Done
```

b. Generate a Cobbler profile report to check that the Cobbler profiles `sles10.2` and `sles10.2-lustre` have been created. `sles10.2` is the profile for diskful Lustre client nodes. `sles10.2-lustre` is the profile for diskful Lustre server nodes, which will run on a Lustre patched kernel.

```
# cobbler profile list
sles10.2
sles10.2-lustre
```

- ***If diskless clients using a SLES operating system*** are to be provisioned:

a. Enter a command similar to the following, where the head node connects to the client nodes on Ethernet interface `eth1`:

```
# sunhpc_setup --profile=sles10sp2-onesis --diskless \
  --netif=eth1
```

b. Generate a Cobbler profile report to check that the Cobbler profiles `sles10.2_onesis` and `sles10.2_onesis-lustre` have been created. `sles10.2_onesis` is the profile for diskless Lustre client nodes. `sles10.2_onesis-lustre` is the profile for diskless Lustre server nodes, which will run on a Lustre patched kernel.

```
# cobbler profile list
sles10.2_onesis
sles10.2_onesis-lustre
```

Note: The procedure of creating a diskless image can be broken into several manual steps. This may be useful when you need to preserve an existing Lustre client or Lustre server configuration while creating another.

- To create only a oneSIS image for a Lustre client:

```
# onesis_setup --rootfs=/var/lib/oneSIS/image/sles10sp2-onesis \
  -c /usr/share/oneSIS/includes/sysimage.conf.sles10sp2
# sunhpc_setup --diskless –netif=eth0 --profile=sles10sp2-onesis \
  --skip-onesis-rootfs --onesis-no-lustre
```

- To keep an existing oneSIS image for a Lustre client and create a oneSIS image for a Lustre server:

```
# onesis_lustre_rootfs /var/lib/oneSIS/image/sles10sp2-onesis \
  /var/lib/oneSIS/image/sles10sp2-onesis-lustre
# sunhpc_setup --diskless --netif=eth0 \
  --profile=sles10sp2-onesis --distro=sles10sp2-onesis \
  --skip-onesis-rootfs
```

Step C. Prepare to Provision the Client Nodes

Overview

The Sun HPC Software manages the client node provisioning process using the Sun HPC Software Management Database (`gtdb`) provided with the Sun HPC Software. To provision the client nodes in the compute cluster, you will first populate `gtdb` using the Sun HPC Software Management Tools (`gtt`). You will then generate configuration files for provisioning tools, such as Cobbler, which will be used to provision each node in the cluster from the head node. See Appendix A for a description of the types of client nodes for which the Sun HPC Software provides provisioning support.

Introduction to the Sun HPC Software Management Database and Tools

The Sun HPC Management Tools (`gtt`) support two primary functions: adding, editing, and deleting information in the Sun HPC Management Database (`gtdb`) and generating configuration files from the database for use by Sun HPC Software components.

The Sun HPC Software Management Database is a SQLite database running under Ruby on Rails used to manage the configuration of an HPC cluster. After populating the database with information about the HPC cluster (such as hostnames, and network addresses) using the Sun HPC Software Management Tools (`gtt`), a cluster administrator can then generate configuration files for supported services (such as ConMan, PowerMan, or SLURM) and system databases (such as `/etc/hosts` or `/etc/genders`).

Adding data to the Sun HPC Software Management Database

Two methods are provided to manage the content of the `gtdb` database. One method is to use the `gtt host` command to directly add, edit, or delete information in the management database. The `gtt host` command can be used to:

- Add, edit, or remove a host
- Add, change, or remove an attribute
- Add or remove a network

The second method is to use the bulk import function to import data from a file. The bulk import file format has the concept of classes allowing you to assign attributes to a class. All nodes in that class will then inherit those attributes.

Note: In the 2.0 release, to assign attributes to hosts, you must first define classes and then assign hosts to those classes. The bulk importer is unable to make attributes specific to a node.

Generating configuration files from the Sun HPC Software Management Database

The `gtt config` command is used to generate common configuration files used by the Sun HPC Software components from the database rather than requiring them to be edited by hand. A configuration file for a single service (such as SLURM) can be generated or updated using a command similar to:

```
gtt config --update slurm
```

All supported configuration files can be generated or updated using:

```
gtt config --update all
```

Configuration files for Cfengine, Cobbler, ConMan, Genders, host file, ntp, PowerMan and SLURM are generated automatically.

Note: Cobbler and SLURM each require that an attribute be set to allow their configuration files to be modified. For Cobbler, set `eth0_bootnet=true` and for SLURM, set `slurm_partition=compute`.

Only a portion of the configuration files are managed with the Sun HPC Software Management Database and Tools. The managed section is marked as shown below. To make changes to this section in a configuration file, use the `gtt host` command to edit the database and then regenerate the configuration file with `gtt config`. Everything outside of this block is safe to edit and manage directly.

```
######### BEGIN GTDB MANAGEMENT -- DO NOT EDIT BELOW THIS LINE #############

192.168.202.253           cl10-0
192.168.201.129           cl10-0-sp

192.168.202.254           cl10-1
192.168.201.130           cl10-1-sp
########## END GTDB MANAGEMENT -- DO NOT EDIT ABOVE THIS LINE ##############
```

The `gtt settings` command can be used to show information about a specific service. For example, to see information about SLURM, enter:

```
gtt settings --show --service slurm
```

The `gtt help` command provides more detailed information about using the `gtt` command.

```
# gtt help
Usage:
  gtt -h/--help
  gtt -v/--version
  gtt command [arguments] [options]

Examples:
  gtt host --show --name compute[0-10]

  gtt config --update all

Further help:
  gtt help commands           List all 'gtt' commands
  gtt help examples           Show some examples of usage
  gtt help <COMMAND>          Show help on COMMAND
                               (e.g. 'gtt help host')
```

To find out what options are available for adding hosts, run the `gtt help host` command.

```
# gtt help host
Actions:
       --add                      Add a host
       --edit                     Edit a host
       --remove                   Remove a host or hostlist
       --addnet                   Add a network to an existing host
       --removenet                Remove a network from a host
       --addattr                  Add an attribute to host or hostlist
       --changeattr               Change an attribute for a host or hostlist
       --removeattr               Remove an attribute from host or hostlist
       --show                     Show details for host or hostlist
Options:
       --name [hostname]          Hostname or hostlist
       --network [network]        Network string or device
       --attribute [attribute]    Attribute string or name
       --fields [fields]          Host fields to update
General Info:
    -v, --version                    Show the version number and quit.
    -h, --help                       Show this help message and quit.
Examples:
    /usr/bin/gtt host --add --name host00 \
       --network
"hwaddr=00:01:02:03:04:05,ipaddr=192.168.1.1,device=eth0,bootnet=true" \
       --attribute "mds" --attribute "fsname=work"

    /usr/bin/gtt host --edit --name host00 --fields "primary_interface=ib0"

    /usr/bin/gtt host --remove --name host00
    /usr/bin/gtt host --remove --name compute[23,34,100-128]

    /usr/bin/gtt host --addattr --name host00 --attribute "smell=funky"
    /usr/bin/gtt host --addattr --name oss[00-32] --attribute "oss" \
       --attribute "fsname=work"

    /usr/bin/gtt host --changeattr --name host00 --attribute "smell=strange"
```

```
    /usr/bin/gtt host --removeattr --name oss[01,05-07,23] \
       --attribute "fsname=work"

    /usr/bin/gtt host --show --name oss[00-02,06]
```

To display a list of tools managed by `gtdb`, enter:

```
# gtt settings --show --service system --component configs
```

A list similar to the following will be displayed:

```
system:configs = cfagent cfservd cfupdate cobbler conman genders hosts ntp
```

Preparing to provision the Client nodes

Follow the procedure below to populate the Sun HPC Software Management Database (`gtdb`) and generate configuration files for provisioning.

1. If you have not done so already, create an inventory of nodes in your HPC cluster (see Appendix A for an example of a cluster inventory).
2. Populate the `gtdb` database. Enter HPC cluster configuration information into `gtdb` using one of the two options below while referring to your cluster inventory as needed. Option 1 describes how to use the `gtt` command to create or edit entries in the database. Option 2 provides a procedure for importing a text file containing node information into the database.

   Option 1 – Use the `gtt` command to add hosts to or edit hosts in `gtdb`.

   Several examples are provided below to show how the `gtt` command can be used to add or edit hosts in `gtdb`.

   The first example below shows how to add a diskful host `cl10-9` that will be running a RHEL operating system. The profiles `rhel5.3-lustre` and `rhel5.3-onesis` were created previously (see Step B. Prepare the Head Node to Provision the Cluster). The option `--attribute "profile=rhel5.3-lustre"` can be changed to the name of another previously created profile if needed. The profile specified by the `profile=` option must match a profile in Cobbler that was created when `sunhpc_setup –-profile` was run. To view the existing cobbler profiles, enter `cobbler list`.

```
# gtt host --add --name cl10-9 \
--network "hwaddr=00:23:8B:03:C6:DA,ipaddr=192.168.202.243,\
  device=eth0,bootnet=true" \
--network "hwaddr=00:23:8B:03:C8:70,ipaddr=192.168.201.138,\
  device=sp,module=sun-ilom" \
--attribute "profile=rhel5.3-lustre" --attribute static

# gtt host --add --name cl10-9 \
```

```
--network "hwaddr=00:23:8B:03:C6:DA,ipaddr=192.168.202.243,\
  device=eth0,bootnet=true" \
--network "hwaddr=00:23:8B:03:C8:70,ipaddr=192.168.201.138,\
  device=sp,module=sun-ilom" \
--attribute "profile=rhel5.3-onesis" --attribute static
```

The second example shows how to add several diskless hosts that will be running a SLES 10 SP2 operating system. The example assumes that the profiles `sles10.2_onesis` and `sles10.2_onesis-lustre` were created previously (see Step B. Prepare the Head Node to Provision the Cluster).

```
gtt host --add --name cl10-5 \
--network "hwaddr=00:14:4F:F7:2E:D0,ipaddr=192.168.202.249,\
  device=eth0,bootnet=true" \
--network "hwaddr=00:21:28:14:B9:61,ipaddr=192.168.201.134,\
  device=sp,module=sun-ilom" \
--attribute "profile=sles10.2_onesis" --attribute static

gtt host --add --name cl10-9 \
--network "hwaddr=00:23:8B:03:C6:DA,ipaddr=192.168.202.243,\
  device=eth0,bootnet=true" \
--network "hwaddr=00:23:8B:03:C8:70,ipaddr=192.168.201.138,\
  device=sp,module=sun-ilom" \
--attribute "profile=sles10.2_onesis-lustre" --attribute static
```

In the example above, only one network interface is used for provisioning, designated by `bootnet=true`. Other networks can be added by including additional `--network` options.

This example includes a `--network` option in which service processor information is provided (`device=sp, module=sun-ilom`). When this information is included, a ConMan configuration file is automatically generated. See the section Configuring the ConMan Console Management Tool for how to configure and use ConMan.

The option `--attribute static` enables clients to be provided with a static IP address after provisioning. Without this attribute, the clients will be provided with a dynamic IP address allocated by the DHPC server running on the head node.

Option 2 – Import a text file to create host entries in `gtdb`.

For clusters with many hosts, running individual `host add` commands is neither convenient nor efficient. In the case of a cluster containing hundreds or thousands of hosts, the preferred option is to define a cluster import file. In addition to the improved speed and ease-of-use, the cluster import feature provides atomicity to the import. If one host fails to import, the entire transaction is rolled back, so that the administrator can easily fix the problem and start over again.

Cluster administrators have complete control over how hostnames and IP addresses will be assigned to the imported hosts. If desired, the import file can specify a hostname and IP address for each host entry, or the import file can define templates for hostname and IP networks. Each host matching a class will then be assigned a hostname and IP address according to its position in the import file (for example, `host00`, `host01`, `host02`) for comments.

The bulk importer skips over lines that start with # indicating comments.

Note: Use `gtt import help` to get more information about `gtt import` including access to an example `import.txt` file in which each field is explained.

Complete the steps below:

a. Create a text file `import.txt.`

- In this example, the same ten nodes are used as in Option 1 above, with administrator-specified hostnames and network addresses. Three host classes are defined: the mandatory `default` class, the `sles10_onesis` class, and the `sles10_onesis_lustre` class.

Note: The class only exists while the import is in process. The class itself is not stored in the database.

```
# cat import.txt
log_host_creation: true

default: eth0_network=192.168.202.0/24; eth0_bootnet=true; \
sp_network=192.168.201.0/24; sp_module=sun-ilom
sles10_onesis: attributes=profile=sles10.2_onesis
sles10_onesis_lustre: attributes=profile=sles10.2_onesis_lustre

name=cl10-5; class=sles10_onesis; \
eth0=00:14:4F:F7:2E:D0; eth0_ipaddr=192.168.202.249; \
sp=00:21:28:14:B9:61; sp_ipaddr=192.168.201.134

name=cl10-6; class=sles10_onesis; \
eth0=00:14:4F:F7:36:36; eth0_ipaddr=192.168.202.248; \
sp=00:21:28:14:BC:31; sp_ipaddr=192.168.201.135

name=cl10-7; class=sles10_onesis; \
eth0=00:1E:68:2E:EF:F2; eth0_ipaddr=192.168.202.247; \
sp=00:1E:68:EE:F8:96; sp_ipaddr=192.168.201.136

name=cl10-8; class=sles10_onesis_lustre; \
eth0=00:23:8B:03:C6:DC; eth0_ipaddr=192.168.202.246; \
sp=00:23:8B:03:C8:79; sp_ipaddr=192.168.201.137

name=cl10-9; class=sles10_onesis_lustre; \
```

```
eth0=00:23:8B:03:C6:DA; eth0_ipaddr=192.168.202.243; \
sp=00:23:8B:03:C8:70; sp_ipaddr=192.168.201.138
hpc-x4600-2:~ #
```

- This example shows the addition of an attribute for SLURM to allow the database to generate a `slurm.conf` file automatically:

```
log_host_creation: true

default: eth0_network=192.168.202.0/24; eth0_bootnet=true; \
         sp_network=192.168.201.0/24; sp_module=sun-ilom
sles10_onesis: attributes=profile=sles10.2,
slurm_partition=compute
sles10_onesis_lustre: attributes=profile=sles10.2-lustre

name=cl10-5; class=sles10_onesis; \
  eth0=00:14:4F:F7:2E:D0; eth0_ipaddr=192.168.202.249; \
  sp=00:21:28:14:B9:61; sp_ipaddr=192.168.201.134

name=cl10-6; class=sles10_onesis_lustre; \
  eth0=00:14:4F:F7:36:36; eth0_ipaddr=192.168.202.248; \
  sp=00:21:28:14:BC:31; sp_ipaddr=192.168.201.135

name=cl10-7; class=sles10_onesis; \
  eth0=00:1E:68:2E:EF:F2; eth0_ipaddr=192.168.202.247; \
  sp=00:1E:68:EE:F8:96; sp_ipaddr=192.168.201.136

name=cl10-8; class=sles10_onesis; \
  eth0=00:23:8B:03:C6:DC; eth0_ipaddr=192.168.202.246; \
  sp=00:23:8B:03:C8:79; sp_ipaddr=192.168.201.137

name=cl10-9; class=sles10_onesis; \
  eth0=00:23:8B:03:C6:DA; eth0_ipaddr=192.168.202.243; \
  sp=00:23:8B:03:C8:70; sp_ipaddr=192.168.201.138
```

b. To add the file to `gtdb`, using one of the options below:

- Import a text file using the `gtt` command:

```
# gtt import -f import.txt
```

You will see a result similar to the following:

```
Import executed successfully.
<========== Import Options ==========>
host_counter_starts_at: 0
import_is_atomic: true
log_host_creation: true
max_errors: 10
skip_duplicate_hosts: false

<========== Host Classes ==========>
```

```
class: Default
networks:
eth0: 192.168.202.0/24, bootnet = true
sp: 192.168.201.0/24, module = sun-ilom
attributes:
class: Sles10Onesis
networks:
eth0: 192.168.202.0/24, bootnet = true
sp: 192.168.201.0/24, module = sun-ilom
attributes:
profile=sles10.2_onesis
class: Sles10OnesisLustre
networks:
eth0: 192.168.202.0/24, bootnet = true
sp: 192.168.201.0/24, module = sun-ilom
attributes:
profile=sles10.2_onesis_lustre

<========== Host Imports ==========>
host: cl10-5
network: eth0 00:14:4F:F7:2E:D0 192.168.202.249
network: sp 00:21:28:14:B9:61 192.168.201.134
attribute: profile=sles10.2_onesis
host: cl10-6
network: eth0 00:14:4F:F7:36:36 192.168.202.248
network: sp 00:21:28:14:BC:31 192.168.201.135
attribute: profile=sles10.2_onesis
host: cl10-7
network: eth0 00:1E:68:2E:EF:F2 192.168.202.247
network: sp 00:1E:68:EE:F8:96 192.168.201.136
attribute: profile=sles10.2_onesis
host: cl10-8
network: eth0 00:23:8B:03:C6:DC 192.168.202.246
network: sp 00:23:8B:03:C8:79 192.168.201.137
attribute: profile=sles10.2_onesis_lustre
host: cl10-9
network: eth0 00:23:8B:03:C6:DA 192.168.202.243
network: sp 00:23:8B:03:C8:70 192.168.201.138
attribute: profile=sles10.2_onesis_lustre

<========== Summary ==========>
5 hosts imported.
```

- Use the simpler import shown below, which assigns hostnames and IP addresses according to position in the import:

```
# cat import.txt

log_host_creation: true

#Host Classes
default: name=cl10-%d; eth0_network=192.168.202.0/24;
eth0_bootnet=true; \

sp_network=192.168.201.0/24; sp_module=sun-ilom
sles10_onesis: attributes=profile=sles10.2_onesis

sles10_onesis_lustre: attributes=profile=sles10.2_onesis_lustre
```

```
# Host Entries
class=sles10_onesis; eth0=00:14:4F:F7:2E:D0; sp=00:21:28:14:B9:61

class=sles10_onesis; eth0=00:14:4F:F7:36:36; sp=00:21:28:14:BC:31

class=sles10_onesis; eth0=00:1E:68:2E:EF:F2; sp=00:1E:68:EE:F8:96

class=sles10_onesis_lustre; eth0=00:23:8B:03:C6:DC;
sp=00:23:8B:03:C8:79

class=sles10_onesis_lustre; eth0=00:23:8B:03:C6:DA;
sp=00:23:8B:03:C8:70
```

You will see a result similar to the following:

```
Import executed successfully.

