RedHawk Linux® Cluster Manager User's Guide

concurrent

0898016-110
July 2007

To report an error or comment on a specific portion of the manual, photocopy the page in question and mark the correction or comment on the copy. Mail the copy (and any additional comments) to Concurrent Computer Corporation, 2881 Gateway Drive, Pompano Beach, Florida, 33069. Mark the envelope **"Attention: Publications Department**." This publication may not be reproduced for any other reason in any form without written permission of the publisher.

Printed in U. S. A.

Revision History:

Date	Level	Effective With
September 2006	000	RedHawk Linux Release 4.1.*x*
December 2006	100	RedHawk Linux Release 4.1.*x*
July 2007	110	RedHawk Linux Release 4.2

Contents

Chapter 2 Grid Engine Software

Illustrations

Tables

Preface

Scope of Manual

This manual is intended for users responsible for the installation and use of the RedHawk Linux Cluster Manager product, Concurrent model number WA9017-L.

Structure of Manual

This guide consists of the following sections:

- Chapter 1, *Getting Started*, provides an overview of the RedHawk Linux Cluster Manager product and detailed procedures for installing and configuring Cluster Manager on the master system and cluster nodes.
- Chapter 2, *Grid Engine Software*, describes the Grid Engine software used to manage resources and submit jobs and provides instructions for configuring your cluster.
- Appendix A, *Node Information Worksheet*, is an easy-to-use worksheet for recording information needed when configuring the cluster.
- The *Index* contains an alphabetical reference to key terms and concepts and the pages where they occur in the text.

Syntax Notation

The following notation is used throughout this manual:

italic	Books, reference cards, and items that the user must specify appear in *italic* type. Special terms may also appear in *italic*.
`list bold`	User input appears in **`list bold`** type and must be entered exactly as shown. Names of directories, files, commands, options and man page references also appear in **`list bold`** type.
`list`	Operating system and program output such as prompts, messages and listings of files and programs appears in `list` type.
`[]`	Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify these options or arguments.
hypertext links	When viewing this document online, clicking on chapter, section, figure, table and page number references will display the corresponding text. Clicking on Internet URLs provided in **blue** type will launch your web browser and display the web site. Clicking on publication names and numbers in **red** type will display the corresponding manual PDF, if accessible.

Related Publications

The following table lists related documentation. Click on the red entry to display the document PDF. RedHawk documents are also available by clicking on the "Documents" icon on the desktop and from Concurrent's web site at **www.ccur.com**.

RedHawk Linux Operating System Documentation	**Pub No.**
RedHawk Linux Release Notes Version x.x	0898003
RedHawk Linux User's Guide	0898004
RedHawk Linux Frequency-Based Scheduler (FBS) User's Guide	0898005
Real-Time Clock and Interrupt Module (RCIM) PCI Form Factor User's Guide	0898007
iHawk Optimization Guide	0898011
RedHawk Linux FAQ	N/A
Partner Documentation	**Pub No.**
Sun N1 Grid Engine 6.1 Installation Guide	820-0697
Sun N1 Grid Engine 6.1 Administration Guide	820-0698
Sun N1 Grid Engine 6.1 User's Guide	820-0699
Sun N1 Grid Engine 6.1 Release Notes	820-0700

where *x.x* = release version

For more information about Grid Engine, see Chapter 2.

1
Getting Started

1
Getting Started

This chapter describes RedHawk Linux Cluster Manager and provides procedures for installing and configuring the product.

What is RedHawk Cluster Manager?

RedHawk™ Linux® Cluster Manager contains everything needed to install and configure Concurrent's iHawk™ systems into a highly integrated, high performance computer cluster and the user interface to effectively utilize the cluster's full capabilities.

A cluster contains a master host and multiple nodes. Each node contains its own CPU, memory, operating system and I/O subsystem and is capable of communicating with each other. Clusters are used to run parallel programs for time-intensive computations, such as simulations and other CPU-intensive programs that would take an inordinate amount of time to run on regular hardware.

Nodes can contain a hard disk or can be diskless. Any Concurrent iHawk system can be configured as a node in a cluster.

Cluster Manager includes Grid Engine, an open source batch-queuing system, developed by Sun Microsystems, that manages and schedules the allocation of distributed resources such as processors, memory, disk-space, and software licenses. Grid Engine is designed for use on computer clusters and is responsible for accepting, scheduling, dispatching and managing the remote execution of large numbers of standalone, parallel or interactive user jobs.

Cluster Manager is an optional product that can be installed on systems running the corresponding version of the RedHawk Linux operating system; for example, Cluster Manager 4.2 on a RedHawk 4.2 system.

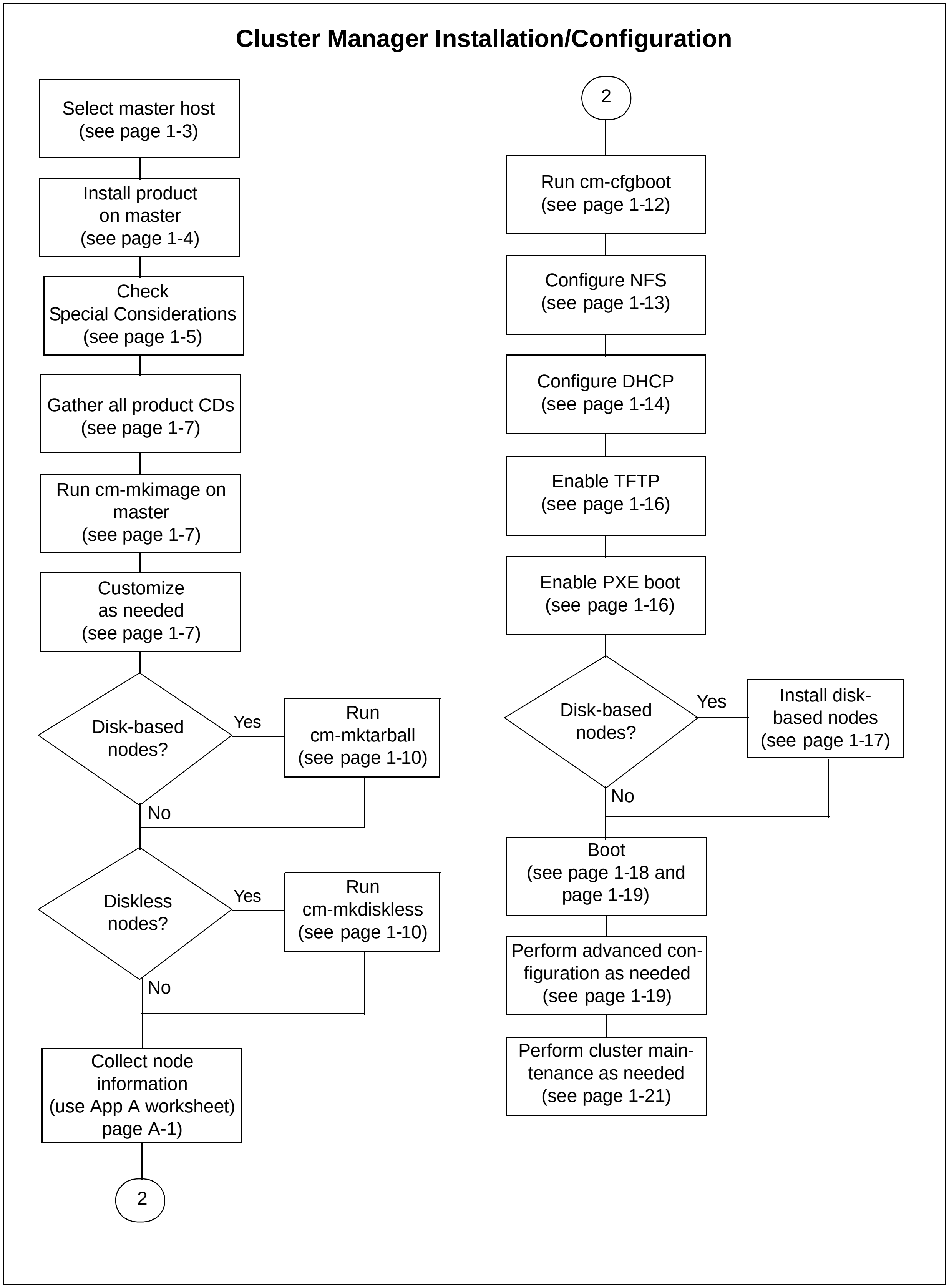
Note that Cluster Manager is based on the open source YACI (Yet Another Cluster Installer) project (pronounced Yak-E) developed at Lawrence Livermore National Laboratories. The string "yaci" is mentioned in various places within this document and while running the Cluster Manager installation and configuration programs. More information about YACI is available at the official YACI web site:
http://www.llnl.gov/linux/yaci/yaci.html

Procedure Flow Chart

Figure 1-1 is a flow chart that illustrates the procedure for installing and configuring your cluster with RedHawk Cluster Manager.

The page numbers in the flow chart are hyperlinks to the appropriate sections where complete information about the step can be found.

Figure 1-1 Cluster Manager Installation/Configuration Flow Chart



Installing Cluster Manager

Prerequisites

Hardware

- Cluster nodes may be any Concurrent iHawk system with at least one NIC that supports PXE booting.

  NOTE: Some BIOSes do not provide an option to boot with PXE. You may use the etherboot utility to work around this. Concurrent does not support this configuration.

- Cluster nodes may contain a local hard disk or may be diskless.
- The cluster master requires that **`/tftpboot/yaci`** have a minimum of 11 GB of free space for creating the cluster file system image.

  NOTE: It is recommended (but not required) that all nodes in the cluster have matching hardware.

Software

- RedHawk Linux 4.2 or later

Before Installing

Before installing the software, you should identify one system to be the cluster "master". It is on this system that you will install the Cluster Manager software. This system will be the master for all of the cluster nodes. It should be a separate dedicated system that can also be used as the Grid Engine master system. If you build a cluster containing mixed architectures (i386 and x86_64), each architecture should have its own master host.

The cluster master requires that **`/tftpboot/yaci`** have a minimum of 11 GB of free space for creating the cluster file system image. This differs from the standard RedHawk configuration. If RedHawk is already installed on the master host, a new disk should be added and mounted under **`/tftpboot/yaci`** ***before*** installing Cluster Manager.

Installing the Product CD

Follow these steps to install Cluster Manager on the master host.

1. On the system designated as the cluster master with RedHawk Linux 4.2 or later running, insert the disc labeled "RedHawk Linux Cluster Manager 4.2" appropriate to your system's architecture and insert it into the CD-ROM drive.

2. To mount the cdrom device, execute the following command:

   NOTE: **`/media/cdrom`** is used in the examples that follow. Depending on the type of drive attached to your system, the actual mount point may differ. Check **`/etc/fstab`** for the correct mount point.

   ```
   # mount /media/cdrom
   ```

3. To install, execute the following commands:

   ```
   # cd /media/cdrom
   # ./install-cm
   ```

4. When the installation completes, execute the following commands:

   ```
   # cd /
   # umount /media/cdrom
   # eject
   ```

5. Remove the disc from the CD-ROM drive and store.

Product Updates

As Cluster Manager updates are issued, they will be made available for downloading from Concurrent's RedHawk Updates website, **http://redhawk.ccur.com**.

Configuring the Cluster Manager Master System

Configuration Summary

Setting up a cluster involves the following steps, which are described in detail in the sections that follow:

1. Create the file system image which will be used by each of the cluster nodes.

2. Configure various network services (e.g., PXE, DHCP, NFS) on the master system for all of the cluster nodes.

3. Enable TFTP on the master system.

Creating a Cluster File System Image

Creating a cluster file system image involves the following steps:

- run **cm-mkimage** to create a file system image directory
- customize the file system image directory (if desired)
- run **cm-mktarball** to create a compressed tar file used to install disk-based nodes
- run **cm-mkdiskless** to create a root file system ramdisk used to boot diskless nodes

When you run **cm-mkimage**, you are effectively performing a full installation of RedHawk within the cluster's image directory. The installation almost exactly mirrors the process of installing RedHawk onto an actual iHawk system. As such, you should be somewhat familiar with the RedHawk installation process (see the *RedHawk Linux Release Notes* for more information).

Running **cm-mktarball** is only necessary if you plan to install cluster nodes with local hard disks.

Running **cm-mkdiskless** is only necessary if you plan to boot diskless cluster nodes.

The minimum disk space requirements in **/tftpboot/yaci** are as follows:

- **cm-mkimage** 8.2 GB
- **cm-mktarball** 2.8 GB
- **cm-mkdiskless** 256 MB

The actual disk space requirements may increase depending on what software you decide to install in the cluster file system image.

Before You Begin: Special Considerations

The default settings for Cluster Manager should be acceptable for most cluster installations. However, the following sections describe areas that may need site-specific modification. If this is the case for your site, these need to be addressed before running **cm-mkimage**.

For additional information, refer to the configuration files and scripts in the **/tftpboot/yaci/etc** and **/tftpboot/yaci/scripts** directories on the master system.

Node Types

cm-mkimage, **cm-mktarball**, and **cm-mkdiskless** take a 'type' argument. The type argument is optional and will default to 'redhawk' if unspecified. Generally, you should not need to specify a type name explicitly; only specify a type name if you are creating a cluster with multiple node types (see "Multiple Node Types" on page 1-19).

Disk Partitioning

Disk partitioning applies only to disk-based nodes. The disk partitioning of the hard disks on each of the nodes is controlled by the **/tftpboot/yaci/etc/***type***/partition_list** file on the master system. By default, the file **/tftpboot/yaci/etc/partition_list** is used and contains the following settings:

```
#Device MountPoint      Format  SizeMB  Bootable
sda     /boot           ext3    512     *
sda     /               ext3    16384
sda     swap            swap    4096
sda     /home           ext3    rest
```

If you wish to change the default, copy **partition_list** to *type***/partition_list** and edit this new file before running **cm-mkimage**. **cm-mkimage** will copy **partition_list** to *type***/partition_list** (and use the defaults) only if *type***/partition_list** does not exist.

The contents should be fairly self-explanatory. Note that you must ***not*** specify the disk partition number in the device column; that is, use "sda" and not "sda1". The YACI installer will automatically determine the optimal physical partition mapping required.

You are free to modify this file as needed by the requirements of your specific cluster, however be sure to use "sd*" style device names for SCSI and SATA disks and "hd*" style device names for IDE disks.

For more information on disk partitioning, see the **fdisk(8)**, **sfdisk(8)** and **fstab(5)** man pages.

Variables

A few default parameters for the nodes may be changed prior to running **cm-mkimage** using the following variables:

USE_SERIAL — specifies whether the serial console or directly connected console is used as the console. The default is the serial port and baud rate specified by the SERIAL_PORT and BAUD variables, respectively. To use a directly connected console (tty0), set USE_SERIAL to null (="").

SERIAL_PORT — specifies the serial port of the serial console. The default is ttyS0. Not used if USE_SERIAL="".

BAUD — specifies the baud rate of the serial console. The default is 115200. Not used if USE_SERIAL="".

DHCP_DEVICE — specifies the Ethernet device used to broadcast DHCP requests. The default is eth0.

EXTRA_RAMDISK_MB — additional space (in megabytes) to allocate in diskless ramdisk.

STATIC_NETWORK — static configuration of the hostname and booting network interface on disk-based nodes when they are installed. The default is to use DHCP each time the nodes boot.

If you wish to change these defaults, you may set shell variables in the file **/tftpboot/yaci/etc/***type***/variables** before running **cm-mkimage**. Below are some examples:

```
USE_SERIAL="no"
BAUD="9600"
SERIAL_PORT="ttyS1"
DHCP_DEVICE="eth1"
EXTRA_RAMDISK_MB=50
STATIC_NETWORK="yes"
```

Building the Cluster File System

The **cm-mkimage** script prompts you to insert several CDs during the installation. These CDs were supplied with your original iHawk system or in a later optional purchase.

The CDs that are required are:

- Red Hat Enterprise Linux 4.0 with Update 4 Install Discs 1-5
- Red Hat Enterprise Linux 4.0 Updates
- RedHawk Linux 4.2 OS

The following CDs are optional products that you may also choose to install:

- RedHawk Linux 4.2 Documentation
- RedHawk Linux 4.2 Cluster Manager (needed only if SGE will be used on the nodes)
- RedHawk Linux 4.2 Frequency-Based Scheduler
- RedHawk Linux 4.2 PCI-to-VME Bridge Library
- NightStar Tools for RedHawk Linux

To build the cluster file system, invoke the following command as root:

`#` **cm-mkimage** [*type*]

Insert the CDs as requested and follow the on-screen instructions.

After you have completed the installation of the above CDs, **cm-mkimage** will prompt you whether you wish to install other packages into the cluster file system image. If you choose to do additional installations, you will be placed into a configuration shell where you may install other CDs or download and install updates.

Additional customization, including software installation, may be done at any time. The section “Customizing the Cluster File System” discusses this in more detail.

When **cm-mkimage** is finished, the cluster file system image will be in the directory **/tftpboot/images/***type*.

Customizing the Cluster File System

A cluster file system may be customized by modifying the files in the **/tftpboot/images/***type* directory. This may include manually editing or overwriting configuration files as well as installing additional software. This section discusses some common customizations you may wish to do.

Users and Groups

cm-mkimage automatically copies the **/etc/passwd**, **/etc/shadow**, and **/etc/group** files from the master’s root file system to the cluster file system. You may wish to edit or overwrite these files.

Time Zone

cm-mkimage automatically copies **/etc/sysconfig/clock** and **/etc/localtime** from the master's root file system to the cluster file system. You may wish to edit or overwrite this file.

Default Run Level

By default, cluster nodes will boot to run level 3. To change this, edit **/etc/inittab** in the cluster file system directory.

Default Kernel

cm-mkimage configures **/boot/grub/grub.conf** to boot the RedHawk trace kernel by default on disk-based nodes. Edit this file to change the default kernel that is booted on disk-based nodes. See "Kernel Selection" on page 1-20 for more information.

Network Configuration

cm-mkimage configures the cluster file system so that networking is entirely configured using the DHCP protocol when a cluster node boots. (See "Configuring DHCP" on page 1-14 to configure the DHCP server on the master host.)

To configure a static **/etc/hosts** file, edit or overwrite this file under the cluster file system directory.

To statically configure DNS, create the file **/etc/resolv.conf** under the cluster file system directory and configure it appropriately.

To statically configure a default gateway, add the following line to the file **/etc/sysconfig/network** under the cluster file system directory:

```
GATEWAY=nnn.nnn.nnn.nnn
```

The network interface used to install and boot cluster nodes will be configured using the DHCP protocol. To configure additional network interfaces with DHCP, create the file **/etc/sysconfig/network-scripts/ifcfg-eth***N* (where *N* is the appropriate interface number) to contain:

```
DEVICE=ethN
BOOTPROTO=dhcp
```

It is not possible to statically configure network interfaces on diskless nodes, but this is possible on disk-based nodes once the node is installed. To statically configure network interfaces on disk-based nodes, edit the **/etc/sysconfig/network-scripts/ifcfg-eth***N* files on the node's disk after it is installed.

Installed Software

Software may be installed, removed, and updated in the cluster file system directory by using the **cm-chroot** command. This command runs a shell with the root directory being the cluster image directory. All changes made to system files (including software installed or removed) will be done in the cluster image directory only. The master's root file system will not be affected.

cm-chroot must be run as root and has the following usage:

cm-chroot [-x *command*] [*type*]

The default type is 'redhawk'. If a command is given with the **-x** option, that command is executed; otherwise, an interactive bash shell is started.