<========== Import Options ==========>
host_counter_starts_at: 1
import_is_atomic: true
log_host_creation: true
max_errors: 10
skip_duplicate_hosts: false

<========== Host Classes ==========>
class: Default
networks:
eth0: 192.168.202.0/24, bootnet = true
sp: 192.168.201.0/24, module = sun-ilom
attributes:
class: Sles10Onesis
networks:
eth0: 192.168.202.0/24, bootnet = true
sp: 192.168.201.0/24, module = sun-ilom
attributes:
profile=sles10.2_onesis
class: Sles10OnesisLustre
networks:
eth0: 192.168.202.0/24, bootnet = true
sp: 192.168.201.0/24, module = sun-ilom
attributes:
profile=sles10.2_onesis_lustre

<========== Host Imports ==========>
host: cl10-1
network: eth0 00:14:4F:F7:2E:D0 192.168.202.1
network: sp 00:21:28:14:B9:61 192.168.201.1
```

```
attribute: profile=sles10.2_onesis
host: cl10-2
network: eth0 00:14:4F:F7:36:36 192.168.202.2
network: sp 00:21:28:14:BC:31 192.168.201.2
attribute: profile=sles10.2_onesis
host: cl10-3
network: eth0 00:1E:68:2E:EF:F2 192.168.202.3
network: sp 00:1E:68:EE:F8:96 192.168.201.3
attribute: profile=sles10.2_onesis
host: cl10-4
network: eth0 00:23:8B:03:C6:DC 192.168.202.4
network: sp 00:23:8B:03:C8:79 192.168.201.4
attribute: profile=sles10.2_onesis_lustre
host: cl10-5
network: eth0 00:23:8B:03:C6:DA 192.168.202.5
network: sp 00:23:8B:03:C8:70 192.168.201.5
attribute: profile=sles10.2_onesis_lustre

<========== Summary ==========>
5 hosts imported.
```

3. Generate a set of configuration files from `gtdb`.

```
# gtt config --update all
```

A list of configuration files with their updates will be displayed. For example:

```
Updating config: cfservd
/var/lib/sunhpc/cfengine/var/cfengine/inputs/cfservd.conf: Wrote 35
lines
Updating config: cfupdate
/var/lib/sunhpc/cfengine/var/cfengine/inputs/update.conf: Wrote 68 lines
Updating config: cobbler
/var/lib/sunhpc/cfengine/tmp/cobbler.csv: Wrote 5 lines
Updating config: conman
/var/lib/sunhpc/cfengine/etc/conman.conf: Wrote 183 lines
Updating config: genders
/var/lib/sunhpc/cfengine/etc/genders: Wrote 6 lines
Updating config: hosts
/var/lib/sunhpc/cfengine/etc/hosts: Wrote 15 lines
Updating config: ntp
/var/lib/sunhpc/cfengine/etc/ntp.conf: Wrote 24 lines
Updating config: powerman
/var/lib/sunhpc/cfengine/etc/powerman/powerman.conf: Wrote 7 lines
Updating config: slurm
/var/lib/sunhpc/cfengine/etc/slurm/slurm.conf: Wrote 37 lines
```

4. Update the local configuration files on the head node by running `cfagent`, which will copy files from `/var/lib/sunhpc/cfengine` into the appropriate places.

```
# cfagent -q
```

5. Generate data for Cobbler from the Cobbler configuration file `cobbler.csv` by entering:

```
# populate_cobbler_system /var/lib/sunhpc/cfengine/tmp/cobbler.csv

Internet Systems Consortium DHCP Server V3.0.5-RedHat
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw
Shutting down dhcpd:
Starting dhcpd:                              done
```

6. Use `cobbler list` to get a summary of clients (referred to as the "system" in Cobbler) and client profiles.

```
# cobbler list
distro rhel5.3
   profile rhel5.3
      system cl10-0
   profile rhel5.3-lustre
      system cl10-1
distro rhel5.3-onesis
   profile rhel5.3-onesis
      system cl10-4
distro rhel5.3-onesis-lustre
   profile rhel5.3-onesis-lustre
      system cl10-2
repo sunhpc_base_rhel5.3
repo sunhpc_extras_rhel5.3
repo sunhpc_lustre_rhel5.3root
repo sunhpc_updates_rhel5.3
```

You are now ready to boot the client compute nodes.

Step D. Provision the Client Nodes

Follow the procedure below to provision the client nodes in your cluster.

1. Verify the node configuration by generating a report and comparing the contents to your cluster inventory.

```
# cobbler system report
```

Add the option `--name=[client name]` to narrow the scope of the report if necessary. For example:

```
# cobbler system report --name=cl10-6
system                  : cl10-6
profile                 : sles10.2
comment                 :
created                 : Wed May 6 05:45:00 2009
gateway                 :
hostname                : cl10-6
image                   :
kernel options          : {'ksdevice': 'eth0'}
kernel options post     : {}
kickstart               : <<inherit>>
ks metadata             : {}
mgmt classes            : []
modified                : Wed May 6 06:43:57 2009
name servers            :
netboot enabled?        : True
owners                  : ['admin']
server                  : <<inherit>>
template files          : {}
virt cpus               : <<inherit>>
virt file size          : <<inherit>>
virt path               : <<inherit>>
virt ram                : <<inherit>>
virt type               : <<inherit>>
power type              : ipmitool
power address           :
power user              :
power password          :
power id                :
interface          : eth0
  mac address      : 00:14:4F:F7:36:36
  bonding          :
  bonding_master   :
  bonding_opts     :
  is static?       : True
  ip address       : 192.168.202.248
  subnet           : 255.255.255.0
  static routes    : []
  dns name         : cl10-6
  dhcp tag         :
  virt bridge      :
```

2. Reboot the clients over the network from the head node (the head node must have access to the client node management network interface):

```
# ipmi-chassis-config -h [client node name or IP for ILOM] -u root \
  -p [Root password] -e "Chassis_Boot_Flags:Boot_Device=PXE" –commit
# ipmipower -h [client node name or IP for ILOM] -u root \
  -p [Root password] –reset
```

For example:

```
# ipmi-chassis-config -h cl10-[0-9]-sp -u root \
  -p changeme -e "Chassis_Boot_Flags:Boot_Device=PXE" --commit
# ipmipower -h cl10-[0-9]-sp -p changeme --reset
```

Note: On older hardware, such as the Sun Fire V20z Server and the Sun Fire V40z Server, the `-u` option must be omitted from the `ipmi-chassis-config` command.

After the clients reboot, the provisioning process will start. If the head node is running a RHEL operating system, you can use `cobbler status` to check the progress of the provisioning process.

```
# cobbler status
ip             |target          |start                   |stat
192.168.202.248|system:cl10-6   |Wed May  6 06:47:33 2009|finished
192.168.202.251|system:cl10-3   |Wed May  6 06:47:27 2009|finished
```

3. Once the client provisioning completes, run the following commands to test password-less `ssh` access to the provisioned clients and add them to `.ssh/known_hosts`.

```
# PDSH_SSH_ARGS_APPEND="-o StrictHostKeyChecking=no" pdsh -g \
  profile hostname
```

Warning messages similar to the following are displayed to indicate the clients have been added to the `known_hosts` list:

```
Warning: Permanently added 'cl10-0,192.168.202.253' (RSA) to the list of
known hosts.
```

4. Run a simple `pdsh` command to check if all the provisioned clients are accessible. A typical result is:

```
[root@hpc-x4600-1 ~]# pdsh -g profile uptime
cl10-2:  14:25:36 up  2:45,  0 users,  load average: 0.02, 0.02, 0.00
cl10-1:  13:25:52 up  2:44,  0 users,  load average: 0.09, 0.03, 0.00
cl10-4:  14:25:59 up  2:45,  0 users,  load average: 0.01, 0.01, 0.00
cl10-0:  19:25:39 up  1:49,  2 users,  load average: 0.00, 0.00, 0.08
cl10-3:  18:25:49 up  1:55,  0 users,  load average: 0.00, 0.00, 0.04
```

Configuring a Serial Console (Optional)

A serial console is often used to remotely manage an HPC cluster. By default, the `sunhpc_setup` script creates a serial console set by default to `ttyS0,9600` or to another serial console configuration if the option `--bootparams` is used. If the serial console has been configured to the wrong port, output will not be directed to the console. You can edit the serial console configuration at profile and system level through Cobbler.

To edit the Cobbler profile (in this example, `rhel5.3cob`) to change the serial console configuration, enter:

```
# cobbler profile edit --name=rhel5.3 --kopts="console=ttyS1,9600"
# cobbler sync
```

To edit the serial console configuration at the system level for a client node (`cl10-1` in the example), enter:

```
# cobbler system edit --name=cl10-1 --kopts="console=ttyS1,9600"
# cobbler sync
```

Configuring the Lustre File System (Optional)

Once the client nodes have been provisioned, they can serve as Lustre server nodes or Lustre client nodes regardless of whether they are diskful or diskless. To configure the Lustre file system, follow the configuration procedure in the Lustre documentation at http://wiki.lustre.org/index.php?title=Mount_Conf.

For detailed information about configuring the Lustre File system, refer to the Lustre wiki or attend a Lustre training provided by the Sun Training. For more information, go to:

- http://wiki.lustre.org/index.php/Main_Page
- http://www.sun.com/training/catalog/courses/CL-400.xml

Chapter 4:
Managing the HPC Cluster

The Sun HPC Software, Linux Edition 2.0 includes several commonly used tools for managing an HPC cluster including PowerMan, ConMan, pdsh, Cfengine, and Cobbler. This section describes these tools and also includes a procedure for setting up Secure Shell (`ssh`) public key authentication for several Sun HPC Software components.

Setting Up SSH Keys

Some Sun HPC Software components, such as `pdsh` and `pdcp`, require Secure Shell (`ssh`) public key authentication to access clients in the cluster. The Sun HPC Software automatically creates `ssh` keys for `root` and distributes them to all diskful and diskless provisioned nodes.

If you need to add keys for another user, change a key, or give a key to a non-provisioned node, this section describes how to do a basic `ssh` key setup and use `ssh` keys. For more information, see the `ssh` man page in your Linux distribution

Creating SSH keys

Follow the steps below to set up `ssh` keys.

1. Create the ssh public key on the head node.

```
ssh-keygen -t <Specify type of key to create> -b <number of bits> -N
<can be used to give passphrase>
```

   For example:

```
# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
15:ee:73:c3:6e:8f:a8:92:86:84:01:cc:50:f3:24:50 root@hpc-x4600-2
```

2. Copy the public key to the client nodes.

   - *For **diskless clients***, use a command sequence similar to the following.

```
# pwd
/root/.ssh
# ls
id_rsa id_rsa.pub known_hosts
```

   Verify `/root.ssh` exists with correct permissions

```
# ls -lart /var/lib/oneSIS/image/sles10.2_onesis/root/.ssh/
drwx------ 2 root root 4096 Apr 23 18:22 .
# cat id_rsa.pub >>
/var/lib/oneSIS/image/sles10.2_onesis/root/.ssh/authorized_keys
```

   You may need to create a directory or modify permissions. For example:

```
# mkdir -p /var/lib/oneSIS/image/xxx/root/.ssh
# chmod 700 /var/lib/oneSIS/image/xxx/root/.ssh
```

   - *For **diskful clients***, copy the key at provisioning time or copy it to each host after the systems have been provisioned using the `scp` command (secure copy) or a similar tool.

3. Test access to the clients. In this example, `pdsh` is used with a password-less `ssh` key to access cl10-6 and cl10-7 to run the `uptime` command.

```
# pdsh -w cl10-[6-7] "uptime"
cl10-7: 6:27pm up 5:48, 0 users, load average: 0.00, 0.00, 0.00
cl10-6: 6:27pm up 5:48, 0 users, load average: 0.00, 0.00, 0.00
```

4. If your site has security policies that prevent the use of a null key, use `ssh-agent` to store the `passphrase` in memory so that you do not need to enter it each time a host is accessed. This procedure can be scripted to run at login time.

```
ssh-agent -s > file
source file
ssh-add
```

   You can use `ssh-add -l` to list the fingerprints of all identities currently represented by the authentication agent.

Generating SSH keys for hosts defined in a Genders configuration file

To generate keys for host entries in an `/etc/genders` file, complete these steps.

1. Define the nodes in the Sun HPC Software Management Database `gtdb` (see Appendix A for an example of a cluster inventory).
2. Run the `gtt config` command to create a Genders configuration file:

```
# gtt config --update genders
Updating config: genders
/var/lib/sunhpc/cfengine/etc/genders: Wrote 7 lines
```

3. Verify the content of the `genders` file generated from the database.

```
cat /var/lib/sunhpc/cfengine/etc/genders

#### BEGIN GTDB MANAGEMENT -- DO NOT EDIT BELOW THIS LINE ###########
cl10-5              profile=sles10.2_onesis
cl10-6              profile=sles10.2_onesis
cl10-7              profile=sles10.2_onesis
cl10-8              profile=sles10.2_onesis_lustre
cl10-9              profile=sles10.2_onesis_lustre
########## END GTDB MANAGEMENT #####################
```

4. Use `cfagent` to update the generated configuration to `/etc/genders`.

```
# cfagent -q
```

5. Verify the contents of `/etc/genders`.

```
# cat /etc/genders
##### BEGIN GTDB MANAGEMENT -- DO NOT EDIT BELOW THIS LINE #############
cl10-5              profile=sles10.2_onesis
cl10-6              profile=sles10.2_onesis
cl10-7              profile=sles10.2_onesis
cl10-8              profile=sles10.2_onesis_lustre
cl10-9              profile=sles10.2_onesis_lustre
##################### END GTDB MANAGEMENT ##############################
```

Note: These nodes must either be in `/etc/hosts` or must be able to be resolved through DNS in order for Genders to work.

Configuring the PowerMan Power Management Tool

PowerMan is a centralized power management tool capable of handling a large number of machines. As shipped with the Sun HPC Software, PowerMan supports the Sun Integrated Lights Out Manager (ILOM).

Note: To configure ILOM, refer to inventory of your HPC cluster devices with their MAC and IP addresses used for the installation and provisioning procedures in Chapter 2 (see Appendix A for an example of a cluster inventory).

To configure ILOM in the PowerMan power management tool and run PowerMan, complete the following steps.

1. Ensure that the hosts imported into the Sun HPC Software Management Database `gtdb` have an associated `sp` network entry. For example:

```
# gtt host --add --name host00 \
  --network "device=eth0,hwaddr=00:01:02:03:04:05,\
  ipaddr=192.168.1.1" --network "device=sp,\
  hwaddr=01:02:03:04:05:06,ipaddr=172.168.1.1,\module=ilom"
```

2. Generate a new powerman.conf from the imported ILOM entries:

```
# gtt config --update powerman
Updating config: powerman
/var/lib/sunhpc/cfengine/etc/powerman/powerman.conf: Wrote 5 lines
```

3. Use `cfagent` to update the generated configuration `/etc/powerman/powerman.conf`:

```
# cfagent -q
```

4. Start powerman.

```
# /etc/init.d/powerman start
```

Note: Before using PowerMan for the first time, edit `/etc/ipmipower.conf` to enter appropriate values for username and password. For Sun ILOM service processors, the default username is `root` and the default password is `changeme`. After setting these values, restart the `powerman` service.

You can use PowerMan to power on, power off, and power cycle machines as shown in the examples below:

```
# pm -q host[00-12]
on: host[00-05,07,11-12]
off: host[06,08-10]
unknown:
```

```
# pm --on host06
```

```
# pm --off host[11-12]
```

```
# pm --cycle host[00,02]
```

```
# pm --on -g "fsname=lustre00"
```

Configuring the ConMan Console Management Tool

ConMan is a centralized console management tool capable of handling a large number of machines. As shipped with the Sun HPC Software, ConMan supports the Sun Integrated Lights Out Manager (ILOM).

To configure ILOM in the ConMan serial console management tool and run ConMan, complete the following steps. Refer to the list of the devices with their MAC and IP addresses created in Appendix A.

1. Ensure that the hosts imported into the Sun HPC Software Management Database `gtdb` have an associated `sp` network entry. For example:

```
# gtt host --add --name host00 --network
"device=eth0,hwaddr=00:01:02:03:04:05,ipaddr=192.168.1.1" \
 --network
"device=sp,hwaddr=01:02:03:04:05:06,ipaddr=172.168.1.1,module=sun-ilom"
```

2. Ensure the ConMan `username` is correct:

```
# gtt settings --show --service conman --component username
conman:username = root
```

   This shows the `username` is set to `root`. If this is incorrect (commonly seen on older Sun hardware), change it:

```
# gtt settings --edit --service conman --component username \
  --value admin
Updated setting: conman:username
```

3. Generate a new `conman.conf` file from the imported ILOM entries:

```
# gtt config --update conman
Updating config: conman
/var/lib/sunhpc/cfengine/etc/conman.conf: Wrote 184 lines
```

4. Use `cfagent` to update the generated configuration in `/etc/conman.conf`.

```
# cfagent -q
```

5. Edit the password file `/etc/conman.pswd` if needed. By default, the `sunhpc_configuration` RPM included with the Sun HPC Software supplies a conman password file that specifies a *host regex* that matches all hosts.