The following example demonstrates using a software CD to install software in the 'redhawk' cluster image directory:

```
# cm-chroot
redhawk-cluster-image# mount /dev/cdrom /media/cdrom
redhawk-cluster-image# cd /media/cdrom
```

(follow CD-specific instructions)

```
redhawk-cluster-image# umount /dev/cdrom
redhawk-cluster-image# exit
```

Running X Applications

Running X applications under **cm-chroot** requires the following special X configuration on the master host:

1. Edit **/etc/X11/gdm/gdm.conf** and ensure that it has the following line:

   ```
   DisallowTCP=false
   ```

2. Restart the X server.

3. Run the command:

   ```
   $ xhost +
   ```

Updating Software with NUU

Concurrent's Network Update Utility (NUU) is a graphical user interface which allows a user to install, update and remove RPM packages on a system. The packages are downloaded from a remote yum repository. For more information about NUU, see **http://redhawk.ccur.com/nuu**.

NUU may be used to maintain the software installed in a cluster image directory. To do so, perform the following configuration steps:

1. Configure the master host to run X applications (i.e. NUU) under **cm-chroot** (see "Running X Applications" above).

2. Ensure that **/etc/resolv.conf** is configured correctly under the cluster image directory so that NUU can resolve external domain names. One way to do this is to copy **/etc/resolv.conf** from the master host. For example:

   ```
   # cp /etc/resolv.conf /tftpboot/yaci/images/redhawk/etc
   ```

Once configuration is complete, you may use **cm-chroot** to run **nuu**. For example:

```
# cm-chroot -x nuu
```

Building a Compressed Tar File for Disk-Based Nodes

Before installing disk-based cluster nodes, you must first create a compressed tar file of the cluster file system image. This tar file will be used to image the hard disk of disk-based cluster nodes.

To create the compressed tar file, invoke the following command as root:

cm-mktarball [*type*]

When **cm-mktarball** is finished, the cluster file system image will be compressed and placed in the **/tftpboot/tarfiles/***type***.tgz** file where it will be made available to install on nodes with hard disks.

Building a Ramdisk Image for Diskless Nodes

In order to boot diskless nodes, you must first create a ramdisk image containing the root file system used by diskless nodes. Other file systems will be mounted from the master host over NFS once a diskless node is booted.

To create the ramdisk image, invoke the following command as root:

$ **cm-mkdiskless** [*type*]

When **cm-mkdiskless** is finished, the ramdisk image will be compressed and placed in the **/tftpboot/images/***type***-ramdisk.gz** file where it will be made available to diskless nodes.

Configuring the Cluster Network Using cm-cfgboot

The **cm-cfgboot** utility automates much of the network and boot configuration for your cluster. DHCP and NFS must be configured manually, but **cm-cfgboot** provides guidance in these areas. The following sections provide details.

Collecting Node Information

Before running **cm-cfgboot**, use the **Node Information Worksheet** provided in Appendix A to record the following information.

1. Collect the following information for the entire cluster:
    - The subnet to use for the cluster (e.g. 192.168.1.0)
    - The netmask to use for the cluster (e.g. 255.255.255.0)
    - Broadcast address (e.g. 192.168.1.255)
    - Default router(s) (if any)
    - Domain name server(s) and domain name (if any)

    Note that a cluster can exist on a network that includes other systems that are not in the cluster.

2. For each node in the cluster, collect the following information:

- MAC address of the PXE-capable Ethernet controller in the node
- IP address to assign to the node
- Hostname to assign to the node

3. If you choose to have more than one node type in your cluster, establish that scheme at this time. See “Multiple Node Types” on page 1-19 for more information.

Running cm-cfgboot

`cm-cfgboot` can be used to add nodes to a cluster, delete a node, change a node’s configuration or display the current configuration.

Syntax

`cm-cfgboot` [**`--help`**|**`-h`**] [**`--list`**|**`-l`**] [**`--delete`**|**`-d`**] [*nodename*]

Options

`--help`
`-h` displays help

`--list`
`-l` displays the current configuration for the cluster, including the names, MAC addresses, images and boot methods for all nodes

`--delete`
`-d` deletes the specified node from the configuration and updates the network files

nodename the node on which **`cm-cfgboot`** is to operate. If no *nodename* is specified, a list of defined nodes is provided for selection.

Invoke **`cm-cfgboot`** for each node you wish to add to the cluster:

`#` **`cm-cfgboot`** *nodename*

`cm-cfgboot` will prompt you for the following:

- MAC address of this node’s PXE-enabled NIC. For more information about MAC addresses, see the section “The MAC Information File” on page 1-12.
- the cluster image to be used for this node (a list of available images is supplied). Cluster images are described in the section “Creating a Cluster File System Image” on page 1-5.
- the method for booting this node (available methods are supplied). Boot methods are described in the section “Booting Cluster Nodes” on page 1-16.

The MAC Information File

When adding a node to a cluster, **cm-cfgboot** prompts you for the MAC address of the node's PXE-enabled NIC. **cm-cfgboot** reads and writes into the **/tftpboot/yaci/etc/MAC.info** file for all nodes; the user does not need to edit this file.

MAC.info contains the mapping of MAC addresses to host names. The MAC addresses can be obtained by entering the system's BIOS or, if the cluster node has already been loaded, by running **ifconfig -a** .

Mapping entries in **MAC.info** take the following form:

hostname MAC type

For example:

```
node1       00:0D:56:BA:CE:CF       redhawk
node2       00:0D:56:BA:CE:D2       redhawk
node3       00:0D:56:BA:CE:D4       redhawk
...
```

The *type* field is normally 'redhawk', though for advanced cluster installations with more than one node type defined it may be something other than 'redhawk' (see "Multiple Node Types" on page 1-19).

cm-cfgboot automatically updates **MAC.info**, backing up any previous version to **MAC.info.bak** first.

PXE Configuration

cm-cfgboot configures PXE automatically to instruct all nodes what to do when they boot.

Once the cluster file system image is created, the following configuration files will reside in the **/tftpboot/yaci/pxelinux.cfg** directory. For the 'redhawk' node type these files would be:

redhawk.install	instructs a node to (re)install its local hard disk
redhawk.local	instructs a node to boot from its local hard disk
redhawk.diskless	instructs a node to boot diskless

To determine which nodes perform which of the above actions, symbolic links are created to these files by **cm-cfgboot** according to the file naming rules used by PXE-linux. These rules are:

- It will search for a config file using its own IP address in upper case hexadecimal; e.g., 192.168.1.1 -> C0A80101
- If that file is not found, it will repeatedly remove one hex digit and try again.
- If that fails, it will look for a file named **default**.

As an example, if the IP address is 192.168.1.1 it will try (in order):

```
pxelinux.cfg/C0A80101
pxelinux.cfg/C0A8010
pxelinux.cfg/C0A801
```

```
pxelinux.cfg/C0A80
pxelinux.cfg/C0A8
pxelinux.cfg/C0A
pxelinux.cfg/C0
pxelinux.cfg/C
pxelinux.cfg/default
```

The **gethostbyname** program included with Cluster Manager returns the IP addresses of nodes and converts them to hexadecimal. Here are some examples:

```
$ gethostbyname node1
192.168.1.1
$ gethostbyname -x node1
C0A80101
$ gethostbyname -x 192.168.1.1
C0A80101
```

Examples

The following symbolic link in the **pxelinux.cfg** directory boot all nodes diskless:

```
default -> redhawk.diskless
```

The following symbolic link in the **pxelinux.cfg** directory installs the local hard disk on all nodes:

```
default -> redhawk.install
```

Note that once a node installs its local hard disk, a symbolic link is automatically created to boot that system from its local disk. For example, after a few node installations, your **pxelinux.cfg** directory may look like:

```
default -> redhawk.install
C0A80101 -> redhawk.local
C0A80102 -> redhawk.local
C0A80103 -> redhawk.local
```

Configuring NFS

NFS must be configured manually. This is accomplished through the **/etc/exports** file.

cm-cfgboot provides instructions for modifying **/etc/exports** after a node is added, deleted or modified.

To configure **/etc/exports** for the cluster, perform the following steps:

1. Edit **/etc/exports** on the master system. Add an entry for each node on the system. For example:

```
/tftpboot/yaci node1(rw,no_root_squash,sync)
/tftpboot/yaci node2(rw,no_root_squash,sync)
/tftpboot/yaci node3(rw,no_root_squash,sync)
```

2. Once you have the contents of **/etc/exports** updated correctly, enable and start the NFS daemons. Issue the following commands as the root user:

```
$ chkconfig nfs on
$ service nfs start
```

   or

   If you already have NFS configured and running on the master system, you may simply issue the following command as root to cause the NFS daemons to re-read the **/etc/exports** file:

```
$ exportfs -rv
```

   The NFS service should start (or restart) properly. If it does not, the most common reason is a syntax error in the **/etc/exports** file. See the **exports(5)** man page for more information.

Configuring DHCP

DHCP must be configured manually to allow communication among the nodes. This is accomplished through the **/etc/dhcpd.conf** file.

cm-cfgboot provides instructions for modifying **dhcpd.conf** after a node is added, deleted or modified.

To configure **dhcpd.conf** for the cluster, perform the following steps:

1. On the master system, copy the example **dhcpd.conf** file to **/etc**:

```
$ cp /usr/share/doc/ccur-yaci-4.1/dhcpd.conf /etc/dhcpd.conf
```

   Edit the file by providing the values shown in italics below with those appropriate to your cluster as recorded on your **Node Information Worksheet**. This example shows a master and three cluster nodes. Refer to **dhcpd.conf(5)** if necessary for more information about configuring DHCP.

```
ddns-update-style ad-hoc;
server-name "master-name";
allow bootp;

subnet subnet netmask netmask {
    option subnet-mask netmask;
    option broadcast-address broadcast-address;

    # default gateway
    option routers routers;

    # DNS setup
    option domain-name-servers domain-name-servers;
    option domain-name "domain-name";

    group {
        filename "yaci/pxelinux.0";
        use-host-decl-names on;
```

```
            host node1 {
                hardware ethernet node1_MAC;
                fixed-address node1_ipaddress;
            }
            host node2 {
                hardware ethernet node2_MAC;
                fixed-address node2_ipaddress;
            }
            host node3 {
                hardware ethernet node3_MAC;
                fixed-address node3_ipaddress;
            }
        }
    }
```

Each node in the cluster must have a unique "host" entry. If you are already serving DHCP from the master server, entries for other machines that are not part of the cluster are allowed and will not interfere with the cluster operation.

2. If cluster nodes have multiple network interfaces which must be configured, they can also be configured from the master's DHCP server provided that the master is also on the same networks.

   For each additional network, add a subnet declaration and configure IP addresses for the cluster node network interfaces on that network. For example:

```
subnet 192.1.0.0 netmask 255.255.0.0 {
    option subnet-mask 255.255.0.0;
    option broadcast-address 192.1.255.255;

    group {
        host node1-if2 {
            hardware ethernet 00:30:48:59:F7:B7;
            fixed-address 192.1.1.3;
        }
        host node2-if2 {
            hardware ethernet 00:30:48:59:6B:15;
            fixed-address 192.1.1.4;
        }
        host node3-if2 {
            hardware ethernet 00:30:48:59:F7:A3;
            fixed-address 192.1.1.5;
        }
    }
```

3. Once you have the contents of **dhcpd.conf** updated correctly, enable and start the DHCP service by issuing the following commands as the root user:

```
$ chkconfig dhcpd on
$ service dhcpd start
```

   The service should start up properly. If it does not, the most common reason is a syntax error in **/etc/dhcpd.conf**. See the **dhcpd.conf(5)** man page for more information.

Enabling TFTP

Cluster Manager uses TFTP to download files to the cluster nodes when booting. Enable TFTP on the master server by issuing the following commands as the root user:

```
$ chkconfig tftp on
$ service xinetd restart
```

No separate configuration step is required for this operation, however if your site is concerned with security, you may wish to tighten the access controls of the **tftpd** daemon. For more information, refer to the **tftpd(8)** man page and the **/etc/xinetd.d/tftp** control file.

Booting Cluster Nodes

The cluster nodes can be booted once the master system is configured and the cluster file system image is built. Disk-based nodes will be installed the first time they are booted. The following subsections discuss the steps involved in disk-based node installation and subsequent booting of disk-based and diskless nodes.

Enabling PXE Booting

Pre-Execution Environment (PXE) booting must be enabled on every cluster node.

NOTE

If the BIOS on your system does not support PXE booting, 'etherboot' may be an option; however, Concurrent does not support this configuration. The following instructions apply to systems with a BIOS that supports PXE booting.

To enable PXE booting, do the following:

1. Reboot the cluster node and stop the system immediately after POST (Power-On Self-Test), normally by pressing F2, to get into the BIOS settings menu.
2. Each iHawk machine type has a slightly different BIOS settings menu, however the general rule is to navigate to the 'PCI Device' or the 'Integrated Devices' section of the BIOS menu and enable PXE boot on the first Ethernet interface that is present. Ensure that the chosen interface is connected to a switch that is present on the same network as the master system.

NOTE

The MAC address of the Ethernet interface on which you choose to enable PXE booting must match the MAC address for the node in the **`MAC.info`** file (see “Collecting Node Information” on page 1-10).

3. Verify that the ‘Boot Device Order’ is set so that the system will attempt to PXE boot ***first*** before it attempts to boot from either the floppy, CD-ROM or hard-disk. This step is very important as the node will not successfully perform auto-installation unless PXE booting is the first boot method tried.

NOTE

If there are no PXE devices listed for Boot Device Order, save the BIOS settings, exit the BIOS settings menu, restart the system and re-enter the BIOS menu in order to make the PXE device options enabled in step 2 available for this step.

4. Once the BIOS settings are correct, save the settings and exit the BIOS settings menu.

Understanding the Boot Sequence

Once PXE is enabled, a cluster node will perform the following sequence of events when booting:

1. Send a DHCP broadcast
2. Receive a DHCP response from the master system—response specifies that the ‘pxelinux.0’ boot-loader should be booted
3. Boots the pxelinux.0 boot-loader and searches for a PXE configuration file to use for this node
4. pxelinux.0 follows the instructions in the PXE configuration found on the master system

Installing Disk-Based Nodes

To install a cluster image on a disk-based node, run **`cm-cfgboot -l`** *nodename* to verify that the **`/tftpboot/yaci/pxelinux.cfg`** directory on the master system is configured so that the node will use the ‘*type*.**`install`**’ PXE file (see “Running cm-cfgboot” on page 1-11 and “PXE Configuration” on page 1-12 for more information). If it is not set correctly, run **`cm-cfgboot`** *nodename* to update the boot method to ‘install’. Then reboot the node.

When the system boots, the pxelinux.0 boot-loader does the following:

1. Downloads and boots the YACI installation kernel from the master system
2. Zeroes the entire contents of the local hard disk
3. Partitions the local hard disk
4. Installs the cluster file system image on the local hard disk

5. Configures per-node system files (networking, hostname, etc.)
6. Installs grub into the Master Boot Record of the local hard disk
7. Creates a new PXE configuration file for this node on the master system such that the next boot will be off the local hard disk
8. Reboots

During installation, the node's system console output is redirected to the first serial communications port, known as COM1 or **/dev/ttyS0**. In order to view the node's console output, you must connect a serial terminal device to the correct serial port connector on the node.

Installation Logs

Cluster node installation generally completes without problems once the cluster master is properly configured. However, during the initial configuration of the master system it is possible that a master system configuration error will result in early cluster node installations failing.

Normally, during cluster node installation the serial console of the node displays an ASCII picture of a yak with text printed below it detailing the installation progress. If no text is being output, the installation has almost certainly run into a snag. Fortunately, a log file containing installation progress is written to the master system for each node in the cluster. The log files are located and named according to the following template:

```
/tftpboot/yaci/log/$NODENAME.log
```

By examining the contents of the node-specific log file, you can view the progress made during the node installation and see where the installer stopped if a problem occurred. The most common problems are mis-configurations of **MAC.info**, **dhcpd.conf** and **/etc/exports**. Also, verify that the NFS, DHCP and TFTP servers are running on the master system.

Booting Disk-Based Nodes

To boot a disk-based node from the local hard disk, run **cm-cfgboot -l** *nodename* to verify that the **/tftpboot/yaci/pxelinux.cfg** directory on the master system is configured so that the node will use the '*type*.**local**' PXE file (see "Running cm-cfgboot" on page 1-11 and "PXE Configuration" on page 1-12 for more information). If it is not set correctly, run **cm-cfgboot** *nodename* to update the boot method to 'local'. Then reboot the node.

When the system boots, the pxelinux.0 boot-loader boots the grub boot-loader from the local hard disk.

Grub will pause for 10 seconds and display "Press Any Key To Continue" on both the first serial port and the node's attached VGA console (if any). If no key is pressed on the VGA console's keyboard, the node's console will be automatically re-directed to the first serial port. If a key is pressed, the system's console will display on the VGA console's attached monitor.

Grub will then display a menu that presents a choice of kernels to boot on the system's console. If no key is pressed within 10 seconds, the default kernel will be booted (see "Kernel Selection" on page 1-20 for more information). You can use the menu to select an alternate kernel, or edit kernel command line options. See the help text printed below the on-screen menu for more information.

Note that this entire process happens automatically following the installation of a disk-based node.

Booting Diskless Nodes

No installation is required to boot a diskless node. The kernel and file system image are loaded directly from the master system.

To boot a diskless node, run **`cm-cfgboot -l`** *nodename* to verify that the **`/tftpboot/yaci/pxelinux.cfg`** directory on the master system is configured so that the node will use the '*type*.**`diskless`**' PXE file (see "Running cm-cfgboot" on page 1-11 and "PXE Configuration" on page 1-12 for more information). If it is not set correctly, run **`cm-cfgboot`** *nodename* to update the boot method to 'diskless'.

When the system boots, the pxelinux.0 boot-loader displays a prompt on the system's console presenting a choice of kernels to boot. If no key is pressed within 7 seconds, the default kernel will be booted (see "Kernel Selection" on page 1-20 for more information). You can type an alternate kernel name at the prompt. The kernel and root ramdisk are then downloaded from the master system and the kernel is booted.

Advanced Configuration

The following sections discuss more advanced configuration issues that may be suitable to your cluster.

Multiple Node Types

The default node type is 'redhawk'. Additional node types may be used if some nodes must use a different file system image. Each node type uses one file system image. You may create as many file system images as you like, provided you have enough disk space.

NOTE

The creation of each cluster file system image takes considerable disk space. Be sure to configure the master system so that the **`/tftpboot/yaci`** directory is on a large disk partition if you plan to define and create several node types. Refer to "Creating a Cluster File System Image" on page 1-5 for sizing information.

To switch the node type of a disk-based node that has already been installed, refer to "Reinstalling Disk-based Nodes" on page 1-22.

If you decide to create additional node types, for each additional node type desired:

1. Repeat the steps in "Creating a Cluster File System Image" on page 1-5 using the desired type name wherever a type name is optional.
2. Repeat the steps in "Configuring the Cluster Network Using cm-cfgboot" on page 1-10 using the desired type name instead of 'redhawk'.

Red Hat Kernels

This section applies only to disk-based nodes. During the creation of the cluster file system image, Cluster Manager assumes that the hardware configuration of the master system will exactly match that of the cluster nodes. In practice, this is not always true (e.g. one node may have a RAID controller for increased disk performance). If the cluster contains non-uniform hardware configuration, the root image on a given cluster node may not be able to successfully boot the Red Hat kernels that are supplied in the root image.

In this case, you will need to manually create an **initrd** file that contains the correct kernel modules needed to boot the Red Hat kernel on the non-uniform node. To do this,

1. First boot the node with the RedHawk kernel.
2. Then, log into the node as the root user and issue the following command:

```
# mkinitrd /boot/initrd-2.4.21-27.0.2.ELsmp.img 2.4.21-27.0.2.EL
```

This command will examine the current hardware configuration and produce an updated and customized **initrd** for the Red Hat kernel that will allow the kernel to successfully boot on the current node.