```
# cat /etc/conman.pswd
# /etc/conman.pswd
#
# This file is consulted by various expect scripts in \
   /usr/lib/conman/exec
# to find the password for a console if it is not specified on the
# command-line. As part of the Sun HPC Software stack, it is shipped \
   with
# the default credentials for Sun Service Processors.
#
```

```
# The format of each record is:
# host regex : username : password
#
.* : root : changeme
```

6. Start `conman`:

```
#/etc/init.d/conman start
```

7. Verify `conman` is now logging to the conman log file `/var/log/conman`. Example contents are shown below:

```
-rw-------  1 root root   1034 Apr 23 11:28 cl10-4.log
-rw-------  1 root root   3182 Apr 29 12:55 cl10-7.log
-rw-------  1 root root   2984 Apr 29 12:55 cl10-5.log
-rw-------  1 root root    198 Apr 29 12:55 cl10-0.log
-rw-------  1 root root   3182 Apr 29 12:55 cl10-9.log
-rw-------  1 root root    198 Apr 29 12:55 cl10-1.log
-rw-------  1 root root   3263 Apr 29 12:55 cl10-8.log
-rw-------  1 root root   1232 Apr 30 12:33 cl10-3.log
-rw-------  1 root root 902823 Apr 30 12:37 cl10-6.log
```

8. To access a specific console after conman has been configured and started, use the `conman` command:

```
# conman cl10-0
<ConMan> Connection to console [cl10-0] opened
```

Other commands that can be used to operate `conman` are shown below.

- To query for remote consoles that can be connected to by `conman`, use:

```
# conman -q
```

- To connect to the console on a server, use:

```
# conman [-j] [-f] nodename
```

  where:

  –`f` terminates sessions used by other users and forces a connection to this session.

  –`j` joins to a session in use by other users.

- To terminate a `conman` connection to a session, enter:

```
&.
```

  Nothing will be displayed in response to this command

Setting Up and Using pdsh

The pdsh (Parallel Distributed SHell) utility is used to perform simultaneous actions in parallel across multiple hosts in a cluster. As shipped with the Sun HPC Software, pdsh is configured to use ssh as its underlying transport and can utilize information in a Genders database for host selection. The pdsh utility can be used "out of the box" with no additional configuration to access fixed sets of hosts when the host lists are explicitly defined as arguments to pdsh. Adding free-form attributes to hosts in the Sun HPC Software database `gtdb` and then updating the Genders database allows for more flexible host selection.

To set up pdsh, complete the steps below.

1. Create host entries in `gtdb`. You can add arbitrary host attributes with values in the form of `key` or `key=value`.

```
# gtt host --addattr --name host00 --attribute smell=funny \
  --attribute smelly
# gtt host --addattr --name mds00 --attribute mds \
  --attribute fsname=lustre00
# gtt host --addattr --name oss00 --attribute oss \
  --attribute fsname=lustre00 --attribute ost00 --attribute ost01
# gtt host --addattr --name oss01 --attribute oss \
  --attribute fsname=lustre00 --attribute ost02 --attribute ost03
```

2. Update the Genders database

```
# gtt config --update genders
```

You can use pdsh to access hosts by `hostname` or Genders attributes as shown in the examples below.

```
# pdsh -w 'oss[00-01]' hostname
    oss00: oss00
    oss01: oss01

# pdsh -g 'fsname=lustre00&&ost02' \
   'cat /proc/fs/lustre/obdfilter/lustre00-ost0002/recovery_status'

# pdsh -g 'fsname=lustre00' 'cat /proc/fs/lustre/health_check' \
   | dshbak -c
----------------
oss[00-01]
----------------
healthy
----------------
mds00
----------------
NOT HEALTHY
```

Setting up Cfengine to Manage Configuration Files on Clients

Cfengine (http://www.cfengine.org) allows configuration files (and more) to be managed on a large number of nodes. The Sun HPC Software includes a minimal Cfengine configuration in which configuration files are copied from `/var/lib/sunhpc/cfengine/` on the head node to all cluster nodes.

Although configuration files can be distributed from the head node using either a push mode or a pull mode, the Cfengine configuration provided by the Sun HPC Software uses the pull mode. The Cfengine server daemon (`cfservd`) runs on the head node, while the program `cfagent` must be run on each client to update the client's configuration files. Clients can be updated regularly by, for example, using `cron` to run `cfagent`.

To update configuration files on a subset of all nodes, complete the following steps:

1. Identify the hostname(s) or profile of the node(s) to be updated.

   Sun HPC software defines node types in `/etc/genders`. Depending on the profile names chosen during `sunhpc_setup`, `/etc/genders` may look like this:

   ```
   cl10-0              profile=sles10.2
   cl10-1              profile=sles10.2
   cl10-2              profile=sles10.2
   cl10-3              profile=sles10.2_lustre
   cl10-4              profile=sles10.2_lustre
   cl10-5              profile=sles10.2_onesis
   cl10-6              profile=sles10.2_onesis
   cl10-7              profile=sles10.2_onesis
   cl10-8              profile=sles10.2_onesis_lustre
   cl10-9              profile=sles10.2_onesis_lustre
   ```

   In this example, two diskful profiles (`sles10.2` and `sles10.2_lustre`) and two diskless profiles (`sles10.2_onesis` and `sles10.2_onesis_lustre`) are defined.

2. ***For diskful nodes***, update the configuration files with Cfengine using commands similar to the examples below:

   - To update all nodes assigned to the profile `sles10.2`,enter:
     ```
     pdsh -g profile=sles10.2 cfagent
     ```
   - To update selected nodes only, specify the hostnames of the nodes to be updated:
     ```
     pdsh -w cl10-0,cl10-1 cfagent
     ```

3. ***For diskless nodes***, copy the configuration files generated by `gtt` to all oneSIS images by entering the following command on the cluster head node (bash shell syntax):

```
for i in /var/lib/oneSIS/image/* ; do
    cp -r /var/lib/sunhpc/cfengine/[ev]* $i ;
    chown daemon:daemon $i/etc/munge/munge.key
done
```

   This command copies all files in `/var/lib/sunhpc/cfengine/etc` and `/var/lib/sunhpc/cfengine/var` to all oneSIS images.

Setting up Cfengine on a head node

Cfengine must be set up on the head node before the client nodes can be provisioned. For this purpose, a script `/usr/sbin/setup_cfengine` is provided with the Sun HPC Software. During the installation and provisioning of the cluster, this script is run as a part of the `sunhpc_setup` configuration script. See Step B: Prepare the Head Node to Provision the Cluster.

Cfengine requires three variables to be set:

- `policyhost` – The name/IP address of the network interface connecting to all client nodes.
- `domain` – the domain name for the (internal) network.
- `cfnetwork` – the associated network mask.

These variables are set by the `setup_cfengine` script, which executes the following steps:

1. Parses `/etc/cobbler/settings` to determine the correct values for `policyhost` and `cfnetwork`.
2. Tries to find the correct domain name by parsing `/etc/hosts`. If no domain name is found, a default domain name (`sunhpc`) is set.
3. Updates the values found in the Sun HPC Software database `gtdb`.
4. Rebuilds the configuration files for Cfengine by calling `gtt config --update all`.
5. Copies additional configuration files to `/var/lib/sunhpc/cfengine`, such as the munge key and time zone settings.
6. Copies updated Cfengine configuration files (`cfservd.conf` and `update.conf`) to `/var/cfengine/masterfiles/inputs` and `/var/cfengine/inputs`.
7. Starts `cfservd` and adds `cfservd` to the services started at boot.
8. Updates all the configuration files handled by Cfengine on the head node.

Adding Cfengine configuration files

In the default Cfengine configuration provided by the Sun HPC Software, all files in `/var/lib/sunhpc/cfengine/etc` on the head node are copied to `/etc` on all client nodes. These files are:

- `/etc/hosts`
- `/etc/munge/munge.key`
- `/etc/slurm/slurm.conf`
- `/etc/powerman/powerman.conf`
- `/etc/genders`
- `/etc/localtime`
- `/etc/conman.conf`
- `/etc/ntp.conf`

You can include additional configuration files by copying them to `/var/lib/sunhpc/cfengine/etc` or any sub-directory.

Customizing the Cfengine configuration

The Cfengine configuration provided by the Sun HPC Software is minimal. The configuration can be customized by editing the configuration files in `/var/cfengine/masterfiles/inputs` on the head node. To activate the new configuration files, copy them to `/var/cfengine/inputs` on the head node by completing these steps:

1. Update the configuration files on the head node by calling `cfagent` on the head node:
```
cfagent -q --update-only
```
2. Restart the Cfengine server daemon on the head node by entering:
```
/etc/init.d/cfservd restart
```
3. ***For diskful nodes:***

   a. Roll out the new configuration by running `cfagent` on all diskful client nodes. For example, to roll out the configuration to a group of nodes, such as all client compute nodes, enter:
```
# pdsh -g <groupname> cfagent
```
   b. If you have followed the examples for populating the Sun HPC Software Management Database `gtdb` in Step C: Preparing to Provision the Client Nodes, the default group profile can be used to push changes out to all nodes in the database:
```
pdsh -g profile cfagent
```

4. ***For diskless nodes:***

   a. Verify that the images are available on each node. The two methods below should show the same results.

      - Display a list of Cobbler profiles corresponding to the images created:

      ```
      # cobbler list |grep profile
         profile sles10.2
         profile sles10.2-lustre
      ```

      - Display the list of images in `/var/lib/oneSIS/image/`:

      ```
      # ls -lart /var/lib/oneSIS/image/
      total 1
      drwxr-xr-x  3 root root  72 May 12 05:15 ..
      drwxr-xr-x  4 root root 104 May 20 12:33 .
      drwxr-xr-x 25 root root 720 May 20 12:52 sles10.2-lustre
      drwxr-xr-x 26 root root 744 May 21 19:31 sles10.2
      ```

   b. Copy the files from the head node to the the correct oneSIS image. For example:

   ```
   # cp -R /var/lib/sunhpc/cfengine/
   /var/lib/oneSIS/image/sles10.2/
   ```

Note: `cfagent` cannot be used to update diskless images because it attempts to write into `/var` which is read-only for diskless image**s**. Instead, use `cp` to copy the configuration files on the head node into a oneSIS diskless image.

Note: Whenever you update a configuration, you will need to update either the diskful nodes in the cluster or the diskless client images.

More details on how to customize Cfengine can be found at http://www.cfengine.org/docs/cfengine-Tutorial.html.

Using the Cobbler Provisioning Tool

Cobbler is a Linux provisioning server that provides tools for automating software installation on large numbers of Linux systems, including PXE configurations and boots, re-installation, and virtualization. Cobbler provides functions such as:

- Generates configurations from templates for components such as DHCP, PXE, and Kickstart, and manages these configurations.
- Manages repositories including copying a repository from remote repo site and re-creating it locally. Cobbler has been enhanced by Sun to support Boot-over-IB (diskless) and the YaST repositories.
- Provides profile-based provisioning. For example, one profile could be used for a client (such as a Lustre client) and another for a server (such as a Lustre server).

Cobbler supports both a graphical user interface and a command line interface.

During the initial setup of the head node, the `sunhpc_setup` command populates the Sun HPC Software Management Database (`gtdb`) with information about the nodes in the cluster. A cobbler configuration file `cobbler.csv` is generated from the database and then used to provision the clients (for more details, see Chapter 2. Cobbler uses PXE and Kickstart to install the client nodes.

After completing the setup and provisioning process, you may need to make changes to the initial configuration. This section describes how to make changes to the configuration, such as adding or removing a node. For more information about Cobbler, see http://fedorahosted.org/cobbler.

Adding a node

To add a node to the cluster configuration, complete the steps below.

1. Populate the cobbler configuration from the `cobbler.csv` file.

```
# populate_cobbler_system /tmp/cobbler.csv
Internet Systems Consortium DHCP Server V3.0.5-RedHat
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/
Shutting down dhcpd:
Starting dhcpd:                                  done
```

2. Check that the node was added to the configuration.

```
# cobbler list
distro sles10.2
   profile sles10.2
      system cl10-5
      system cl10-6
      system cl10-7
```

```
   profile sles10.2-lustre
      system cl10-8
      system cl10-9
repo sunhpc_base_sles10.2
repo sunhpc_base_sles10.2_yast
repo sunhpc_lustre_sles10.2
repo sunhpc_lustre_sles10.2_yast
repo sunhpc_updates_sles10.2
repo sunhpc_updates_sles10.2_yast
```

Deleting a node

To delete a node from the cluster configuration, complete the steps below.

1. Remove the system from the Cobbler configuration and synchronize the Cobbler configuration files.

```
# cobbler system remove --name=cl10-7
# cobbler sync
```

2. Check that the node was deleted from the configuration.

```
# cobbler list
distro sles10.2
   profile sles10.2
      system cl10-5
      system cl10-6
      system cl10-7
   profile sles10.2-lustre
      system cl10-8
      system cl10-9
repo sunhpc_base_sles10.2
repo sunhpc_base_sles10.2_yast
repo sunhpc_lustre_sles10.2
repo sunhpc_lustre_sles10.2_yast
repo sunhpc_updates_sles10.2
repo sunhpc_updates_sles10.2_yast
```

Changing options in a Cobbler profile

You can change the option settings saved in a cobbler profile using the `cobbler profile edit` command.

1. Check that the profile exists.

```
# cobbler profile list
sles10.2
sles10.2-lustre
```

2. Display the current profile option settings.

```
# cobbler profile report --name=sles10.2
profile                 : sles10.2
distro                  : sles10.2
comment                 :
created                 : Fri Apr 17 04:45:01 2009
dhcp tag                : default
enable menu             : True
kernel options          : {'selinux': '0', 'console': 'ttyS0,9600', \
   'install': 'http://192.168.202.214/cobbler/ks_mirror/sles10.2'}
kickstart               : /etc/cobbler/autoinst.xml
ks metadata             : {'bootdisk': 'hda'}
mgmt classes            : []
modified                : Fri Apr 17 04:45:01 2009
name servers            : []
owners                  : ['admin']
post kernel options     : {}
redhat mgmt key         : <<inherit>>
repos                   : ['sunhpc_base_sles10.2_yast']
server                  : <<inherit>>
template_files          : {}
virt bridge             : xenbr0
virt cpus               : 1
virt file size          : 5
virt path               :
virt ram                : 512
virt type               : xenpv
```

3. Edit the profile. In the example below, the console device is changed from `ttyS0` to `ttyS1`. The `--in-place` option allows you to edit a particular `kopts` value without changing the other values for that option.

```
# cobbler profile edit --name=sles10.2 --in-place
--kopts="console=ttyS1,9600"
```

4. Check your changes by displaying the current profile option settings.

```
# cobbler profile report –name=sles10.2profile              : sles10.2
distro                  : sles10.2
comment                 :
created                 : Fri Apr 17 04:45:01 2009
dhcp tag                : default
enable menu             : True
kernel options          : {'selinux': '0', 'console': 'ttyS1,9600', \
   'install': 'http://192.168.202.214/cobbler/ks_mirror/sles10.2'}
kickstart               : /etc/cobbler/autoinst.xml
ks metadata             : {'bootdisk': 'hda'}
mgmt classes            : []
modified                : Fri Apr 17 08:46:09 2009
name servers            : []
owners                  : ['admin']
post kernel options     : {}
redhat mgmt key         : <<inherit>>
repos                   : ['sunhpc_base_sles10.2_yast']
server                  : <<inherit>>
template_files          : {}
virt bridge             : xenbr0
virt cpus               : 1
```