Note that the specific kernel version numbers may vary based on the version of Cluster Manager being used. Look in the **/boot** directory on the node to see exactly which Red Hat kernel versions are available in the root image.

Kernel Selection

By default, the RedHawk 'trace' kernel is automatically booted on each of the cluster nodes. You can change this default.

For Disk-based Nodes

You can change the default kernel boot setting by editing the **/boot/grub/grub.conf** file.

The **grub.conf** file has a 'default' line that selects which kernel to boot. Normally, the default setting looks as follows:

```
default=#
```

The following table shows how the ‘default’ setting can be used to select different kernels.

#	Kernel Suffix	Trace	Debug
0	trace	yes	no
1	debug	yes	yes
2	(none)	no	no

Changing this value will change the kernel that is booted by default on each of the cluster nodes.

Note that once a node is installed, it is always possible to log into the node and change the **/boot/grub/grub.conf** file on that node individually (just like almost every other aspect of the node’s configuration).

For Diskless Nodes

The kernel that is booted on diskless nodes is configured in the **/tftpboot/yaci/pxelinux.cfg/***type***.diskless** PXE configuration file (see “PXE Configuration” on page 1-12).

You may choose which kernel to boot by default by changing:

```
DEFAULT redhawk-trace
```

to be one of:

```
DEFAULT redhawk
DEFAULT redhawk-debug
```

Cluster Maintenance

Successful long-term deployment of a cluster requires the ability to maintain cluster file system images. The following sections describe how to perform the following tasks:

- adding nodes to a cluster
- re-installing the hard disk on disk-based nodes
- recreating a cluster file system image
- updating software and/or configuration files on a cluster file system image

Adding, Deleting or Modifying Nodes in a Cluster

Cluster nodes can be added, deleted or modified at any time. Follow the directions in “Running cm-cfgboot” on page 1-11”.

Reinstalling Disk-based Nodes

Disk-based nodes may be reinstalled at any time. To reinstall a disk-based node, run **`cm-cfgboot -l`** *nodename* to verify that the **`/tftpboot/yaci/pxelinux.cfg`** directory on the master system is configured so that the node will use the '*type*.**`install`**' PXE file (see "Running cm-cfgboot" on page 1-11 and "PXE Configuration" on page 1-12 for more information). If it is not set correctly, run **`cm-cfgboot`** *nodename* to update the boot method to 'install'. Then reboot the node.

Recreating a Cluster File System Image

A new cluster file system image may be created from scratch at any time. Note that disk-based nodes will have to be reinstalled and diskless nodes will have to be rebooted in order to use the new image.

NOTE

All diskless nodes using a cluster file system image must be shutdown prior to creating the new image.

To create a new cluster file system image, simply repeat the procedure detailed in "Creating a Cluster File System Image" on page 1-5.

Updating a Cluster File System Image

It is possible to modify a cluster file system image once it is created. Note that disk-based nodes will have to be re-installed and diskless nodes will have to be rebooted in order to use the modified image.

To modify a cluster file system image, perform the following steps:

1. Modify files in the cluster file system image directory (see "Customizing the Cluster File System" on page 1-7).
2. Create a new tar image for disk-based nodes (see "Building a Compressed Tar File for Disk-Based Nodes" on page 1-10).
3. Create a new ramdisk image for diskless nodes (see "Building a Ramdisk Image for Diskless Nodes" on page 1-10).
4. Run **`cm-cfgboot`** on the master to update the PXE configuration so that disk based nodes will be reinstalled on the next boot (see "Running cm-cfgboot" on page 1-11).
5. Reboot all nodes.

2
Grid Engine Software

2
Grid Engine Software

This chapter describes Grid Engine, the software that manages and schedules jobs across the cluster, and provides instructions for installing the product and configuring your cluster.

Overview

RedHawk Cluster Manager includes Grid Engine 6.1, an open source batch-queuing system developed by Sun Microsystems that accepts, schedules, dispatches and manages the remote execution of large numbers of user jobs. Grid Engine, also referred to as SGE, integrates multiple clusters into a grid environment.

A *grid* is a collection of computing resources that perform tasks. It can provide a single point of access to a large powerful distributed resource, or it may provide many access points, but to the user, it appears as a single computational resource.

Concurrent's binary distribution of SGE includes the following:

- Default build of the **sge-V61_TAG-src.tar.gz** source without Java support.
- Linked against version 4.5.20 of Berkeley DB from Oracle Corporation (**db-4.5.20.tar.gz**). The required files of the Berkeley DB are installed under **$SGE_ROOT/lib**, **$SGE_ROOT/utilbin**, and **$SGE_ROOT/include**.
- Also linked against OpenSSL version 0.9.7l (**openssl-0.9.7l.tar.gz**) provided by the OpenSSL Project.
- DRMAA language bindings installed under **$SGE_ROOT/drmaa** (perl, python and ruby).
- Files, scripts and wrappers developed or customized by Concurrent to facilitate installing and using SGE:
    - **$SGE_ROOT/inst_cell** – a wrapper around **inst_sge** that automates many of the manual steps required by **inst_sge**
    - **$SGE_ROOT/util/ssh_setup.sh** – a script that establishes password free **scp**/**ssh** access to the execution hosts from the master host
    - **/etc/profile/sge.sh** – customized environment variable file for use under RedHawk
    - **/etc/services** – includes **sge_qmaster** and **sge_execd** port services

Open Source Resources

Grid Engine 6.1 source code and other resources can be accessed from the following websites:

Grid Engine Project Home:
http://gridengine.sunsource.net/

Documents download site:
http://docs.sun.com/app/docs/coll/1017.4
Note that these documents are included with RedHawk Cluster Manager (see "Grid Engine Documentation" on page 2-3)

Source code download (V61_TAG CVS tag):
http://gridengine.sunsource.net/servlets/ProjectSource

Source code download (tarball):
http://gridengine.sunsource.net/servlets/ProjectDocumentList

Sun Industry Standards Source License (SISSL):
http://gridengine.sunsource.net/license.html
A copy of the license can be found in **$SGE_ROOT/LICENSE**

Oracle Berkeley DB source code and other resources can be accessed at the following web sites:

Oracle Berkeley DB home:
http://www.oracle.com/technology/products/berkeley-db/db/index.html

Source code download (db-4.5.20.tar.gz):
http://www.oracle.com/technology/software/products/berkeley-db/index.html

License for Oracle Berkeley DB:
http://www.oracle.com/technology/software/products/berkeley-db/htdocs/oslicense.html
A copy of the license can be found in **$SGE_ROOT/LICENSE**

OpenSSL source code and other resources can be accessed at the following web sites:

OpenSSL home:
http://www.openssl.org/

Source code download (openssl-0.9.7l.tar.gz):
http://www.openssl.org/source/

OpenSSL License:
http://www.openssl.org/source/license.html
A copy of the license can be found in **$SGE_ROOT/LICENSE**

Grid Engine Documentation

This section provides documentation and other resources you will need to administer and use Grid Engine on your cluster.

Manuals

The following manuals, produced by Sun Microsystems, are applicable to the Grid Engine open source software package included with Cluster Manager:

Manual Name	Publication Number
Sun N1 Grid Engine 6.1 Installation Guide	820-0697
Sun N1 Grid Engine 6.1 Administration Guide	820-0698
Sun N1 Grid Engine 6.1 User's Guide	820-0699
Sun N1 Grid Engine 6.1 Release Notes	820-0700

These document PDFs are supplied with RedHawk Cluster Manager. Click on the red entry to display the document pdf. They can also be viewed or downloaded from the Internet at **http://docs.sun.com/app/docs/coll/1017.4**.

Refer to the section "Functionality Differences" below for differences between the revision of Grid Engine included in Cluster Manager and the documentation.

Man Pages

All Grid Engine man pages are available online. To view a man page, type: **`man`** *command*.

A summary of the man pages is provided in Table 2-1.

Table 2-1 Grid Engine Man Page Summary

Command Name	Description
Grid Engine User Commands (/usr/local/sge/man/man1)	
gethostbyaddr	gets hostname via IP address
gethostbyname	gets local host information for specified hostname
gethostname	gets local hostname
getservbyname	gets configured port number of service
qacct	reports and accounts for Grid Engine usage
qsub	submits a batch job to Grid Engine
qsh	submits an interactive X-windows session to Grid Engine
qlogin	submits an interactive login session to Grid Engine

qrsh	submits an interactive rsh session to Grid Engine
qalter	modifies a pending batch job of Grid Engine
qresub	submits a copy of an existing Grid Engine job
qconf	Grid Engine Queue Configuration
qdel	deletes Grid Engine jobs from queues
qhold	holds back Grid Engine jobs from execution
qhost	shows the status of Grid Engine hosts, queues, jobs
qmake	distributed parallel make, scheduling by Grid Engine
qmod	modifies a Grid Engine queue
qmon	X-Windows OSF/Motif graphical user interface for Grid Engine
qping	checks application status of Grid Engine daemons
qquota	shows current usage of Grid Engine resource quotas
qrls	releases Grid Engine jobs from previous hold states
qselect	used to modify queue attributes on a set of queues
qstat	shows the status of Grid Engine jobs and queues
qtcsh	tcsh v6.09 with transparent remote execution by use of qrsh
sge_ckpt	Grid Engine checkpointing mechanism and checkpointing support
sge_intro	a facility for executing UNIX jobs on remote machines
sgepasswd	modifies the Grid Engine password file of Grid Engine
sge_types	Grid Engine type descriptions
submit	describes Grid Engine User Commands
Grid Engine Standard Applications API (DRMAA) (/usr/local/sge/man/man3)	
DRMAA is a set of standard APIs developed by the Global Grid Forum for application builders, portal builders and ISV's.	
Grid Engine DRMAA job template (jt) handling:	
drmaa_allocate_job_template()	allocates a new jt
drmaa_delete_job_template()	releases all resources associated with the jt
drmaa_set_attribute()	stores the value under "name" in the jt
drmaa_get_attribute()	returns specified number of bytes from "name" in the jt
drmaa_set_vector_attribute()	stores the strings under "name" in the jt
drmaa_get_vector_attribute()	returns all string values stored in "name" in the jt
drmaa_get_next_attr_value()	returns specified number of bytes stored next in "values" in the jt
drmaa_get_num_attr_values()	returns the number of entries in the DRMAA values string vector
drmaa_release_attr_values()	releases all resources associated with "values" in the jt
DRMAA job template (jt) attributes:	
drmaa_get_attribute_names()	returns into "values" the set of supported jt attribute names
drmaa_get_next_attr_name()	returns specified number of bytes into "value" from next string in "values"
drmaa_get_num_attr_names()	returns the number of names in the names string vector