```
virt file size       : 5
virt path            :
virt ram             : 512
virt type            : xenpv
```

Chapter 5: Monitoring the HPC Cluster

Monitoring the health of an HPC system is an important and ongoing task throughout the life of the system. The Sun HPC Software includes several monitoring tools that provide different views of the HPC system to help detect changes in the system. These tools are:

- ConMan – A serial console management tool that provides an ongoing log of each system's console output.
- Ganglia – A distributed monitoring system utilizing agents on each node that provide in-band information on the running system.
- Nagios – A distributed monitoring system that provides in-band and out-of-band methods for gathering information about the running system.

The following sections describe each of these tools, how to configure them, and what information they can provide.

Using ConMan to Capture and View Console Logs

ConMan provides an ongoing log of the activity seen on the consoles of nodes in the cluster system. In the procedure for setting up ConMan configuration files described in Configuring the ConMan Console Management Tool, the following default log file locations and log file names are set:

`server logdir="/var/log/"` (or the directory containing the ConMan log files)

`server logfile="conman_server"` (the file to which the ConMan server daemon will log)

`global log="conman_client_%N"` (the files to which clients will log, where `%N` is the hostname)

When the HPC system is running, all console output is captured and logged into the appropriate log files. For a healthy system, few entries will appear in these logs. However, in the event of a kernel panic or other node event, you can view these files to see the current or historical output from the console.

ConMan also provides a way to interact directly with the serial console on each of the nodes in the cluster, providing a useful tool for investigating a troubled system.

Using Ganglia to Monitor the Cluster

Ganglia is a scalable, cluster-wide monitoring tool with three main components:

- `gmond` – A daemon that runs on each monitored client.
- `gmetad` – A daemon that runs on the head node.
- Web interface – A user interface located on head node, by default at http://localhost/ganglia.

The `gmond` daemon communicates using Multicast Transport Protocol. Thus, the clients do not require a direct connection to the management (head) node, allowing the head node to collect information in a more efficient manner.

Setting up and starting Ganglia

Ganglia comes pre-configured with SunHPC Software Linux Edition version 2.0 and will typically not require modifications. The Ganglia main screen (see Figure 4) shows an overview of the cluster resources, such as node state, load information and memory usage. In the lower part of the screen, all monitored nodes are listed and their current load information shown. Figure 4 shows a small cluster with one head node and two compute nodes.

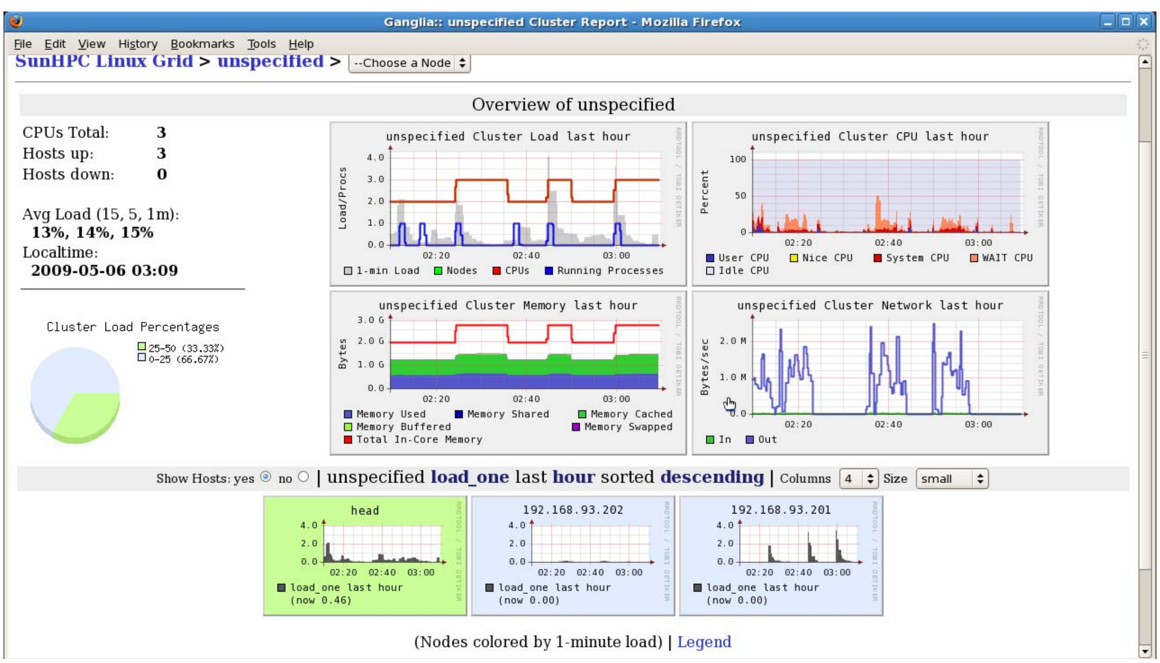

Figure 4. Ganglia main screen

To get more detailed information for a node, click on the image to show information similar to that shown in Figure 5.

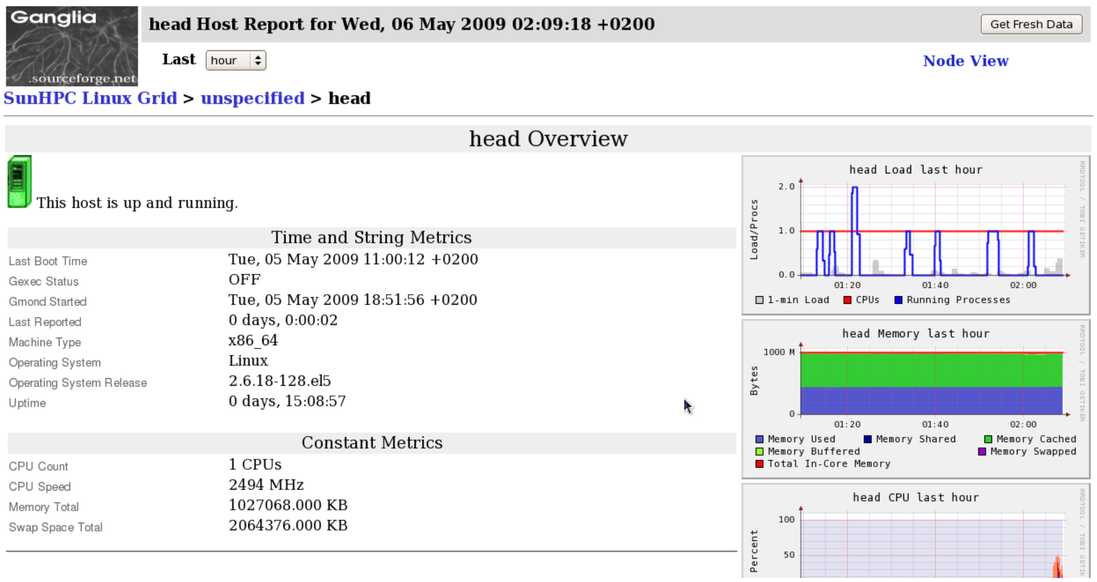

Figure 5. Ganglia node overview

Customizing the Ganglia configuration

To define the cluster name, modify `/etc/ganglia/gmond.conf` on each client. You can use an advanced Ganglia configuration for your cluster environment, but the simplest configuration assumes a single cluster name.

```
# vi /var/lib/oneSIS/image/rhel5.3-onesis/etc/ganglia/gmond.conf

- snip -

* NOT be wrapped inside of a <CLUSTER> tag. */
cluster {
name = "hpc_cluster"
owner = "unspecified"
latlong = "unspecified"
url = "unspecified"
}

- snip -
```

Use `pdsh` to re-start the Ganglia daemon `gmond` on all nodes in the cluster.

```
# pdsh -g profile /etc/init.d/gmond restart
```

On the head node, in the file `/etc/ganglia/gmetad.conf`, change `gridname` to the name of the cluster.

```
# vi /etc/ganglia/gmetad.conf

- snip -

# The name of this Grid. All the data sources above will be wrapped \
in a GRID
# tag with this name.
# default: Unspecified
gridname "hpc_cluster"
#

- snip -
```

Re-start the Ganglia daemon `gmetad` on the head node.

```
# /etc/init.d/gmetad restart
```

Note: If the head node connects to one or more cluster nodes through a network interface other than `eth0` (for example, `eth1`), add an additional `udp_send_channel` and `udp_recv_channel` entry to `/etc/ganglia/gmond.conf` as shown in the example below.

```
udp_send_channel {
 mcast_join = 239.2.11.71
 port = 8649
 ttl =  3
 mcast_if = eth1
}

udp_recv_channel {
 mcast_join = 239.2.11.71
 port = 8649
 bind = 239.2.11.71
 mcast_if = eth1
}
```

Then, restart the Ganglia daemons on the head node:

```
# /etc/init.d/gmetad restart
# /etc/init.d/gmond restart
```

Using Nagios to Monitor the Cluster

Nagios provides a flexible cluster monitoring solution that uses a polling method to retrieve information about different kinds of hardware and software in a cluster.

Nagios communicates through a built-in pull method in contrast to Ganglia, which communicates using Multicast Transport Protocol. The Nagios communication method provides these benefits:

- Easy connection to servers, service processors, and other devices accessible by `ssh`.
- No additional daemon running on client nodes.
- Can be configured to send email alerts.

Nagios and Ganglia provide similar information about the state of a cluster system, but each uses a different method. Either one can be used independently, or both together, depending on the needs of the system.

Nagios on a SunHPC system

Nagios comes pre-configured with a minimal configuration that monitors only the head node of the Sun HPC cluster. The Nagios web interface can be accessed through http://localhost/nagios on the head node of the cluster. The opening screen of the web interface is shown in Figure 6.

The default user/password is `nagiosadmin/nagiosadmin`. It is recommended that you change this as soon as possible. The password file is at `/etc/nagios/htpasswd.users` and can be modified using `htpasswd` or `htpasswd2`.

Figure 6. Nagios web interface start page

Select **Host Groups** to show all monitored systems and their current state. The default configuration will appear similar to Figure 7. A summary of all monitored hosts and services appears at the top of the web page with more detailed information below.

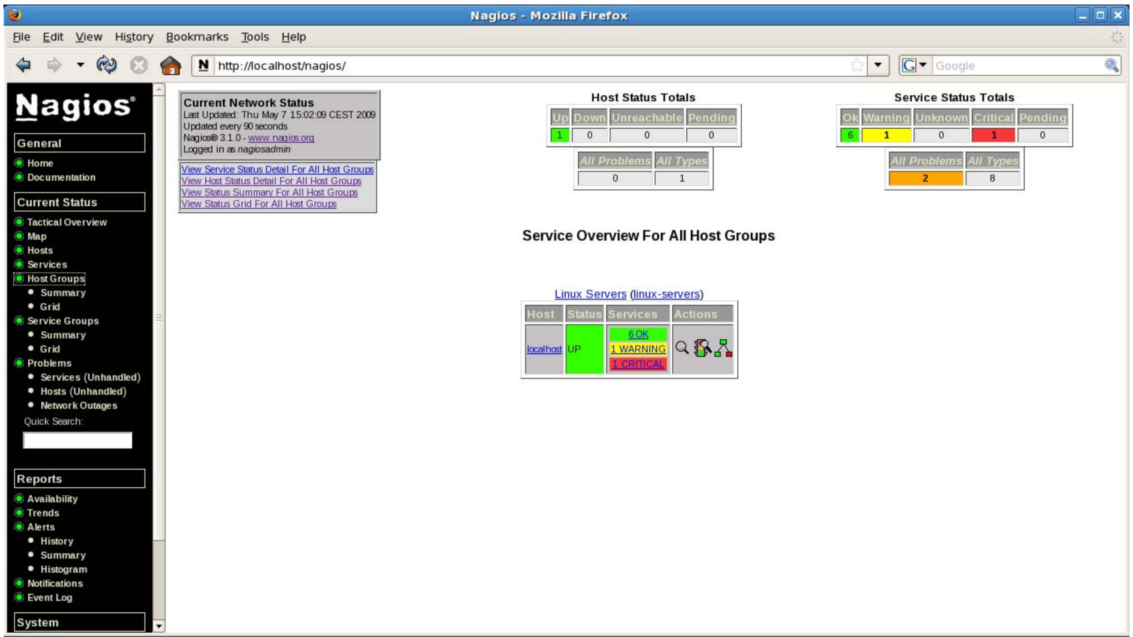

Figure 7. Nagios Host Groups sub-page showing the status of all currently monitored systems

Customizing Nagios

The Sun HPC Software installs the following packages on the management node of the cluster:

- `nagios-3.1.0` – Provides core Nagios functionality.
- `nagios-plugins-1.4.13` – Plug-ins that allow Nagios to monitor other kinds of hardware and software.
- `nagios-www-3.1.0` – Web front-end for Nagios.

To localize the Nagios installation, edit the configuration files in `/etc/nagios` on the head node.

1. In the `/etc/nagios/nagios.cfg` file, set the `cfg_file` variable to point to the local configuration file.

```
cfg_file=/etc/nagios/objects/cluster.cfg
```

2. Copy the file /etc/nagios/objects/localhost.cfg to /etc/nagios/objects/cluster.cfg to use as a template for your cluster and open this file in an editor to complete the following steps.

   a. To create a host entry for each node to be monitored, edit the define host section. For example:

```
define host{
use linux-server ; Name of host template to use
host_name mgmt
alias mgmt
address 10.0.0.100
}
```

   b. To create a hostgroup for each type of service to be monitored, edit the hostgroup section. A hostgroup can contain any arbitrary set of member hosts and is used to make selecting specific groups of hosts easier. For example:

```
define hostgroup{
hostgroup_name vayu ; The name of the hostgroup
alias vayu ; Long name of the group
members cl10-0,cl10-1,cl10-2,hpc-x4540-1,hpc-x4540-2; Comma
separated list of hosts that belong to this group
}
```

   c. To define the services to be monitored and how they will be checked, edit the define services section. For example:

```
define service{
use local-service ; Name of service template to use
host_name cl10-0,cl10-1,cl10-2,hpc-x4540-1,hpc-x4540-2;
service_description PING ;
check_command check_ping!100.0,20%!500.0,60% ;
}
```

   The check_command refers to a test that is defined in the /etc/nagios/objects/commands.cfg file.

To start and stop Nagios, use the following commands:

```
/etc/init.d/nagios start
/etc/init.d/nagios stop
```

If an error is displayed when you start the nagios service daemon, such as "Running configuration check... CONFIG ERROR! Restart aborted. Check your Nagios configuration.", use the command below to view the exact errors in the configuration file.

```
# /usr/sbin/nagios -v /etc/nagios/nagios.cfg
```

The Nagios web front-end display for **Host Groups** is shown in Figure 8 for the example cluster above. Figure 8 shows two services (`ssh` and `ping`), two hostgroups (`sunhpc-server` and `sunhpc-service-processors`) and ten hosts in each group.

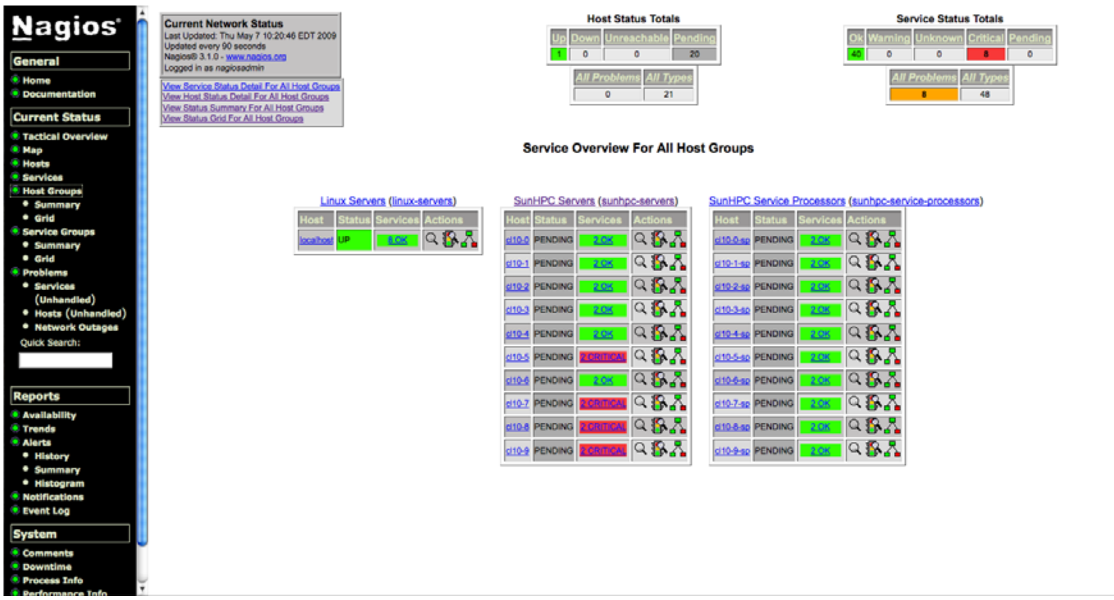

Figure 8. Nagios Host Groups page for example cluster

The **Services** screen for the example cluster is shown in Figure 9.

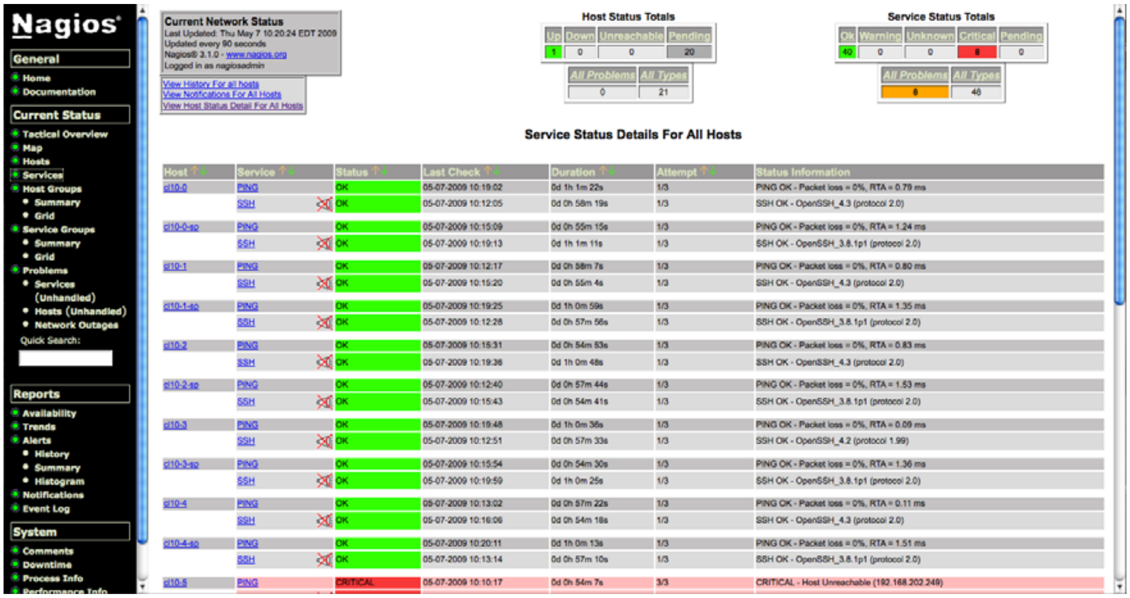

Figure 9. Nagios Services page for example cluster

Nagios can be configured to show much more than what is shown in this document. More information about how to customize Nagios can be found at http://www.nagios.org.

Chapter 6: Parallel Computing

The Sun HPC Software includes a toolkit and set of pre-compiled MPI libraries to help developers develop parallel applications. This section provides an overview of the Sun HPC ClusterTools and the pre-compiled MPI distributions included with the Sun HPC Software, and describes how to add a new MPI distribution.

Using the Sun HPC ClusterTools

Sun HPC ClusterTools 8.1 software is an integrated toolkit based on Open MPI 1.3 that can be used to create and tune Message Passing Interface (MPI) applications that run on high performance clusters. The Sun HPC Software includes Sun HPC ClusterTools 8.1 as the default Message Passing Interface (MPI) distribution. For more information about the Sun HPC ClusterTools 8.1, visit: http://www.sun.com/software/products/clustertools/.

Features of the Sun HPC ClusterTools MPI module

To verify that the Sun HPC ClusterTools 8.1 MPI module is loaded, log into a node on which the Sun HPC Software is installed and enter:

```
# module list
Currently Loaded Modulefiles:
1) clustertools_gcc/8.1
```

The `clustertools_gcc/8.1` module sets the `MANPATH`, the shared library path `LD_LIBRARY_PATH`, and the `PATH` to use `openmpi` compiled with the `gcc` compiler:

The module sets these default paths:

```
MANPATH=/usr/mpi/gcc/clustertools-8.1/share/man:/usr/share/man: \
  /usr/local/man:/usr/X11R6/man:/opt/gnome/share/man

LD_LIBRARY_PATH=/usr/mpi/gcc/clustertools-8.1/lib64

PATH=/usr/mpi/gcc/clustertools-8.1/bin/:/sbin:/usr/sbin:/usr/local/sbin: \
  /opt/gnome/sbin:/root/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin: \
  /usr/games:/opt/gnome/bin:/opt/kde3/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin
```

Default locations are:

```
Shared libraries /usr/mpi/gcc/clustertools-8.1/lib64
Executables: /usr/mpi/gcc/clustertools-8.1/bin/ (mpirun, mpicc, etc)
Include files: /usr/mpi/gcc/clustertools-8.1/include
```

Checking to see if MPI has been installed correctly

1. Verify that the ClusterTools toolkit was correctly installed.

   a. On the head node, enter:

```
# rpm -qa clustertools*
clustertools_pathscale-8.1-sunhpc7
clustertools_intel-8.1-sunhpc7
clustertools_gcc-8.1-sunhpc8
clustertools_sunstudio-8.1-sunhpc8
clustertools_pgi-8.1-sunhpc7
```

   b. Use the `module` command to see which CusterTools have been loaded by default (usually gcc/8.1):

```
# module list
Currently Loaded Modulefiles:
1)  clustertools_gcc/8.1

# which mpirun
/usr/mpi/gcc/clustertools-8.1/bin/mpirun
```

2. Use the `mpirun` command to test communication between the head node and a client node. The `mpirun` command is used to launch an MPI job on a compute resource. This quick test requires a provisioned compute node.

   In the example below, a job is launched from hpc-x4600-2 (head node) to cl10-0 (compute node).The command `/bin/date` is executed on the compute host cl10-0 and the result is returned.

```
hpc-x4600-2:~ # mpirun -host cl10-0 /bin/date
Thu May 21 19:16:52 EDT 2009
```

   Expanding this example to run on multiple hosts:

```
hpc-x4600-2:~ # mpirun -host cl10-0,cl10-1,cl10-2 /bin/hostname
cl10-1
cl10-0
cl10-2
```

At this point, the MPI library has not yet been used to execute code. To do this requires building and running an MPI test program. An example can be found in the section A basic MPI example.

Setting up user accounts

The mpi module must be loaded on all client nodes on which your job will be running. Assuming users have a shared file system that is mounted on all client nodes, one common way to do this is to add the following `.bashrc` file to each user's home directory:

```
# Source global definitions
if [ -f /etc/bashrc ]; then
   . /etc/bashrc
fi
```

This code executes `/etc/profile.d/*.sh`. The result is that `/etc/profile.d/module.sh` loads modules by default, including the `clustertools_gcc` module:

```
Currently Loaded Modulefiles:
   1) clustertools_gcc/8.1
```

If you see the error shown below, it usually indicates that the path to ClusterTools has not been set up because modules have not been initialized.

```
mpirun -host cl10-1,cl10-2 hostname
bash: orted: command not found
```

To address this issue, include the `.bashrc` file shown above in your home directory. The mpi module will then be set up so `mpirun` will work.

Using mpirun

The basic syntax for `mpirun` is:

```
$ mpirun [ -np X ] [ --hostfile <filename> ]  <program>
```

Where:

`-np`	Is the number of copies of the executable that is run on given set of nodes
`--host`	Identifies the names of the hosts on which the program is to be executed
`--hostfile`	Is a plain text file containing the hostnames of hosts on which the program is to be executed
<program>	Is the name of the program that will be execute on the remote hosts.

Below is a basic example showing how to use a hostfile to run a single copy of the hostname command on the remote hosts:

```
hpc-x4600-2:~ # cat myhostfile
cl10-1
cl10-0
cl10-2
cl10-4
cl10-6
cl10-8
```

```
hpc-x4600-2:~ # mpirun -hostfile myhostfile /bin/hostname
cl10-0
cl10-8
cl10-4
cl10-2
cl10-6
cl10-1
```

Because `/bin/hostname` is not an MPI program, it cannot be used to test the MPI library or run multiple copies. However by default the Sun HPC Software installs IOR and hpcc, so the example below shows how to use hpcc to run multiple copies on a set of remote nodes:

```
mpirun -wdir /tmp -np 10 -hostfile myhostfile /usr/bin/hpcc
```

A basic MPI example

Below is a basic “hello world” MPI example from http://beige.ucs.indiana.edu/I590/node60.html:

```
#include <stdio.h>  /* printf and BUFSIZ defined there */
#include <stdlib.h> /* exit defined there */
#include <mpi.h>    /* all MPI-2 functions defined there */

int main(argc, argv)
int argc;
char *argv[];
{
   int rank, size, length;
   char name[BUFSIZ];

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Get_processor_name(name, &length);

   printf("%s: hello world from process %d of %d\n", name, rank, size);

   MPI_Finalize();

   exit(0);
}
```

To compile the code to obtain an executable, enter:

```
# mpicc hello_mpi.c -o hello_mpi.exe
```

Make sure the executable is on a file system that is available to the compute nodes. The run the job and check the results:

```
# cat myhostfile
cl10-0
cl10-1
cl10-2
cl10-4

# mpirun -np 4 --hostfile myhostfile /root/jsalinas/hello_mpi.exe
cl10-1: hello world from process 1 of 4
cl10-4: hello world from process 3 of 4
cl10-2: hello world from process 2 of 4
cl10-0: hello world from process 0 of 4
```

The following example in FORTRAN contains more complex code:

```
# cat mpi.f
C---------------------------------------------------------------------
C     This program times blocking send/receives, and reports the
C     latency and bandwidth of the communication system.  It is
C     designed to run on an even number of nodes. It duplicates the
C     kernel of the Airplane code (I think) so that we can come up with
C     the critical message size.
C
C     Ramesh Menon
C---------------------------------------------------------------------
      program bounce
      parameter (nsizes=8)
      parameter (maxcount=1000000)
      implicit real*8 (a-h,o-z)
      include "mpif.h"
      dimension sbuf(maxcount), rbuf(maxcount)
      dimension length(nsizes),nRepeats(nsizes)
      integer   status(MPI_STATUS_SIZE)

C---------------------------------------
C     define an array of message lengths
C---------------------------------------
      length(1) = 1
      length(2) = 128
      length(3) = 512
      length(4) = 2048
      length(5) = 8192
      length(6) = 32768
      length(7) = 131072
      length(8) = 524288
```

```
      nRepeats(1)=1000
      nRepeats(2)=1000
      nRepeats(3)=1000
      nRepeats(4)=1000
      nRepeats(5)=1000
      nRepeats(6)=1000
      nRepeats(7)=100
      nRepeats(8)=100
C------------------------------------
C     set up the parallel environment
C------------------------------------
      call mpi_init(ierr)
      call mpi_comm_size(mpi_comm_world,nNodes,ierr)
      call mpi_comm_rank(mpi_comm_world,nodeID,ierr)
C
      if (mod(nNodes,2) .ne. 0) then
         if (nodeID .eq. 0) then
            write(6,*) ' You must specify an even number of nodes.'
         end if
         call mpi_finalize(ierr)
         stop
      end if
C--------------------------------------------------------
C     send or receive messages, and time it.
C     even nodes send, odd nodes receive, then the reverse
C--------------------------------------------------------
      do ns=1, nsizes
         call mpi_barrier(MPI_COMM_WORLD, ierr)
         answer=0.d0
         time1 = MPI_Wtime()
         do nr=1, nRepeats(ns)
C---------------------------------------------
C           Change the data on each iteration
C---------------------------------------------
           const=nr+0.1*nodeID
           do i=1,length(ns)
             sbuf(i)=const
           enddo
C---------------------------------------------
C           send in one direction i->i+1 and then
C           send in the reverse direction i+1->i
C---------------------------------------------
           if (mod(nodeID,2) .eq. 0) then
             call mpi_send(sbuf, length(ns), MPI_REAL8, nodeID+1, 1,
     &                 MPI_COMM_WORLD, ierr)
             call mpi_recv(rbuf, length(ns), MPI_REAL8, nodeID+1, 1,
```

```
     &                 MPI_COMM_WORLD, status, ierr)
           else
             call mpi_recv(rbuf, length(ns), MPI_REAL8, nodeID-1, 1,
     &                 MPI_COMM_WORLD, status, ierr)
             call mpi_send(sbuf, length(ns), MPI_REAL8, nodeID-1, 1,
     &                 MPI_COMM_WORLD, ierr)
           end if

C----------------------------------------------
C          Touch all the data received
C----------------------------------------------
           do i=1,length(ns)
             answer=answer+rbuf(i)
           enddo
         end do
         time2 = MPI_Wtime()
C--------------------------------------------------------
C         Now subtract all the additional work done above
C--------------------------------------------------------
         do nr=1, nRepeats(ns)
           const=nr+0.1*nodeID
           do i=1,length(ns)
             sbuf(i)=const
           enddo
           do i=1,length(ns)
             answer=answer+rbuf(i)
           enddo
         enddo
         time3 = MPI_Wtime()
         tottime=2.d0*((time2-time1)-(time3-time2))
         if (nodeID .eq. 0) then
            if (ns .eq. 1) then
              write(6,'(A)')
     &        ' bytes     bandwidth MB/s    Answer     Latency(sec)'
              write(6,'(A)')
     &        ' --------  --------------  -----------  ------------'
            end if
            tlatency = tottime/nRepeats(ns)
            bw = length(ns)*8/((tottime)/nRepeats(ns))/(1024*1024)
            write(6,'(1x,i8,2x,f12.4,3x,f12.0,2x,f12.8)')length(ns)*8,
     &        bw,answer,tlatency
         end if
      end do
      call mpi_finalize(ierr)
      end
```

Compile the code:

```
# mpif90 mpi.f -o mpi.exe
```

Run the job:

```
# mpirun -np 2 --hostfile myhostfile /root/mpi.exe
 bytes     bandwidth MB/s    Answer     Latency(sec)
 --------  --------------  -----------  ------------
        8         0.0382       1500700.    0.00019977
     1024         4.9445     192089596.    0.00019751
     4096         9.3574     768358383.    0.00041745
    16384        16.0116    3073433530.    0.00097585
    65536        19.7303   12293734120.    0.00316772
   262144        25.0321   49174936481.    0.00998717
  1048576        26.8474    1975255007.    0.03724750
  4194304        27.6401    7901020029.    0.14471742
```

Using the ClusterTools libraries with gcc and makefile

The Sun HPC Software provides pre-compiled MPI libraries and tools to use for parallel jobs that use MPI. The example below shows how to compile an MPI application and link in the ClusterTools MPI libraries. This example is taken from the `eff_bw` communications benchmark included in the Pallas MPI Benchmark (PMB) Suite. The benchmark follows a format sometimes used in which a makefile with generic information is provided that can be edited to make it system-specific. A set of basic `make_xxx` files are provided with the `eff_bw` package. In this example, the file `make_linux` is edited to include system-specific information for the application to be compiled. With this method, users do not have to edit the makefile directly, but provide the same basic type of information that would be edited in a makefile. Here is the example:

```
cat make_linux
MPI_HOME = /usr/mpi/gcc/clustertools-8.1
MPI_INCLUDE =$(MPI_HOME)/include
LIB_PATH = -L$(MPI_HOME)/lib64/
LIBS = -lmpi
CC = gcc
CLINKER = gcc
CPPFLAGS = -DnoCHECK

# make
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c EFF_BW.c
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c declare.c
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c EFF_BW_init.c
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c BenchList.c
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c Warm_up.c
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c PingPong.c
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c Output.c
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK -c err_handler.c
gcc -o EFF_BW EFF_BW.o declare.o EFF_BW_init.o BenchList.o g_info.o
Warm_up.o PingPong.o Output.o err_handler.o \
  -L/usr/mpi/gcc/clustertools-8.1/lib64/ -lmpi
```

Using the ClusterTools libraries with mpicc and makefile

The clustertools module also provides MPI tools. One of these tools is mpi*XX* (mpicc, mpiCC, mpif90, etc) to help compile MPI programs. All of these programs are links to `opal_wrapper`. For more information see `man opal_wrapper`.

The example below shows how to compile the `eff_bw` benchmark with mpicc:

```
cat make_linux
MPI_HOME = /usr/mpi/gcc/clustertools-8.1
MPI_INCLUDE =$(MPI_HOME)/include
LIB_PATH = -L$(MPI_HOME)/lib64/
LIBS = -lmpi
CC = mpicc
CLINKER = mpicc
CPPFLAGS = -DnoCHECK

# make
mpicc -DnoCHECK -c EFF_BW.c
mpicc -DnoCHECK -c declare.c
mpicc -DnoCHECK -c EFF_BW_init.c
mpicc -DnoCHECK -c BenchList.c
mpicc -DnoCHECK -c Warm_up.c
mpicc -c PingPong.c
mpicc -DnoCHECK -c Output.c
mpicc -DnoCHECK -c err_handler.c
mpicc -o EFF_BW EFF_BW.o declare.o EFF_BW_init.o BenchList.o g_info.o
Warm_up.o PingPong.o Output.o err_handler.o -L/usr/mpi/gcc/clustertools-8.1/
lib64/ -lmpi
```

Running an MPI application

The MPI module has already set up the shared library path `LD_LIBRARY_PATH` and `PATH` to allow your application to use the library at run time.

Use the `mpirun` command to start the application:

```
# which mpirun
/usr/mpi/gcc/clustertools-8.1/bin/mpirun

# mpirun -np 2 ./EFF_BW
# Running PingPong; see file "bench.out_2" for results
# Running PingPong; see file "bench.out_2" for results

************************************************************

Running on 2 PEs

sampling from 2^0 to 2^20 bytes

Effective Bandwidth: 1070.69 [MB/sec]

************************************************************
```

Note: Normally, `mpirun` or `mpirun_rsh` is run from a login node used to gain access to the cluster and the hosts are client compute nodes dedicated to running compute jobs.

For more information, visit the Sun HPC ClusterTools 8 Documentation website at:
http://docs.sun.com/app/docs/coll/hpc-clustertools8?l=en

Using the SunStudio/PGI/Intel/Pathscale Compilers

Sun HPC Software provides pre-compiled HPC MPI distributions for these compilers:

- gcc 4.1.2
- Sunstudio 11/2008
- Intel compiler version 11.0
- Pathscale compiler version 3.2
- PGI compiler version 8.0-3

Installing additional MPI distributions

The Sun HPC Software includes RPMs for MVAPICH and MVAPICH2.

MVAPICH :

```
/media/sun_hpc_linux/SunHPC/x86_64/mvapich_pathscale-1.1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/mvapich_sunstudio-1.1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/mvapich_gcc-1.1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/mvapich_pgi-1.1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/mvapich_intel-1.1-sunhpc5.x86_64.rpm
```

MVAPICH2:

```
/media/sun_hpc_linux/SunHPC/x86_64/mvapich2_pgi-1.2p1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/mvapich2_intel-1.2p1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/mvapich2_gcc-1.2p1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/\
  mvapich2_pathscale-1.2p1-sunhpc5.x86_64.rpm
/media/sun_hpc_linux/SunHPC/x86_64/\
  mvapich2_sunstudio-1.2p1-sunhpc5.x86_64.rpm
```

To install either of these RPMs, use `yum`. For example:

```
# yum install mvapich2_gcc-1.2p1-sunhpc5
Setting up Install Process
Parsing package install arguments
Resolving Dependencies
--> Running transaction check
---> Package mvapich2_gcc.x86_64 0:1.2p1-sunhpc5 set to be updated
--> Finished Dependency Resolution
```

```
Dependencies Resolved

=============================================================================
Package Arch Version Repository Size
=============================================================================
Installing:
mvapich2_gcc x86_64 1.2p1-sunhpc5 sunhpc-local 8.6 M

Transaction Summary
=============================================================================
Install 1 Package(s)
Update 0 Package(s)
Remove 0 Package(s)

Total download size: 8.6 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
Installing: mvapich2_gcc ######################### [1/1]

Installed: mvapich2_gcc.x86_64 0:1.2p1-sunhpc5
Complete!
```

To install another MPI distribution into a diskless image, use the `yum` command with the `--installroot` option. This can only be done after `sunhpc_setup` has been used to set up diskless images. The steps are:

1. Use the oneSIS tool `mk-sysimage` to revert the links in the diskless image.
2. Use `yum` to install the software into the image.
3. Rerun `mk-sysimage` to reestablish the links in the diskless image.

Below is an example of the command to do this in a SLES 10.2 diskless image created by `sunhpc_setup`. For RHEL, change the image name to `rhel5.3`.