drmaa_get_vector_attribute_names()	returns into "values" the set of supported vector jt names
drmaa_release_attr_names()	releases all resources associated with "values"
Monitor and control DRMAA jobs:	
drmaa_job_ps()	returns the status of the job id into the integer pointed to by "remote_ps"
drmaa_control()	applies control operations on jobs
Start/finish Grid Engine DRMAA session:	
drmaa_init()	initializes the DRMAA API library and creates a new session
drmaa_exit()	closes the session before process termination
Miscellaneous DRMAA functions:	
drmaa_strerror()	returns a message associated with the DRMAA error number
drmaa_get_contact()	returns a string containing contact info related to the current session
drmaa_version()	returns the major and minor version numbers of the DRMAA library
drmaa_get_DRM_system()	returns specified number of chars from Grid Engine version string
drmaa_get_DRMAA_implementation()	returns specified number of chars from DRMAA version string
DRMAA Job submission:	
drmaa_run_job()	submits job with attributes defined in the job template (jt)
drmaa_run_bulk_jobs()	submits an array job like qsub option ‘-t start-end:incr’
drmaa_get_next_job_id()	returns specified number of chars into "value" of next entry in "values"
drmaa_release_job_ids()	releases all resources associated with "values" job id
Waiting for DRMAA jobs to finish:	
drmaa_synchronize()	blocks calling thread until all specified job_ids have terminated
drmaa_wait()	blocks calling thread until a job terminates
drmaa_wifaborted()	stores non-zero value into the integer pointed to by "aborted"
drmaa_wifexited()	stores non-zero value into the integer pointed to by "exited"
drmaa_wifsignaled()	stores non-zero value into the integer pointed to by "signaled"
drmaa_wcoredump()	stores non-zero value into the integer pointed to by "core_dumped"
drmaa_wexitstatus()	stores non-zero value into the integer pointed to by "exit_code"
drmaa_wtermsig()	stores non-zero value into the integer pointed to by "signal"
Grid Engine File Formats (/usr/local/sge/man/man5)	
access_list	Grid Engine access list file format
accounting	Grid Engine accounting file format
bootstrap	Grid Engine bootstrap file
calendar_conf	Grid Engine calendar configuration file format
checkpoint	Grid Engine checkpointing environment configuration file format
complex	Grid Engine complexes configuration file format
host_aliases	Grid Engine host aliases file format
host_conf	Grid Engine execution host configuration file format

hostgroup	host group entry file format
project	Grid Engine project entry file format
qtask	file format of the qtask file
queue_conf	Grid Engine queue configuration file format
reporting	Grid Engine reporting file format
sched_conf	Grid Engine default scheduler configuration file
sge_aliases	Grid Engine path aliases file format
sge_conf	Grid Engine configuration files
sgepasswd	Modify the Grid Engine password file of Grid Engine
sge_pe	Grid Engine parallel environment configuration file format
sge_priority	Grid Engine job priorities
sge_qstat	Grid Engine default qstat file format
sge_request	Grid Engine default request definition file format
sge_resource_quota	Grid Engine resource quota file format
share_tree	Grid Engine share tree file format
user	Grid Engine user entry file format
usermapping	user mapping entry file format
Grid Engine Administrative Commands (/usr/local/sge/man/man8)	
sge_execd	Grid Engine job execution agent
sge_qmaster	Grid Engine master control daemon
sge_schedd	Grid Engine job scheduling agent
sge_shadowd	Grid Engine shadow master daemon
sge_shepherd	Grid Engine single job controlling agent

Functionality Differences

The N1 Grid Engine documentation by Sun Microsystems is used by the open source Grid Engine project as applicable documentation. Concurrent's distribution of Grid Engine is the default build of Grid Engine version 6 update 8 as documented in those documents with the following differences:

- No Java support
- No ARCO support
- Windows is not officially supported or tested

Java and ARCO support can be supplied through an RIQ with Concurrent Professional Services.

Configuring Your Cluster

Overview

It is suggested that the person responsible for installing Grid Engine and setting up the cluster have the N1 Grid Engine 6 documents available and that they familiarize themselves with the Grid Engine cluster architecture. For the example presented in this document, however, it should not be necessary to devote a great deal of study to the N1 Grid Engine 6 documents before attempting to set up the basic cluster, which is defined to be a single “master host” and one or more “execution hosts”.

The ccur-sge-V61-1 binary rpm installation makes it easy to quickly configure any number of cluster nodes with minimal effort. When the rpm is installed on a system, that system becomes capable of assuming any role in the cluster. It is only a matter of configuring individual nodes to assume the role(s) they are assigned by running a handful of configuration scripts and by making some common configuration files accessible to the appropriate group of nodes.

It is possible to assign any number of roles to a given node. It is generally true that the master host should be dedicated to the job of being a master host. The master host is the brains of the cluster and should be left to the complex task of coordinating the efforts of the execution hosts.

The ccur-sge-V61-1 rpm also installs BerkeleyDB 4.5.20, which is used by Grid Engine for spooling.

Users are expected to make their own decisions on how best to configure their cluster based on individual needs. The examples provided here are designed to be simple and do not necessarily represent an ideal configuration.

Grid Engine is a complex application and can be configured in many different ways. An in depth study of the N1 Grid Engine 6 documentation will be necessary in order to fully optimize a cluster.

Grid Engine File System Requirements

Disk-based nodes should have the following minimums:

- Master host: 100 MB memory, 500 MB disk space
- Execution host: 20 MB memory, 50 MB disk space
- File server: 20 MB disk space + 20 MB per architecture

Diskless nodes should have the following minimums:

- Master host: 1 GB RAM; 2 GB is recommended.
- Execution host: 512 MB RAM; 1 GB is recommended.

Procedures

Procedures for setting up a "cluster grid cell" as defined by Sun Microsystems are given in this section and are included in **$SGE_ROOT/SETUP**.

You will be using two scripts provided by Concurrent, **inst_cell** and **ssh_setup.sh**, which are designed to help you rapidly install a working cell configured in this way:

- One Master Host and N execution hosts
- Berkeley DB spooling
- Execution host local spooling

You may also use the scripts provided by the grid engine project to either install manually (**install_qmaster**, **install_execd**), or use **inst_sge**. Refer to the *Sun N1 Grid Engine 6.1 Installation Guide* for details.

inst_cell

inst_cell is a wrapper around **inst_sge** which uses a modified **inst_sge** template and extra bash scripting to automate steps you would normally have to do by hand. **inst_cell** sources the template file **$SGE_ROOT/usr/local/sge/util/install_modules/inst_cell_template.conf**.

You can query the template to see the config. You can modify the defaults but it is recommended you start by experimenting with the default configuration shown here:

```
# inst_cell -q
/usr/local/sge/util/install_modules/inst_cell_template.conf
 SGE_CELL              = redhawk
 SGE_ROOT              = /usr/local/sge
 QMASTER_SPOOL_DIR     = /var/spool/sge/redhawk/qmaster
 DB_SPOOLING_DIR       = /var/spool/sge/redhawk/spooldb
 EXECD_SPOOL_DIR       = /var/spool/sge/redhawk/spool
 EXECD_SPOOL_DIR_LOCAL = /var/spool/sge/redhawk
 BACKUP_ROOT           = /var/spool/sge

 SPOOLING_METHOD       = berkeleydb
 DB_SPOOLING_SERVER    = none

 SCHEDD_CONF           = 1
 RESCHEDULE_JOBS       = wait
 PAR_EXECD_INST_COUNT  = 20
 GID_RANGE             = 20000-20100

 SGE_QMASTER_PORT      = 536
 SGE_EXECD_PORT        = 537
 HOSTNAME_RESOLVING    = true
 DEFAULT_DOMAIN        = none

 MASTER_HOST           =
 SHADOW_HOST           =
 SUBMIT_HOST_LIST      =
 ADMIN_HOST_LIST       =
 EXEC_HOST_LIST        =
```

inst_cell usage:

```
# inst_cell
/usr/local/sge/inst_cell

Usage:
    inst_cell -q    query cell template
    inst_cell -c    install cell
    inst_cell -m    install master host
    inst_cell -x    install execution host(s)
    inst_cell -uc   uninstall cell
    inst_cell -um   uninstall master host
    inst_cell -ux   uninstall execution host(s)
    inst_cell -b    backup cell or execution host
    inst_cell -s    stop cell or execution host service(s)
    inst_cell -r    (re)start cell or execution host service(s)
    inst_cell -k    kill cell master and execution host daemon(s)
    inst_cell -e    enable cell or execution host service(s)
    inst_cell -d    disable cell or execution host service(s)
    inst_cell -u    update cell config
    inst_cell -h    expanded help on these commands
```