```
# mk-sysimage -r /var/lib/oneSIS/image/sles10.2
# yum --installroot /var/lib/oneSIS/image/sles10.2 \
  groupinstall "SunHPC  MVAPICH Packages"
# mk-sysimage /var/lib/oneSIS/image/sles10.2
```

Note: This procedure will not install the MPI packages onto the Lustre nodes because, by default, `sunhpc_setup` creates Lustre servers as separate images. To see the images installed on your system run `cobbler list` and `ls /var/lib/oneSIS/image/`.

For diskful nodes, you must add the MVAPICH group to either AutoYaST (for SLES) or Kickstart (for RHEL). The example below shows how an AutoYaST file is edited to install extra packages into a SLES diskful image:

```
  <software>
    <patterns config:type="list">
      <pattern>base</pattern>
#if $varExists('lustreserver')
      <pattern>SunHPC_Lustre_Node</pattern>
#else
      <pattern>SunHPC_Client_Node</pattern>
     <pattern>SunHPC MVAPICH</pattern>
#end if
    </patterns>
    <remove-packages  config:type="list">
      <package>open-iscsi</package>
      <package>jre</package>
    </remove-packages>
    <post-packages config:type="list">
      <package>jre</package>
#if not $varExists('lustreserver')
      <package>modules</package>
#end if
      <package>gcc</package>
      <package>gcc-c++</package>
      <package>gcc-fortran</package>
    </post-packages>
    <packages config:type="list">
      <package>cfengine</package>
      <package>pdsh</package>
    </packages>
  </software>
```

The example below shows how a Kickstart file is edited to install MPI packages into a RHEL diskful image:

```
echo "Preparing to install SunHPC Software... " | tee $LOG > /dev/console
echo | tee -a $LOG > /dev/console
yum -q makecache

# Remove default OFED in RHEL5.x and install SunHPC software stack
yum -y --disablerepo=$DISTRO groupinstall "SunHPC OFED Infiniband Packages"
| tee -a $LOG > /dev/console

yum -y groupinstall "SunHPC Cluster Verification Tools" "SunHPC Default MPI
Packages" "SunHPC SLURM" | tee -a $LOG > /dev/console
```

```
yum -y groupinstall “SunHPC MVAPICH Packages” | tee -a $LOG > /dev/console
<------- Add This
yum -y install ganglia-gmond | tee -a $LOG > /dev/console
#if $varExists('lustreserver')
yum -y groupinstall "SunHPC Lustre Server" | tee -a $LOG > /dev/console
#else
yum -y groupinstall "SunHPC Lustre Client" | tee -a $LOG > /dev/console
#end if

yum -y --disablerepo=$DISTRO install modules env-switcher cfengine pdsh
pdsh-mod-genders pdsh-rcmd-ssh conman powerman freeipmi ipmitool genders
genders-compat lshw jre fping kernel-ib | tee -a $LOG > /dev/console
```

To get information about an installed package, use `rpm -q`. For example:

```
# rpm -q mvapich2_gcc
mvapich2_gcc-1.2p1-sunhpc5
```

Note: Once the RPMs are installed into a provisioned diskless image, the client must be rebooted to pick up the changes. For diskful nodes, make the changes to Kickstart or AutoYaST before the nodes are provisioned or it will be necessary to re-provision the node.

Using MVAPICH2

In the version of MVAPICH2 provided with the Sun HPC Software, a new tool `mpirun_rsh` has been added to run jobs on compute clusters. This is the preferred tool for large clusters or clusters with an InfiniBand network.The `mpirun_rsh` command is used in place of `mpirun`. The MVAPICH2 web page states, “The `mpirun_rsh/mpispawn` framework launches jobs on demand in a manner more scalable than `mpd/mpiexec`. Using `mpirun_rsh` also alleviates the need to start daemons in advance on nodes used for MPI jobs.” For more information, see the MVAPICH2 documentation at http://mvapich.cse.ohio-state.edu/support/user_guide_mvapich2-1.2.html.

For example:

```
mpirun_rsh -ssh -n 1 f0012 /bin/date
```

Normally, `mpirun_rsh` is run from a login node used to gain access to the cluster and the hosts are client compute nodes dedicated to running compute jobs.

The method used to connect to client nodes is `-ssh` or `-rsh`. Either `ssh` keys or `.rhosts` will have to be set up. In the example, `-ssh` is used. `-n` is the number of processors. The next argument specifies the host or hosts to run on (which can also be specified as a file containing a list of hosts). The final argument specifies the full path to the executable.

If you use the MVAPICH `mpirun` command rather than `mpirun_rsh`, you will need to set up a `.mpd.conf` file for each user and run the multi-purpose `mpd` daemon on each client. A brief overview of `mpd` is provided at
http://www.physics.drexel.edu/~valliere/PHYS405/MPI2/MPI2.html#Daemons.

For more information, see:

http://debianclusters.cs.uni.edu/index.php/MPICH_without_Torque_Functionality

http://debianclusters.cs.uni.edu/index.php/MPICH:_Starting_a_Global_MPD_Ring

http://debianclusters.cs.uni.edu/index.php/MPICH:_Troubleshooting_the_MPD

Building an MPI application

To select the MPI module to be used to build the MPI application, use the `module switch` command:

```
# module switch clustertools_gcc/8.1 mvapich2_gcc/1.2p1
# module list
Currently Loaded Modulefiles:
1) mvapich2_gcc/1.2p1
MANPATH=/usr/mpi/gcc/mvapich2-1.2p1/share/man:/usr/share/man:\
  /usr/local/man:/usr/X11R6/man:/opt/gnome/share/man
LD_LIBRARY_PATH=/usr/mpi/gcc/mvapich2-1.2p1/lib64
PATH=/usr/mpi/gcc/mvapich2-1.2p1/bin/:/sbin:/usr/sbin:/usr/local/sbin: \
  /opt/gnome/sbin:/root/bin:/usr/local/bin:...etc...
```

Compile and link the application using commands similar to those in the example below:

```
mpi library path /usr/mpi/gcc/mvapich2-1.2p1/lib64
mpi include path /usr/mpi/gcc/mvapich2-1.2p1/include

To compile:
gcc -I/usr/mpi/gcc/clustertools-8.1/include -DnoCHECK

To link:
gcc -o MPI mpi.o test.o -L/usr/mpi/gcc/clustertools-8.1/lib64/ -lmpi
```

Running an MPI application

Use the `mpirun` command to run an MPI application on one or more compute nodes. All the MPI distributions included with the Sun HPC Software provide an `mpirun` command.

Important `mpirun` command options are:

- `-n` or `-np` – Number of processors across which to run the job
- `-wdir` – Working directory
- `-host` – Host on which to run the job

An example using this command to run the application `./a.out` is shown below:

```
mpirun -host hpc-x4600-2 -n 4 ./a.out
```

On compute nodes where `slurm` is running on the cluster, you can use the `srun` command to quickly run a job. Important `srun` options are:

- `-n` – Number of tasks
- `-N` – Number of nodes to run on
- `-c` – Number of CPUs per task

In this example, `srun` is used to run IOR on two nodes with two tasks per node:

```
srun -N 2 -n 2 /usr/bin/IOR -t 1m -b 1m -F -i 200
```

Running an Intel MPI Benchmark

The Intel MPI Benchmark (IMB) suite includes a series of tests to measure the MPI performance of a cluster. A simple example showing how to compile and run an IMB MPI benchmark is shown below:

```
#MPI_HOME = ${MPICH}
MPI_HOME = /usr/mpi/gcc/clustertools-8.1
MPI_INCLUDE = $(MPI_HOME)/include
LIB_PATH = -L$(MPI_HOME)/lib64
LIBS = -lmpi
CC = ${MPI_HOME}/bin/mpicc
OPTFLAGS = -O3
CLINKER = ${CC}
LDFLAGS =
CPPFLAGS =

# make
touch exe_mpi1 *.c; rm -rf exe_io exe_ext
make MPI1 CPP=MPI1
make[1]: Entering directory `/IMB-src'
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_declare.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_init.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_mem_manager.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_parse_name_mpi1.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_benchlist.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_strgs.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_err_handler.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_g_info.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_warm_up.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_output.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_pingpong.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_pingping.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_allreduce.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_reduce_scatter.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_reduce.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_exchange.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_bcast.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_barrier.c
```

```
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_allgather.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_allgatherv.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_alltoall.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_sendrecv.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_init_transfer.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_chk_diff.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -DMPI1 -O3 -c IMB_cpu_exploit.c
/usr/mpi/gcc/clustertools-8.1/bin/mpicc -o IMB-MPI1 IMB.o IMB_declare.o
IMB_init.o IMB_mem_manager.o IMB_parse_name_mpi1.o IMB_benchlist.o
IMB_strgs.o IMB_err_handler.o IMB_g_info.o IMB_warm_up.o IMB_output.o
IMB_pingpong.o IMB_pingping.o IMB_allreduce.o IMB_reduce_scatter.o
IMB_reduce.o IMB_exchange.o IMB_bcast.o IMB_barrier.o IMB_allgather.o
IMB_allgatherv.o IMB_alltoall.o IMB_sendrecv.o IMB_init_transfer.o
IMB_chk_diff.o IMB_cpu_exploit.o -L/usr/mpi/gcc/clustertools-8.1/lib64 -lmpi
make[1]: Leaving directory `/root/2.0/jsalinas/mpi/IMB-src'
mpirun -np 2 <path> /IMB-MPI1 or srun -n2 <path>/IMB-src/IMB-MPI1
```

Note: The RPMs and modules must be set up appropriately before compiling an application. For example, if you compile with `mvapich2/gcc` on a login node, you must make sure the RPM `mvapich2/gcc` is installed and the module `mvapich2/gcc` is loaded before compiling your application.

Results are shown below:

```
#---------------------------------------------------
# Intel (R) MPI Benchmark Suite V2.3, MPI-1 part
#---------------------------------------------------
# Date : Thu Apr 16 20:25:51 2009
# Machine : x86_64# System : Linux
# Release : 2.6.16.60-0.21-smp
# Version : #1 SMP Tue May 6 12:41:02 UTC 2008

#
# Minimum message length in bytes: 0
# Maximum message length in bytes: 4194304
#
# MPI_Datatype : MPI_BYTE
# MPI_Datatype for reductions : MPI_FLOAT
# MPI_Op : MPI_SUM
#
#

# List of Benchmarks to run:

# PingPong
# PingPing
# Sendrecv
# Exchange
# Allreduce
# Reduce
# Reduce_scatter
# Allgather
```

```
# Allgatherv
# Alltoall
# Bcast
# Barrier

#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
#bytes #repetitions t[usec] Mbytes/sec
0 1000 0.44 0.00
1 1000 0.47 2.04
2 1000 0.47 4.08
4 1000 0.47 8.15
8 1000 0.47 16.20
16 1000 0.47 32.19
32 1000 0.53 57.16
64 1000 0.54 113.68
128 1000 0.56 219.13
256 1000 0.64 383.02
512 1000 0.85 576.50
1024 1000 1.18 830.41
2048 1000 1.88 1040.25
4096 1000 3.00 1302.75
8192 1000 4.89 1599.10
16384 1000 9.30 1680.65
32768 1000 18.29 1708.82
65536 640 30.58 2043.52
131072 320 54.54 2292.00
262144 160 102.78 2432.27
524288 80 202.36 2470.89
1048576 40 408.97 2445.14
2097152 20 1115.97 1792.16
4194304 10 2325.11 1720.35
...etc...
```

Using Modules to Handle Additional MPI Distributions

If you have installed one or more additional MPI distributions, you will need to set up your environment to use the compiled version you need.

1. To view a list of the available MPI distributions, enter:

```
# module avail

------------------------------------ /usr/share/Modules/modulefiles
------------------------------------
clustertools_gcc/8.1       mvapich2_intel/1.2p1      \
     mvapich_pathscale/1.1
clustertools_intel/8.1     mvapich2_pathscale/1.2p1  \
     mvapich_pgi/1.1
clustertools_pathscale/8.1 mvapich2_pgi/1.2p1        \
     mvapich_sunstudio/1.1
clustertools_pgi/8.1       mvapich2_sunstudio/1.2p1  \
    switcher/1.0.13(default)
clustertools_sunstudio/8.1 mvapich_gcc/1.1
mvapich2_gcc/1.2p1         mvapich_intel/1.1
```

This example shows three MPI distributions, `clustertools`, `mvapich2` and `mvapich`, each of which has been compiled with a `gcc`, `intel`, `pathscale`, `pgi` and `sunstudio` compiler.

2. Load the correct module.

   a. To see which module is currently loaded, enter:

```
# module list
Currently Loaded Modulefiles:
  1) clustertools_gcc/8.1
```

   b. To change to another module, for example, to a `clustertools` MPI distribution that has been compiled with the `intel` compiler, use the `module switch` command:

```
# module switch clustertools_gcc/8.1 clustertools_intel/8.1
```

   When the new module is loaded, the following environment variables are updated:

```
MANPATH=/usr/mpi/intel/clustertools-8.1/share/man:\
  /usr/share/man:/usr/local/man:\
  /usr/X11R6/man:/opt/gnome/share/man
LD_LIBRARY_PATH=/usr/mpi/intel/clustertools-8.1/lib64
PATH=/usr/mpi/intel/clustertools-8.1/bin/:/sbin:/usr/sbin:\
  /usr/local/sbin:/opt/gnome/sbin:/root/bin:/usr/local/bin:\
  /usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/gnome/bin:\
  /opt/kde3/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin
_LMFILES_=/usr/share/Modules/modulefiles/clustertools_intel/8.1
LOADEDMODULES=clustertools_intel/8.1
```

   It is possible to switch any MPI distribution for any other. In this example, the `clustertools_intel` module is changed to the `mvapich2_pathscale` module:

```
# module list
Currently Loaded Modulefiles:
  1) clustertools_intel/8.1
# module switch  clustertools_intel/8.1 mvapich2_pathscale/1.2p1
# module list
Currently Loaded Modulefiles:
  1) mvapich2_pathscale/1.2p1
```

   Your environment is now ready to build or run your code.

3. Build or run your code.

   a. Verify the correct module is loaded:

```
# module list
Currently Loaded Modulefiles:
  1) clustertools_intel/8.1
```

b. Build an MPI application with the `mpicc` compiler or run code compiled with `clustertools_gcc/mpi`. For more information,see Building an MPI Application and Running an MPI Application.

Chapter 7:
Managing Compute Resources

The Sun HPC Software, Linux Edition 2.0 includes two commonly used tools for managing compute resources: Sun Grid Engine and SLURM.

Sun Grid Engine

Sun Grid Engine is integrated into the Sun HPC Software 2.0 release. This section explains how to install and configure Sun Grid Engine on the HPC cluster and schedule a simple job.

Sun Grid Engine online resources include:

- Sun Grid Engine Product page
- Video: Introduction to Grid Engine
- Beginner's Guide to Sun Grid Engine 6.2 Installation and Configuration White Paper
- Sun Grid Engine Wikis

You may also find it useful to attend a Sun Grid Engine training or seek Sun Grid Engine professional support: http://www.sun.com/software/sge/support.xml

Overview of Sun Grid Engine

The Sun Grid Engine system does the following:

- Accepts jobs from the outside world. Jobs are users' requests for computing resources.
- Puts jobs in a holding area until enough resources are available to execute them.
- Schedules jobs from the holding area to execution devices.
- Manages running jobs.
- Logs a record of job execution when jobs are finished.
- May be used to generate usage statistics and do accounting.

Four types of hosts are distinguished in a Sun Grid Engine system:

- *Master host* – The master host, also commonly referred as "qmaster", is central to the overall cluster activity. The master host runs the master daemon `sge_qmaster`. This daemon controls job scheduling and monitors components, such as queues and jobs. The daemon maintains tables that contain information such as the status of the components and user access permissions. By default, the master host is also an administration host.

- *Execution hosts* – Execution hosts are systems that can be used to execute jobs. Therefore, queue instances are attached to the execution hosts. Execution hosts run the execution daemon `sge_execd`.
- *Administration hosts* – Administration hosts are hosts that can be used to carry out any kind of administrative activity for the Sun Grid Engine system by an authorized user.
- *Submit hosts* – Submit hosts enable users to submit and control batch jobs only. In particular, a user who is logged in to a submit host can submit jobs with the `qsub` command, can monitor the job status with the `qstat` command, and can use the Sun Grid Engine system OSF/1 Motif graphical user interface QMON, which is described in *QMON, the Grid Engine System's Graphical User Interface* in the N1 Grid Engine 6 User's Guide (see *Introduction*).

Preparing a Sun Grid Engine installation

Sun Grid Engine (SGE) can be installed on a shared file system as well as on non-shared file systems. Most computing jobs running on a HPC cluster need a shared file system (such as the Lustre file system) to access programs and data. The same file system can be used to install SGE.

If Sun Grid Engine is to be installed on a shared file system, ensure that the shared file system is set up and configured correctly and can be accessed by all nodes (for read and write), before installing SGE components.

If local file systems will be used for SGE, at least 100 MB hard disk space must be available on each node. During execution of SGE jobs, additional hard disk space may be required to store information such as spooling information.

In most cases it is useful to use pdsh to execute commands on all SGE execution hosts. You can set up pdsh to execute commands by completing these steps:

1. Add an extra attribute to each execution host's configuration using the Sun HPC Software Management Tool `gtt`. This makes it possible to address all SGE execution hosts at once using pdsh.
```
# gtt host --addattr --name node0001 --attribute sgeexec
```
2. Update the configuration.
```
# gtt config --update genders
```
3. Use `cfagent` to write the updated configuration file to `/etc/genders` on the head node.
```
# cfagent
```

Before installing Sun Grid Engine on an HPC cluster, collect the information shown in the table below:

Parameter	***Example Value***
sge-root directory	`/gridware/sge/`
Cell name	default
Administrative User	root or sgeadmin
sge_qmaster port number	6444
sge_execd port number	6445
Master host	sge-master
Shadow master host	sge-shadow-master
Execution hosts	sge-exec-[001-xxx]
Administration hosts	sge-master, sge-shadow-master, sge-exec-[001-xxx]
Submit hosts	sge-master, sge-shadow-master, sge-exec-[001-xxx]
Group ID range for jobs	20000~20100
Spooling mechanism (Berkeley DB or Classic spooling)	Classic

Install Sun Grid Engine on a shared file system

To install SGE on a shared file system, complete the steps below.

1. Install the RPM packages on the SGE master node:

```
# yum groupinstall "SunHPC SGE"
```

2. When the `yum` installation is completed, verify the Sun Grid Engine installation:

```
[root@headnode ~]# ls /gridware/sge/
3rd_party  dbwriter  include         lib  qmon                 utilbin
bin        doc       install_execd   man  reporting
catman     dtrace    install_qmaster mpi  start_gui_installer
ckpt       examples  inst_sge        pvm  util
```

This will install SGE software on the master node in `/gridware/sge`.

This directory can be copied (or moved) to a directory on a shared file system by entering a command similar to:

```
# cp -r /gridware/sge /lustre/software/
```

Note:

- Refer to the Lustre documentation at wiki.lustre.org/index.php/Lustre_Howto for more information about setting up Lustre.
- Passwordless ssh access is configured by default as part of the provisioning process. For more information see Setting Up SSH Keys in Chapter 4.

Installing Sun Grid Engine qmaster

To install the SGE qmaster, complete the steps below.

1. Go to the new `sge` directory and call `install_qmaster`:

```
# cd /lustre/software/sge
# ./install_qmaster
```

   The install script will guide you through the installation process by asking a number of questions. Most can be answered using information in the table above. Check that the SGE_ROOT value is set correctly to the new `sge` directory. More information on installing and configuring the SGE qmaster can be found on the Installing Sun Grid Engine page on the SGE wiki.

2. Check that the `sge_qmaster` daemon is running on the master host:

```
# ps -ax |grep sge
16435 ? Sl 0:01 /gridware/sge/bin/lx24-amd64/sge_qmaster
17437 ttyS0 S+ 0:00 grep sge
```

3. To make sure the SGE settings are correctly loaded after login, link the settings files to `/etc/profile.d/`. Assuming the cell name is set to "default" and SGE software is installed in `/lustre/software/sge`, enter:

```
# ln -s /lustre/software/sge/default/common/settings.sh \
  /etc/profile.d/sge.sh
# ln -s /lustre/software/sge/default/common/settings.csh \
  /etc/profile.d/sge.csh
```

Install the SGE execution hosts

To install an execution host on a shared file system, complete the following steps:

1. Check to see if the node to be added is already known to the SGE qmaster:

```
# qconf -sh
sge-qmaster
```

2. Make the new execution host known to the SGE qmaster. For example, if the new execution host is `node0001`, enter the following command on the qmaster node to add the node to the administrative host list:

```
# qconf -ah node0001
```

3. Login to the new execution host.

4. Change to the SGE installation directory. For example:

```
# cd /lustre/software/sge
```

5. Call `install_execd` and answer the questions that are displayed:

```
# ./install_execd
```

To automate the execution host installation process, an installation configuration file must be defined. A template can be found in `$SGE_ROOT/util/install_modules/inst_template.conf`. After creating a configuration file from this template (e.g. `my_sge_inst.conf`) and storing it in `$SGE_ROOT`, an execution host can be installed using:

```
# ./install_execd -auto my_sge_inst.conf
```

Installing Sun Grid Engine on non-shared file systems

Install the Sun Grid Engine qmaster

If a shared file system is not used to install SGE, execute the following steps to install an SGE qmaster:

1. Install the SGE RPMs:

```
# yum groupinstall "SunHPC SGE"
```

2. Go to `/gridware/sge`:

```
# cd /gridware/sge
```

3. Call install_qmaster and answer the question according to the information collected in the table above:

```
# ./install_qmaster
```

4. Set up the environment for future logins:

```
# ln -s /gridware/sge/default/common/settings.sh /etc/profile.d/sge.sh
# ln -s /gridware/sge/default/common/settings.csh /etc/profile.d/sge.csh
```

Install the SGE execution hosts

To set up SGE execution hosts, complete the following steps:

1. Install the SGE RPMs on all diskful execution hosts from the cluster's head node.

   For RHEL/CentOS, enter:

   ```
   # pdsh -g sge-diskful-exec 'yum -y groupinstall "SunHPC SGE"'
   ```

   For SLES, enter:

   ```
   # pdsh -g sgeexec 'zypper –no-gpg-checks -n in -t pattern SunHPC_SGE'
   ```

   Note: This will not work for diskless clients, since the root file system is mounted as read-only on these nodes and the sge_execd needs to write spooling information to the file system.

   ---

2. Copy the head node configuration to all execution hosts (assuming the default is the cell name chosen on SGE's qmaster):

   ```
   # pdsh -g sgeexec mkdir -p /gridware/sge/default/common
   # pdcp -g sgeexec /gridware/sge/default/common/* \
     /gridware/sge/default/common/
   ```

3. Install the SGE execution daemon on all nodes, login to all nodes, and install the execution daemon.

   ```
   # cd /gridware/sge
   # ./install_execd
   ```

   It is recommended that you use a configuration file. A template can be found in `$SGE_ROOT/util/install_modules/inst_template.conf`. By adapting the template to your settings and saving it as `my_sge_inst.conf`, you can install the execution daemon automatically by entering the following on all execution hosts:

   ```
   #./install_execd -auto my_sge_inst.conf
   ```

4. Set up the environment for future logins

   ```
   # pdsh -g sgeexec ln -s /gridware/sge/default/common/settings.sh \
     /etc/profile.d/sge.sh
   # pdsh -g sgeexec ln -s /gridware/sge/default/common/settings.csh \
     /etc/profile.d/sge.csh
   ```

Configuring and testing the installed Sun Grid Engine instance

To test that the SGE instance is correctly installed, obtain the current state of all SGE execution hosts by entering:

```
# qstat -f
```

Tools such as `qmod`, `qmon` and `qconf` can be used to make modifications to the SGE instance such as changing or defining queues, users, parallel environments, projects or resource quotas.

Schedule a simple job

The Sun Grid Engine software includes a number of example job scripts that can be used to verify that SGE is working correctly. One of these job scripts is `simple.sh`, which can be found in the subdirectory `examples/jobs`.

```
#!/bin/sh
#
# (c) 2009 Sun Microsystems, Inc. All rights reserved. Use is subject to
license terms.
# This is a simple example of a SGE batch script
# request Bourne shell as shell for job
#$ -S /bin/sh
#
# print date and time
date
# Sleep for 20 seconds
sleep 20
# print date and time again
date
```

To submit this job script to SGE, use the `qsub` command. The state of the job can be monitored using `qstat`:

```
# qsub $SGE_ROOT/examples/jobs/simple.sh
Your job 2 ("simple.sh") has been submitted
#
# qstat
job-ID prior name      user state submit/start at    queue  slots ja-task-ID
-----------------------------------------------------------------------------
  2  0.00000 simple.sh root  qw   03/29/2009 18:52:15       1
#
# qstat
job-ID prior name      user state submit/start at    queue  slots ja-task-ID
-----------------------------------------------------------------------------
  2  0.55500 simple.sh root   r   03/29/2009 18:52:28 all.q@node0001  1
#
# qstat
#
```

Further information about administering SGE can be found at wikis.sun.com/display/GridEngine/Administering+Sun+Grid+Engine.

SLURM

SLURM is included in the Sun HPC Software as an open source scheduler. This section briefly describes how to install, setup, and run SLURM on an HPC cluster. For more information about SLURM, see:

- SLURM home page https://computing.llnl.gov/linux/slurm/
- SLURM e-mail list: slurm-dev@lists.llnl.gov.
- SLURM faq: https://computing.llnl.gov/linux/slurm/faq.html

Installing SLURM

The SLURM RPMs are installed by default on the head node:

```
# rpm -qa |grep -i slurm
slurm-plugins-1.3.13-sunhpc3
slurm-munge-1.3.13-sunhpc3
slurm-1.3.13-sunhpc3
```

The SLURM RPMs should be installed on diskful and diskless provisioned nodes by default. However, if the RPMs need to be installed, use `yum` to install them:

```
yum install slurm-1.3.13-sunhpc3
```

Creating a SLURM configuration file

A SLURM configuration file can be generated from the Sun HPC Software Management Database `gtdb`. After all the nodes have been successfully added to the cluster database, run the update command to update the configuration files:

```
# gtt config --update all
Updating config: cfagent
/var/lib/sunhpc/cfengine/var/cfengine/inputs/cfagent.conf: Wrote 34 lines
Updating config: cfservd
/var/lib/sunhpc/cfengine/var/cfengine/inputs/cfservd.conf: Wrote 36 lines
Updating config: cfupdate
/var/lib/sunhpc/cfengine/var/cfengine/inputs/update.conf: Wrote 84 lines
Updating config: cobbler
/var/lib/sunhpc/cfengine/tmp/cobbler.csv: Wrote 5 lines
Updating config: conman
/var/lib/sunhpc/cfengine/etc/conman.conf: Wrote 183 lines
Updating config: genders
/var/lib/sunhpc/cfengine/etc/genders: Wrote 6 lines
Updating config: hosts
/var/lib/sunhpc/cfengine/etc/hosts: Wrote 15 lines
Updating config: ntp
/var/lib/sunhpc/cfengine/etc/ntp.conf: Wrote 24 lines
```

```
Updating config: powerman
/var/lib/sunhpc/cfengine/etc/powerman/powerman.conf: Wrote 7 lines
Updating config: slurm
/var/lib/sunhpc/cfengine/etc/slurm/slurm.conf: Wrote 38 lines
```

The configuration file should look something like this:

```
####### BEGIN GTDB MANAGEMENT -- DO NOT EDIT BELOW THIS LINE #############
AuthType=auth/munge
CacheGroups=0
ClusterName=sunhpc
ControlMachine=headnode
CryptoType=crypto/munge
FastSchedule=1
InactiveLimit=0
JobAcctGatherType=jobacct_gather/none
JobCompLoc=/tmp/slurm_jobcomp.log
JobCompType=jobcomp/filetxt
KillWait=30
MinJobAge=300
MpiDefault=none
ProctrackType=proctrack/linuxproc
ReturnToService=1
SchedulerType=sched/backfill
SelectType=select/linear
SlurmUser=daemon
SlurmctldDebug=3
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmctldPort=6817
SlurmctldTimeout=300
SlurmdDebug=3
SlurmdPidFile=/var/run/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/tmp/slurmd
SlurmdTimeout=300
StateSaveLocation=/tmp
SwitchType=switch/none
Waittime=0

# COMPUTE NODES
NodeName=DEFAULT State=UNKNOWN ThreadsPerCore=1 CoresPerSocket=4
RealMemory=2007 Sockets=2
NodeName=node0001
NodeName=node0002

# PARTITIONS
PartitionName=DEFAULT
```

```
PartitionName=compute Nodes=node0001,node0002 Default=YES
######## END GTDB MANAGEMENT -- DO NOT EDIT ABOVE THIS LINE ##############
```

Note: If you edit `slurm.conf` by hand, be sure that no key values are duplicated in the `GTDB Management` section.

SLURM comes with a web-based tool that can be used to help write the configuration file. A copy of this tool can be found at: https://computing.llnl.gov/linux/slurm/configurator.html. When you provide the appropriate values, the tool will display a text file that can be saved in `/etc/slurm/slurm.conf`. Although the Sun HPC Software Management Tool generates the configuration file, the SLURM web-based configuration tool can be used to include an option not supported by the Sun HPC Software or to create a sample configuration file.

Some key values are:

`ControlMachine=hpc-x4600-2`	Name of the host on which the server daemon `slurmcltd` will run.
`ControlAddr=192.168.202.214`	IP address for the host on which the server daemon will run.
`AuthType=auth/munge`	If `AuthType` is set to `munge`, MUNGE will be used as the authentication service for all SLURM communications. MUNGE is installed as part of the Sun HPC Software.

`# COMPUTE NODE \`
`NodeName=cl10-[6-7] Procs=16 State=UNKNOWN`
OR
`NodeName=DEFAULT ThreadsPerCore=1 \`
`CoresPerSocket=4 RealMemory=16384 \`
`Sockets=2 State=UNKNOWN`
	Defines the two client nodes used for this example. `Procs` are the number processors on each node. `State` should be set to `UNKNOWN`. SLURM will update the state when the client daemons are started.

`PartitionName=debug Nodes=cl10-[6-7]`
`Default=YES MaxTime=INFINITE State=UP`
OR
`PartitionName=DEFAULT`
	Shows the partition to which the client nodes used for this example are assigned .

To find out about other configuration values, enter `man slurm.conf` or refer to the SLURM documentation.

Starting SLURM on clients

To start SLURM on one or more clients:

```
pdsh -w cl10-[6-7] /etc/init.d/slurm start
```

When troubleshooting SLURM , it may be helpful to start the `slurmd` daemon by hand on a client compute node.

```
/usr/sbin/slurmd -D -vvvvvvv
```

In this example, `-D` starts debugging and `-v` starts verbose output. Each `v` adds an extra level of verbose output with `-vvvvvvv` resulting in full debugging and verbose output.

If the clients and server are not on the same network, you may need to add a default route or an additional route on the client node using the `route add` command (see the `route` man page for more information).

Starting the main SLURM daemon

The main server daemon runs on the node set as the value of `ControlMachine` in the SLURM configuration file `/etc/slurm/slurm.conf`. To start the `slurmctld` daemon on the main server, enter:

```
/etc/init.d/slurm start
```

If troubleshooting by hand, you can start the daemon by entering:

```
/usr/sbin/slurmctld -D -vvvvvvv
```

After SLURM has started, you can verify that the SLURM subsystem is running using the `sinfo` command:

```
# sinfo
PARTITION AVAIL  TIMELIMIT NODES  STATE NODELIST
debug*       up   infinite     2   idle cl10-[6-7]
```

If the `sinfo` command reports that the partition or the nodes are down, SLURM may be having a communication problem. If you are using MUNGE or openSSL authentication for communications, make sure all clients and the server node are syncronized in time. This is usually accomplished using the Network Time Protocol operating system daemon `ntpd`.

Using SLURM

It is possible to use SLURM directly with the `srun` command. Important `srun` options are:

- `-n` – Number of tasks
- `-N` – Number of nodes to run on
- `-c` – Number of CPUs per task

Use `man srun` to obtain more information about `srun` options.

In this example, `srun` is used to run `./a.out` on two nodes with two tasks per node.

```
srun -N 2 -n 2 ./a.out
```

Note: `./a.out` must be on shared file system accessible by clients.

For more information about using SLURM, refer to the SLURM tutorial at:
https://computing.llnl.gov/tutorials/slurm/slurm.pdf

Making a batch file

Althought the `srun` command can be used on a multi-user system, it is usually preferable to submit a batch script. `slurm` batch scripts do not recognize all the `slurm` arguments, so it is necessary to pass `slurm` arguments to `slurm` outside the batch script. For example:

```
batch.slurm
#!/bin/bash
srun hostname
srun <path to file>/a.out
```

This job can be submitted using:

```
sbatch -n16 -J jobname -t 120 batch.slurm
```

Where:

- `-n` – Number of tasks
- `-J` – Name of the job
- `-t` – Maximum wall time in minutes

Note: The file containing the batch script `batch.slurm` must be on a shared file system accessible by clients.

The `-N` option has been included in the example below to show an alternate way the job can be submitted:

```
sbatch -N4 -n4 -J jobname -t 120 batch.slurm
```

In this example, `-N4` says to run 1 task per node (normally one processor) and `-n4` says to use 4 nodes in total. Thus, the scheduler will allocate 4x4 or 16 processors for the job.

To prevent sharing of a node's resources, the scheduler will always allocate an entire node.

Appendix A:
Cluster Inventory Example

The Sun HPC Software provides support for provisioning four types of client nodes using a Cobbler service on the head node:

- *Diskful Lustre client mode*. Runs an unpatched Red Hat Linux or SLES SP 2 kernel and a number of software packages on a local disk, such as an MPI program, SGE execution host program, Lustre client software, and InfiniBand software. A diskful Lustre client node generally serves as a compute node in the HPC cluster and has access to a Lustre file system.
- *Diskful Lustre server node.* Runs the kernel patched with Lustre server software. Although other software packages can be installed on a diskful Lustre server node, the major role of this node type is to serve as a metadata server (MDS) or object storage server (OSS) node in a Lustre file system.
- *Diskless Lustre client node.* Runs on a oneSIS image on the head node through an NFS mount. A diskless Lustre client node uses the same kernel as the head node. It generally serves as a compute node in the HPC cluster and has access to a Lustre file system.
- *Diskless Lustre Server Node.* Runs on a oneSIS image on the head node through an NFS mount. A diskless Lustre server node uses a Lustre patched kernel. It typically serves as a metadata server (MDS) or object storage server (OSS) node in a Lustre file system.

Table 1 and Table 2 provide example inventories for common cluster configurations. You will need to adapt the tables to your particular cluster configuration.

Table 1. Example inventory of a provisioning and general communication network

	Provisioning and General Communication Network			
Node ID (hostname on provisioning interface)	**Configuration**	**Role**	**Provisioning Interface MAC Address** (usually `eth0`)	**Provisioning Interface IP Address** (usually `eth0`)
mgmt1	Diskful	Management	00:14:4f:80:14:a0	10.1.80.1
login1	Diskful	Login node	00:14:4f:82:31:5e	10.1.80.2
login2	Diskful	Login node	00:14:4f:9e:a0:ce	10.1.80.3
dfmds01	Diskful	Lustre MDS	00:14:4f:45:26:e2	10.1.80.4
dfmds02	Diskful	Lustre MDS	00:14:4f:11:73:45	10.1.80.5
dfoss01	Diskful	Lustre OSS	00:14:4f:31:a0:5e	10.1.80.6
dfoss02	Diskful	Lustre OSS	00:14:4f:a7:30:9d	10.1.80.7
dflcn001	Diskful	Lustre client/ compute node	00:14:4f:ee:6f:45	10.1.80.8
dflcn002	Diskful	Lustre client/ compute node	00:14:4f:9e:3f:f5	10.1.80.9
dlmds01	Diskless	Lustre MDS	00:14:4f:45:26:d2	10.1.80.10
dlmds02	Diskless	Lustre MDS	00:14:4f:11:7e:4f	10.1.80.11
dloss01	Diskless	lustre OSS	00:14:4f:31:a0:ff	10.1.80.12
dloss02	Diskless	Lustre OSS	00:14:4f:a7:9f:9d	10.1.80.13
dllcn001	Diskless	Lustre client/ compute node	00:14:4f:9e:6f:4f	10.1.80.14
dllcn002	Diskless	Lustre client/ compute node	00:14:4f:1e:3e:f9	10.1.80.15

Table 2. Example inventory of Infiniband and management networks

	InfiniBand Network		***Management Network***		
Node ID (hostname on provisioning interface)	**IB Interface Hostname** (usually ib0)	**IB Interface IP Address** (usually ib0)	**Service Processor Hostname** (ILOM or Management Interface)	**Service Processor MAC Address** (ILOM or Management Interface)	**Service Processor IP Address** (ILOM or Management Interface)
mgmt1	mgmt1-ib0	10.13.80.1	mgmt1-sp	00:14:4f:f0:14:a0	10.2.80.1
login1	login1-ib0	10.13.80.2	login1-sp	00:14:4f:82:f1:5e	10.2.80.2
login2	login2-ib0	10.13.80.3	login2-sp	00:14:4f:9e:a0:3e	10.2.80.3
dfmds01	dfmds01-ib0	10.13.80.4	dfmds01-sp	00:14:4f:45:26:e6	10.2.80.4
dfmds02	dfmds02-ib0	10.13.80.5	dfmds02-sp	00:14:4f:11:73:4f	10.2.80.5
dfoss01	dfoss01-ib0	10.13.80.6	dfoss01-sp	00:14:4f:31:a0:5f	10.2.80.6
dfoss02	dfoss02-ib0	10.13.80.7	dfoss02-sp	00:14:4f:a7:30:9f	10.2.80.7
dflcn001	dflcn001-ib0	10.13.80.8	dflcn001-sp	00:14:4f:ee:6f:4f	10.2.80.8
dflcn002	dflcn002-ib0	10.13.80.9	dflcn002-sp	00:14:4f:9e:3f:fd	10.2.80.9
dlmds01	dlmds01-ib0	10.13.80.10	dlmds01-sp	00:14:4f:45:26:df	10.2.80.10
dlmds02	dlmds02-ib0	10.13.80.11	dlmds02-sp	00:14:4f:11:7e:7f	10.2.80.11
dloss01	dloss01-ib0	10.13.80.12	dloss01-sp	00:14:4f:31:a0:ef	10.2.80.12
dloss02	dloss02-ib0	10.13.80.13	dloss02-sp	00:14:4f:a7:9f:9e	10.2.80.13
dllcn001	dllcn001-ib0	10.13.80.14	dllcn001-sp	00:14:4f:9e:6f:9f	10.2.80.14
dllcn002	dllcn002-ib0	10.13.80.15	dllcn002-sp	00:14:4f:1e:3e:fe	10.2.80.14

Appendix B: *Using Boot Over IB to Deploy Diskless Clients*

This appendix describes how to use the Boot Over InfiniBand (BoIB) solution provided by Mellanox Technologies to deploy diskless clients. For general information about BoIB, refer to:

- *Booting Over InfiniBand for Consolidation Savings* (Sun BluePrint) http://wikis.sun.com/display/BluePrints/Booting+Over+InfiniBand+for+Consolidation+Savings
- *Boot over IB (BoIB) User's Manual* (Mellanox Technologies) http://www.mellanox.com/related-docs/prod_software/Boot-over-IB_User_Manual.pdf

Preparing the IB HCAs

Before booting the InfiniBand Host Channel Adapter (HCA) cards, you will usually need to update the HCA firmware. The HCA firmware can be updated using the Mellanox Firmware Tools.

To download the files needed to update the HCA firmware:

1. Install the Mellanox Firmware Tools if they are not already installed. Assuming the Sun HPC Software is available either through the online repository at `dlc.sun.com` or through a local mount of a Sun HPC Software DVD, enter:

```
# yum install mft-2.5.0-$(uname -r | sed -e “s/-/_/g”)
```

2. Download the Mellanox Technologies BoIB solution package from the Mellanox website:

```
# wget http://www.mellanox.com/downloads/Drivers/PXE/BoIB-2.0.000.tgz
# tar zxvf BoIB-2.0.000.tgz
```

3. Determine the HCA device name (for example, Mellanox Technologies MT25418) by entering:

```
# lspci |grep InfiniBand
```

4. Download the InfiniBand HCA firmware image for your HCA (for example, `fw-25408-rel.mlx`) from the Mellanox Technologies website at: http://www.mellanox.com/content/pages.php?pg=firmware_table_Sun

5. Locate the expansion ROM image for your HCA that was downloaded as part of the Mellanox Technologies BoIB solution package in Step 2 (for example, `CONNECTX_DDR_PORT1_ROM-1.0.0.rom`).

To burn the firmware to the HCA, complete the following steps:

1. Start the Mellanox Software Tools:

```
# mst start
```

2. Find the device to use:

```
# mst status| grep cr0
/dev/mst/mt25418_pci_cr0         - PCI direct access.
```

3. Determine the Board ID:

```
# flint -d /dev/mst/mt25418_pci_cr0 q | grep Board
```

4. Read the configuration from the HCA:

```
# flint -d /dev/mst/mt25418_pci_cr0 dc > SUN0070000001.ini
```

Note: Make sure the `.ini` file is not empty. If the configuration was not read from the HCA, you will be unable to burn the firmware. If this step fails, contact customer service for assistance.

5. Burn the firmware and expansion rom images:

```
# mlxburn -dev /dev/mst/mt25418_pci_cr0 -fw fw-25408-rel.mlx -exp_rom
CONNECTX_DDR_PORT1_ROM-1.0.0.rom -conf SUN0070000001.ini
```

6. Reboot the machine to activate the new firmware.

Notes:

- After the Mellanox Software Tools have been started, HCA device names can be listed using `mst status`.
- The firmware version of the HCA can be verified by entering:

```
flint -d /dev/mst/mt25418_pci_cr0 q
Image type:      ConnectX
FW Version:      2.6.0
Rom Info:        type=GPXE version=2.0.0 devid=25418
Device ID:       25418
Chip Revision:   A0
Description:     Node             Port1            Port2            \
   Sys image
GUIDs:           0003ba0001006d80 0003ba0001006d81 0003ba0001006d82 \
   0003ba0001006d83
MACs:                             0003ba006d81     0003ba006d82
```

```
Board ID:          (SUN0070000001)
VSD:
PSID:              SUN0070000001
```

- To check if the firmware provides the gPXE option, reboot the machine and press CTRL-P during the BIOS initialization on the console. A menu will appear that shows a gPXE option.

Configuring Cobbler for a diskless client

To set up a diskless client configuration, complete the steps below.

1. To prepare the Cobbler repository on the head node for provisioning the clients, enter:

```
# sunhpc_setup --profile=rhel5.3-onesis --diskless \
  --distro=rhel5.3-onesis --netif=ib0 -onesis-exclude=/root
```

   `--netif=ib0` indicates that InfiniBand is to be used

2. To add the diskless client to the cobbler configuration:

```
# cobbler system add --name=hpc-x4540-1 -interface=ib0 \
  --mac=00:03:ba:00:01:00:8d:fd -ip=192.168.203.215 \
  --subnet=255.255.255.0 --hostname=hpc-x4540-1 \
  --profile=rhel5.3-onesis --dns-name=hpc-x4540-1-ib
# cobbler sync
```

Note: InfiniBand HCAs use a GUID rather than a MAC address as a `dhcp` client identifier. For example, the `dhcp` client identifier for `ib0` in the example above is `GUID 0003ba0001008dfd`. The GUID must be converted to a MAC address (in the example, `00:03:ba:00:01:00:8d:fd`) to add it to the cobbler system.

An entry for this system will appear in `/etc/dhpcd.conf` on the head node.

```
    host hpc-x4540-1-ib {
        option dhcp-client-identifier = 00:03:ba:00:01:00:8d:fd;

            fixed-address 192.168.203.215;
            option subnet-mask 255.255.255.0;
            filename "/pxelinux.0";
            next-server 192.168.203.216;

    }
```

Note: before booting the diskless clients over InfiniBand, make sure `openibd` and `opensmd` are disabled in the oneSIS image by entering:

```
# chroot /var/lib/oneSIS/image/rhel5.3-onesis
# chkconfig --del openibd
# chkconfig --del opensmd
# exit
```

Booting the client

Once the client is running on the new firmware, gPXE will appear as a boot option in the BIOS. Boot the client using one of these methods:

- Boot the client manually from the BIOS.
- Use FreeIPMI to set up the client to boot using PXE. Wait until all the Ethernet interfaces fail to boot, after which gPXE will boot.

```
# ipmi-chassis-config -h [client node name or IP for ILOM] -u root \
  -p [Root password] -e "Chassis_Boot_Flags:Boot_Device=PXE" –commit
# ipmipower -h [client node name or IP for ILOM] -u root \
  -p [Root password] –reset
```

For either option, the output displayed on the console is shown below.

```
Mellanox ConnectX Boot over IB v2.0.000
gPXE 0.9.6+ -- Open Source Boot Firmware -- http://etherboot.org
net0: 00:03:ba:00:01:00:8d:fd on PCI81:00.0 (open)
  [Link:down, TX:0 TXE:0 RX:0 RXE:0]
Waiting for link-up on net0... ok
DHCP (net0 00:03:ba:00:01:00:6e:21).... ok
net0: 192.168.203.215/255.255.255.0 gw 192.168.203.216
Booting from filename "/pxelinux.0"
tftp://192.168.203.216//pxelinux.0... ok

PXELINUX 3.11 2005-09-02  Copyright (C) 1994-2005 H. Peter Anvin
UNDI data segment at:   00098F10
UNDI data segment size: 28F0
UNDI code segment at:   00098910
UNDI code segment size: 05FA
PXE entry point found (we hope) at 9891:01D9
My IP address seems to be C0A8CBD7 192.168.203.215
ip=192.168.203.215:192.168.203.216:192.168.203.216:255.255.255.0
TFTP prefix: /
Trying to load: pxelinux.cfg/rhel5.3-onesis
Loading /images/rhel5.3-onesis/vmlinuz-2.6.18-128.el5........
```

Appendix C:

Sun HPC Software Components

Component	***Installed by default on:***	***Description***
Cfengine	All nodes	Cfengine is an automated suite of programs for configuring and maintaining Unix-like computers. (http://www.cfengine.org)
Cobbler	Head node	Cobbler is a Linux provisioning server that provides tools for automating software installation on large numbers of Linux systems, including PXE configurations and boots, re-installation, and virtualization. (https://fedorahosted.org/cobbler).
ConMan	Head node	ConMan is a serial console management program designed to support a large number of console devices and simultaneous users. (http://home.gna.org/conman/).
env-switcher	All nodes	Environment Switcher (env-switcher) is a thin layer on top of the modules package that allows users to manipulate the environment that is loaded for all shells (including non-interactive remote shells) without manually editing their startup dot files. (http://sourceforge.net/projects/env-switcher/)
fakeroot	Not installed	fakeroot allows a command to be run in a simulated root environment to enable file manipulation through the use of features of LD_PRELOAD and SYSV IPC or TCP. (http://fakeroot.alioth.debian.org/)
FreeIPMI	All nodes	FreeIPMI is a collection of Intelligent Platform Management Interface (IPMI) system software that provides in-band and out-of-band software and a development library conforming to the Intelligent Platform Management Interface (IPMI v1.5 and v2.0) standards. (http://www.gnu.org/software/freeipmi/
Ganglia	Head node (`gmond` on all nodes)	Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and grids. (http://ganglia.info/)
Genders	All nodes	Genders is a static cluster configuration database used for cluster configuration management. (https://computing.llnl.gov/linux/genders.html)
Git	All nodes	Git is an open source version control system designed to handle very large projects with speed and efficiency, but just as well suited for small personal repositories. (http://git.or.cz)

gtdb	Head node	The Sun HPC Software Management Database (gtdb) is designed to automatically configure client-server applications to work out of the box after client, login and lustre server nodes are provisioned and installed. See Sun HPC Software Management Database and Tools Overview.
Heartbeat	Lustre servers only	Heartbeat is a GPL-licensed portable cluster management program for high-availability clustering. (http://www.linux-ha.org/Heartbeat)
HPCC Bench Suite	All nodes	HPC Challenge is a collection of benchmarks for measuring various aspects of system performance, such as flop/s, sustainable memory bandwidth, memory read/write rates, network bandwidth, and latency for parallel machines. (http://icl.cs.utk.edu/hpcc/)
IOKit (Lustre)	Not installed	The Lustre I/O kit is a collection of benchmark tools for a Lustre cluster. (http://manual.lustre.org/manual/LustreManual16_HTML/LustreIOKit.html)
IOR	All nodes	Interleaved-Or-Random Filesystem Benchmarking software is used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces. (http://sourceforge.net/projects/ior-sio)
IPMItool	All nodes	IPMItool is a utility for managing and configuring devices that support the Intelligent Platform Management Interface (IPMI) version 1.5 and version 2.0 specifications. (http://ipmitool.sourceforge.net/)
lshw	All nodes	lshw (Hardware Lister) provides detailed information on the hardware configuration of a machine, such as exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache configuration, and bus speed, on DMI-capable x86 or EFI (IA-64) systems and on some PowerPC machines. (http://ezix.org/project/wiki/HardwareLiSter)
Lustre	All nodes	Lustre is a scalable, secure, robust, highly-available cluster file system designed, developed and maintained by Sun Microsystems, Inc. (http://wiki.lustre.org/index.php?title=Main_Page)
Mellanox Firmware Tools	All nodes	Mellanox Firmware Tools (MFT) is a package of firmware management tools for InfiniBand nodes. (http://www.mellanox.com/content/pages.php?pg=management_tools&menu_section=34)
Modules	All nodes	The Environment Modules package provides for the dynamic modification of a user's environment using module files. (http://modules.sourceforge.net/)

MUNGE	All nodes	MUNGE (MUNGE Uid 'N' Gid Emporium) is an authentication service for creating and validating credentials designed to be highly scalable for use in an HPC cluster environment. (http://home.gna.org/munge/)
MVAPICH	Not installed	MVAPICH is a MPI-1 implementation based on MPICH and MVICH that supports a variety of transport interfaces on a wide range of platforms. The name comes from the abbreviation of MPI-1 over OpenFabrics/Gen2, OpenFabrics/Gen2-UD, uDAPL, InfiniPath, VAPI and TCP/IP. (http://mvapich.cse.ohio-state.edu/index.shtml)
MVAPICH2	Not installed	MVAPICH2 is an MPI-2 implementation based on MPICH2 and MVICH. It backward supports all MPI-1 features. It supports several transport interfaces including OpenFabrics-IB, OpenFabrics-iWARP, uDAPL, and TCP/IP. (http://mvapich.cse.ohio-state.edu/index.shtml)
Nagios	Head node	Nagios is a system and network monitoring application that monitors host resources and network services and provides alerts when problems occur or are resolved. (http://www.nagios.org)
NetPIPE	All nodes	Network Protocol Independent Performance Evaluator (NetPIPE) is a protocol independent performance tool that visually represents the network performance under a variety of conditions. (http://www.scl.ameslab.gov/netpipe/)
OFED	All nodes	The OpenFabrics Enterprise Distribution (OFED) is a validated version of the open-source OpenFabrics software stack that supports server and storage clustering and grid connectivity using RDMA-based InfiniBand and iWARP fabrics in a Linux environment. (http://www.openfabrics.org)
oneSIS	Head node	oneSIS is an open-source software tool for administering systems in a large-scale, Linux-based cluster environment. The default oneSIS configuration that results from building the head node is used to begin provisioning nodes in the cluster as diskless clients. (http://www.onesis.org)
OpenSM	All nodes	OpenSM is an InfiniBand compliant subnet manager and administration tool that runs, on top of OpenIB. (https://wiki.openfabrics.org/tiki-index.php?page=OpenSM)
pdsh	All nodes	Parallel Distributed Shell (pdsh) is an efficient, multi-threaded remote shell client that executes commands on multiple remote hosts in parallel. pdsh implements dynamically loadable modules for extended functionality such as new remote shell services and remote host selection. (http://sourceforge.net/projects/pdsh/)

perfctr	Not installed	The perfctr driver enables the use of a Performance Application Programming Interface (PAPI) to collect low level performance metrics. (http://perfctr.sourceforge.net/http://sourceforge.net/projects/perfctr/)
Powerman	Head node	PowerMan is a tool for manipulating remote power control (RPC) devices from a central location. (http://powerman.sourceforge.net/)
RRDtool	Head node	Round Robin Database tool stores and retrieves data from Round Robin Databases (RRDs). (http://oss.oetiker.ch/rrdtool/)
SLURM	All nodes	The Simple Linux Utility for Resource Management (SLURM) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. (https://computing.llnl.gov/linux/slurm/)
Sun Grid Engine	Not installed	Sun Grid Engine (SGE) is an open source batch-queuing system, supported by Sun Microsystems. SGE accepts, schedules, dispatches, and manages the remote execution of large numbers of standalone, parallel or interactive user jobs in a cluster system. It also manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses. (http://www.sun.com/software/sge/)
Sun HPC ClusterTools	All nodes (except Lustre servers)	Sun HPC ClusterTools 8.1 is an integrated toolkit based on Open MPI 1.3 that offers a comprehensive set of capabilities for parallel computing. Sun HPC ClusterTools allows developers to create and tune Message Passing Interface (MPI) applications running on high performance clusters. (http://www.sun.com/software/products/clustertools/)

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO

MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA *
Phone: 1-650-960-1300 or 1-800-555-9SUN (9786) Web: sun.com