inst_cell expanded usage:

```
# inst_cell -h

 inst_cell - automated grid engine cell deployment tool

 Overview:

  This script is a wrapper around the sge_inst script. It uses a modified
  sge_inst template to automate some of the manual steps normally required
  by sge_inst.

 Requirements:

  All nodes must be preinstalled with the grid engine software
  All nodes must grant password free ssh/scp access to the master host

 Commands:

 inst_cell -q   query cell template
        Displays location of the inst_cell template
        Displays relevant template configuration values

 inst_cell -c   install cell
        On MASTER_HOST:    install and start MASTER_HOST
                           install and start EXEC_HOST_LIST

 inst_cell -m   install master host
        On MASTER_HOST:    install and start MASTER_HOST

 inst_cell -x   install execution host(s)
        On MASTER_HOST:    install and start EXEC_HOST_LIST
        On execution host: install and start execution host

 inst_cell -uc  uninstall cell
        On MASTER_HOST:    stop and uninstall EXEC_HOST_LIST
                           stop and uninstall MASTER_HOST

 inst_cell -um  uninstall master host
        On MASTER_HOST:    stop and uninstall MASTER_HOST
```

```
 inst_cell -ux  uninstall execution host(s)
        On MASTER_HOST:    stop and uninstall EXEC_HOST_LIST
        On execution host: stop and uninstall execution host

 inst_cell -b   backup cell
        On MASTER_HOST:    back up MASTER_HOST and EXEC_HOST_LIST
        On execution host: back up execution host

 inst_cell -s   stop cell or execution host service(s)
        On MASTER_HOST:    stop EXEC_HOST_LIST sgeexecd
                           stop MASTER_HOST sgemaster
        On execution host: stop sgeexecd

 inst_cell -r   (re)start cell or execution host service(s)
        On MASTER_HOST:    restart MASTER_HOST sgemaster
                           restart EXEC_HOST_LIST sgeexecd
        On execution host: restart sgeexecd service

 inst_cell -k   kill cell master and execution host daemon(s)
        On MASTER_HOST:    kill EXEC_HOST_LIST sge_execd
                           kill MASTER_HOST sge_schedd sge_qmaster
        On execution host: kill sge_execd daemon

 inst_cell -e   enable cell or execution host service(s)
        On MASTER_HOST:    chkconfig MASTER_HOST sgemaster on
                           chkconfig EXEC_HOST_LIST sgeexecd on
        On execution host: chkconfig sgeexecd on

 inst_cell -d   disable cell or execution host service(s)
        On MASTER_HOST:    chkconfig MASTER_HOST sgemaster off
                           chkconfig EXEC_HOST_LIST sgeexecd off
        On execution host: chkconfig sgeexecd off

 inst_cell -u   update cell config
        On MASTER_HOST:        copy template and SGE_CELL/common to
                               EXEC_HOST_LIST
        On execution host: copy template and SGE_CELL/common from
                               MASTER_HOST

 inst_cell -h   expanded help on these commands
```

ssh_setup.sh

Use **`$SGE_ROOT/util/ssh_setup.sh`** to set up password free **`scp`**/**`ssh`** access to the execution hosts from the master host. Password free **`scp`**/**`ssh`** access is required for any type of install using **`inst_cell`**.

ssh_setup.sh usage:

```
# ssh_setup.sh
/usr/local/sge/util/ssh_setup.sh

 Set up password free ssh/scp access from this system to a remote system
 The remote system can be a single hostname or a list of systems
 You must run this prior to using inst_cell or inst_sge

 This script can be used to set up any system or group of systems
 You must run it from both directions two set up bidirectional access

 usage:
```

```
 ssh_setup.sh EXEC_HOST_LIST
              Set up password free ssh/scp access to EXEC_HOST_LIST
              Use this on the MASTER_HOST prior to running inst_cell
              Don't use a dollar sign (not $EXEC_HOST_LIST)
              EXEC_HOST_LIST must be set in:
              /usr/local/sge/util/install_modules/inst_cell_template.conf

 ssh_setup.sh MASTER_HOST
              Set up password free ssh/scp access to MASTER_HOST
              Use this on an execution host prior to running inst_cell
              Don't use a dollar sign (not $MASTER_HOST)
              MASTER_HOST must be set in:
              /usr/local/sge/util/install_modules/inst_cell_template.conf

 ssh_setup.sh <hostname>
              Set up password free ssh/scp access to hostname

 ssh_setup.sh <hostname list>
              Set up password free ssh/scp access to hostnames in list
              Hostnames must be enclosed in quotes and separated by
            whitespace
```

Notes on the Environment

The ccur-sge-V61-1 rpm installs **/etc/profile.d/sge.sh**. This file sets up environment variables that are required to operate and install Grid Engine. Note that SGE_ARCH is for your convenience only and is not required by sge. You may modify these if you wish, or remove the file and set up your own environment variables as desired. A copy is shown below:

```
# cat /etc/profile.d/sge.sh
[ "sge" = "`/usr/bin/id -gn`" ] && export SGE_ROOT=/usr/local/sge
[ "sge" = "`/usr/bin/id -gn`" ] && export SGE_CELL=redhawk
[ "sge" = "`/usr/bin/id -gn`" ] && export SGE_ARCH=`$SGE_ROOT/util/arch`
[ "sge" = "`/usr/bin/id -gn`" ] && export MANPATH=$MANPATH:$SGE_ROOT/man
[ "sge" = "`/usr/bin/id -gn`" ] && export
PATH=$PATH:$SGE_ROOT:$SGE_ROOT/bin/$SGE_ARCH:$SGE_ROOT/utilbin/$SGE_ARCH:$SG
E_ROOT/util

[ "root" = "`/usr/bin/id -gn`" ] && export SGE_ROOT=/usr/local/sge
[ "root" = "`/usr/bin/id -gn`" ] && export SGE_CELL=redhawk
[ "root" = "`/usr/bin/id -gn`" ] && export SGE_ARCH=`$SGE_ROOT/util/arch`
[ "root" = "`/usr/bin/id -gn`" ] && export MANPATH=$MANPATH:$SGE_ROOT/man
[ "root" = "`/usr/bin/id -gn`" ] && export
PATH=$PATH:$SGE_ROOT:$SGE_ROOT/bin/$SGE_ARCH:$SGE_ROOT/utilbin/$SGE_ARCH:$SG
E_ROOT/util
```

The ccur-sge-V61-1 rpm also creates the **sgeadmin** user and **sge** group. These are for your convenience and are not required. Again, you can copy this profile or modify it to add other groups and users.

All installations must be done as “root”.

See the *Sun N1 Grid Engine 6.1 Administration Guide* for granting access privileges to others to operate/administer grid engine.

Notes on SGE_QMASTER_PORT and SGE_EXECD_PORT

The ccur-sge-V61-1 rpm installs the following entries in **/etc/services**:

```
# ccur sge cluster port services (must be the same as the cell master host)
sge_qmaster     536/tcp
sge_execd       537/tcp
```

Setting SGE_QMASTER_PORT and SGE_EXECD_PORT in **/etc/services** is the best way to enable communication between the **sge_qmaster** and **sge_execd** daemons.

These can be modified if desired or set as environment variables and removed from **/etc/services**.

Preliminary Setup

To prepare the cell for installation, follow these steps:

1. Edit **$SGE_ROOT/util/install_modules/inst_cell_template.conf** and set the following two variables:

   MASTER_HOST=*hostname of master host*

   Example: MASTER_HOST=xbox

   EXEC_HOST_LIST=*blank spaced list of execution hostnames enclosed in quotes*

   Example: EXEC_HOST_LIST="beebo bowser peach sphynx wario"

2. Setup password free **scp**/**ssh** access:

   a. Execute the command:

   ```
   # inst_cell -q
   ```

   You will see the EXEC_HOST_LIST variable listed at the bottom of the output. Verify that it reflects your setting in step 1 above.

   b. From the master host, execute the command as root:

   ```
   # ssh_setup.sh EXEC_HOST_LIST
   ```

   Note: Do not use a dollar sign: (not $EXEC_HOST_LIST)

   This will set up password free **scp**/**ssh** access to all hosts listed in EXEC_HOST_LIST according to **inst_cell_template.conf**.

   Answer 'yes' and then enter the password for each execution host.

   Below is a snipped example of the output for the first host (beebo):

   ```
   # ssh_setup.sh EXEC_HOST_LIST
   Generating public/private rsa key pair.
   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:
   9c:20:5d:cb:51:ca:14:c9:2b:3b:e8:26:a8:01:9b:04 root@xbox
   The authenticity of host 'beebo (129.134.30.25)' can't be
   established.
   ```

```
RSA key fingerprint is
73:a2:3a:d8:16:55:53:2d:33:18:31:61:36:30:62:6d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'beebo,129.134.30.25' (RSA) to the list
of known hosts.
root@beebo's password:password
.
.
.
```

If typed correctly, you should be able to execute the command again and not have to answer any prompts.

3. Maintaining **/etc/hosts**

   Now that you can **scp** without passwords, you can write scripts to copy files from the MASTER_HOST to the EXEC_HOST_LIST. You can also write scripts that ssh into EXEC_HOST_LIST nodes and perform actions.

   Things generally work better if each node in the cell has a copy of **/etc/hosts** identical to the one on the master host. If you are using dhcp, you've got a single point of failure on the dhcp server.

   Displays will also be less cluttered with long names that include the domain.

   Installation can fail if the hostname of a system is found on the local host loopback line in **/etc/hosts**.

   Below is an example of a good **/etc/hosts** file:

```
# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain localhost

# cell master host
129.134.30.92   xbox    xbox.ccur.com
# cell execution hosts
129.134.30.25   beebo   beebo.ccur.com
129.134.30.87   peach   peach.ccur.com
129.134.30.89   bowser  bowser.ccur.com
129.134.30.85   wario   wario.ccur.com
129.134.30.219  sphynx  sphynx.ccur.com
```

   If you do not want to overwrite existing host files, skip this step.

4. Install ccur-sge-V61-1 on all execution hosts if not already installed.

   If the ccur-sge-V61-1 rpm is not installed on all nodes in EXEC_HOST_LIST, copy and install the rpm on the nodes remotely from the MASTER_HOST.

Installing the Default Cluster Grid Cell

To install the default redhawk cell using **inst_cell**, follow the steps below:

1. Install the cell by executing the command from the MASTER_HOST as root:

```
# inst_cell -c
18:11:09        Installing master host
18:11:16        Copying config to execution hosts
18:11:20        Installing execution hosts
```

2. Check the installation by executing the following **qconf** command. The following output should display:

```
# qconf -sconf
global:
execd_spool_dir              /var/spool/sge/redhawk/spool
mailer                       /bin/mail
xterm                        /usr/bin/X11/xterm
load_sensor                  none
prolog                       none
epilog                       none
shell_start_mode             posix_compliant
login_shells                 sh,ksh,csh,tcsh
min_uid                      0
min_gid                      0
user_lists                   none
xuser_lists                  none
projects                     none
xprojects                    none
enforce_project              false
enforce_user                 auto
load_report_time             00:00:40
max_unheard                  00:05:00
reschedule_unknown           00:00:00
loglevel                     log_warning
administrator_mail           none
set_token_cmd                none
pag_cmd                      none
token_extend_time            none
shepherd_cmd                 none
qmaster_params               none
execd_params                 none
reporting_params             accounting=true reporting=false \
                             flush_time=00:00:15 joblog=false
sharelog=00:00:00
finished_jobs                100
gid_range                    20000-20100
qlogin_command               telnet
qlogin_daemon                /usr/sbin/in.telnetd
rlogin_daemon                /usr/sbin/in.rlogind
max_aj_instances             2000
max_aj_tasks                 75000
max_u_jobs                   0
max_jobs                     0
auto_user_oticket            0
auto_user_fshare             0
auto_user_default_project    none
auto_user_delete_time        86400
delegated_file_staging       false
reprioritize                 0
```

3. Proceed to the section "Testing the Configuration".

Testing the Configuration

Once you have configured the cluster with Grid Engine, you should test the configuration.

Instructions are given below and are in the file **`$SGE_ROOT/SGETEST`**.

You must have completed installation as described in “Configuring Your Cluster” on page 2-7.

You might want to spend some time configuring and testing with SGETEST as well as the tests described in the Grid Engine PDFs and web resources.

When running tests, note the following:

- All commands are run from the cell master host.
- The cell master host should never be an execution host for performance.

sgetest

`sgetest` is a single threaded synthetic benchmark. It should run from 10 to 120 seconds, depending on the architecture and load of the node on which it runs. The test performs a number of floating point operations in a loop with branching instructions. It then prints the name of the host on which it ran, the process ID, and the time it took the loop to finish in seconds and nanoseconds. For example:

```
xbox:3370          67.887999788
```

The time it takes to finish the loop doesn’t indicate much by itself, but when compared with the time it takes on other systems you can determine the differences in performance for this particular test on the various systems. This will help you understand some of the things grid engine is doing for you.

The SGETEST harness has been pre-installed under **`$SGE_ROOT/examples`**:

- SGETEST binary: **`jobsbin/$SGE_ARCH/sgetest`**
- Source code: **`sgetest.c`**
- Job script: **`jobs/sgetest.sh`**

You will need to use the “job script” to submit this job to grid engine:

```
# cat /usr/local/sge/examples/jobs/sgetest.sh
#!/bin/bash
$SGE_ROOT/examples/jobsbin/$SGE_ARCH/sgetest
```

Grid engine will invoke the **`sgetest`** binary on one of the CPUs in your cell, if you use the **`qsub`** command, or the **`qmon`** GUI. But before doing that, you might try running the script on your master host:

```
# /usr/local/sge/examples/jobs/sgetest.sh
xbox:3372               68.045150594
```

1. Submit the provided "sgetest" job and use the **qstat** command to monitor status:

```
# qsub /usr/local/sge/examples/jobs/sgetest.sh
# qstat
```

The example below shows the job state changing from "wait" to "transition" to "run" to done (job disappears from queue):

```
# qsub /usr/local/sge/examples/jobs/sgetest.sh
Your job 1 ("sgetest.sh") has been submitted
# qstat
job-ID  prior   name       user  state submit/start at     queue        slots ja-task-ID
-----------------------------------------------------------------------------------------
     1 0.00000 sgetest.sh root   qw    07/06/2007 18:00:29                  1
# qstat
job-ID  prior   name       user  state submit/start at     queue        slots ja-task-ID
-----------------------------------------------------------------------------------------
     1 0.55500 sgetest.sh root   t     07/06/2007 18:00:29 all.q@beebo      1
# qstat
job-ID  prior   name       user  state submit/start at     queue        slots ja-task-ID
-----------------------------------------------------------------------------------------
     1 0.55500 sgetest.sh root   r     07/06/2007 18:00:29 all.q@beebo      1
# qstat
```

The example below shows 10 jobs in different states:

```
# qstat
job-ID  prior   name       user  state submit/start at     queue        slots ja-task-ID
-----------------------------------------------------------------------------------------
      1 0.55500 sgetest.sh root  r     07/06/2007 15:36:36 all.q@beebo      1
      1 0.55500 sgetest.sh root  t     07/06/2007 15:36:36 all.q@beebo      1
      1 0.55500 sgetest.sh root  t     07/06/2007 15:36:36 all.q@beebo      1
      1 0.55500 sgetest.sh root  t     07/06/2007 15:36:36 all.q@beebo      1
      1 0.55500 sgetest.sh root  r     07/06/2007 15:36:36 all.q@bowser     1
      1 0.55500 sgetest.sh root  r     07/06/2007 15:36:36 all.q@peach      1
      1 0.55500 sgetest.sh root  t     07/06/2007 15:36:36 all.q@peach      1
      1 0.55500 sgetest.sh root  r     07/06/2007 15:36:36 all.q@wario      1
      1 0.55500 sgetest.sh root  t     07/06/2007 15:36:36 all.q@wario      1
      1 0.00000 sgetest.sh root  qw    07/06/2007 15:36:36                  1
```

Note that stdout and stderr will be redirected into files on each execution host. The default is to write into two files in the home directory of the user who submitted the job. The files are named by appending the job number to the job name (which defaults to the name of the job script):

`sgetest.sh.o1`	stdout
`sgetest.sh.e1`	stderr

You may ignore the following warning (it is caused by the bash shell):

```
"Warning: no access to tty (Bad file descriptor)."
"Thus no job control in this shell."
```

2. Submit 10 jobs.

   This script will submit and display the q status after each **qsub**, then begin "watching" **qstat** until all the jobs are done:

```
# for i in 1 2 3 4 5 6 7 8 9 10
> do
> qsub /usr/local/sge/examples/jobs/sgetest.sh
> qstat
> done; watch -n1 -d qstat
```

3. You could also start a **watch** in one terminal and submit jobs in another:

   Terminal 1: **watch -n1 -d qstat**

   Terminal 2: **qsub /usr/local/sge/examples/jobs/sgetest.sh**
   (repeat qsubs at random intervals to simulate random activity)

   Terminal 3: **ssh** to one of the systems running a job and run **top**

4. Finally, run a synthetic benchmark on the grid.

   a. First, clean up all the output trash we've built up on the execution hosts up to this point:

```
# for i in beebo bowser peach sphynx wario;do ssh $i "rm -f
sgetest.sh.*";done
```

   b. Next, submit as many jobs as processors and watch the queue empty:

```
# qconf -sep
HOST                      PROCESSOR        ARCH
===============================================
beebo                             8   lx26-amd64
bowser                            2   lx26-amd64
peach                             4   lx26-amd64
sphynx                            8   lx26-amd64
wario                             4   lx26-amd64
===============================================
SUM                              26
# run -c 26 qsub /usr/local/sge/examples/jobs/sgetest.sh; watch -
n1 -d qstat
```

   c. After all the jobs finish take a look at the results:

```
# for i in beebo bowser peach sphynx wario;do ssh $i "cat
sgetest.sh.o* | grep $i"; done
beebo:11935           78.378009295
beebo:11985           78.971428629
beebo:12008           78.805859211
beebo:11991           78.130127030
beebo:12020           78.502641933
beebo:12004           78.239523554
beebo:12010           79.107759520
beebo:12023           78.212307762
bowser:11075          59.489210727
bowser:11086          59.538531448
peach:11110           90.108337377
peach:11148           90.139130453
peach:11144           89.494598956
peach:11142           90.234077756
sphynx:14412          25.978482245
sphynx:14450          26.241343088
```

```
sphynx:14440            25.983305171
sphynx:14478            26.255011607
sphynx:14483            26.285700584
sphynx:14491            25.986674634
sphynx:14480            26.999872530
sphynx:14471            27.040071846
wario:10962             20.784953434
wario:10970             20.797401419
wario:10995             20.780276036
wario:10986             20.874604744
```

d. Now cleanup the output files:

```
# for i in beebo bowser peach sphynx wario;do ssh $i “rm -f
sgetest.sh.*”;done
# for i in beebo bowser peach sphynx wario;do ssh $i “ls
sgetest.sh.*”;done
```

Continue Exploring

1. Refer to the Grid Engine PDFs to master fine tuning and using Grid Engine. Refer to “Grid Engine Documentation” on page 2-3.
2. Learn **qmon**, the SGE GUI. Usage for **qmon** can be found in the Grid Engine PDFs. You will want to enable X11 forwarding if you **ssh** into the master host:

   \# **ssh -X -l root** *master host hostname*

   Start the GUI on the master host:

   \# **qmon&**
3. See “Man Pages” on page 2-3 for a complete list of the man pages supplied with sge.

A
Node Information Worksheet

Use this worksheet to record information about the master and nodes that will compose your cluster. Refer to the section “Configuring the Cluster Network Using cm-cfgboot” in Chapter 1 for more information.

Master Host

Cluster subnet	. . .
Cluster netmask	. . .
Broadcast address	. . .
Routers	. . .
Domain name servers	. . .
Domain name	

Cluster Nodes

Hostname	MAC Address	IP Address	Node Type
	: : : : :	. . .	
	: : : : :	. . .	
	: : : : :	. . .	
	: : : : :	. . .	
	: : : : :	. . .	
	: : : : :	. . .	
	: : : : :	. . .	
	: : : : :	. . .	
	: : : : :	. . .	

Index