Sun HPC ClusterTools™ 4 Administrator's Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 U.S.A.
650-960-1300

Part No. 816-0649-10
August 2001, Revision A

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Contents

Preface

The *Sun HPC ClusterTools Administrator's Guide* explains how to configure and manage a cluster of one or more Sun™ servers running the Sun HPC Cluster Runtime Environment (CRE) job management software. These instructions are designed for an experienced system administrator with networking knowledge.

An alternative to CRE is the Load-Sharing Facility (LSF) resource-management suite from Platform Computing Corporation. If your cluster is configured to run Sun HPC ClusterTools with the LSF suite, you should consult Appendix C of this manual along with the LSF system administration documentation.

Using Solaris Commands

This document may not contain information on basic Solaris™ commands and procedures such as shutting down the system, booting the system, and configuring devices.

See one or both of the following for this information:

- AnswerBook2™ online documentation for the Solaris software environment
- Other software documentation that you received with your system

Typographic Conventions

Typeface or Symbol	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename*.

Shell Prompts

Shell	Prompt
C shell	%
C shell superuser	#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Related Sun Documentation

Application	Title	Part Number
All	*Read Me First: Guide to Sun HPC ClusterTool 4 Documentation*	816-0646-10
All	*Sun HPC ClusterTools 4 Product Notes*	816-0647-10
Installation	*Sun HPC ClusterTools 4 Installation Guide*	816-0648-10
ClusterTools Usage	*Sun HPC ClusterTools 4 User's Guide*	816-0650-10
Sun MPI Programming	*Sun MPI 5.0 Programming and Reference Guide*	816-0651-10
Sun S3L Programming	*Sun S3L 4.0 Programming Guide*	816-0652-10
Sun S3L Programming	*Sun S3L 4.0 Reference Manual*	816-0653-10
Performance Programming	*Sun HPC ClusterTools 4 Performance Guide*	816-0656-10
Prism Environment	*Prism 6.2 User's Guide*	816-0654-10
Prism Environment	*Prism 6.2 Reference Manual*	816-0655-10

Accessing Sun Documentation Online

The `docs.sun.com`SM web site enables you to access a select group of Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

`http://docs.sun.com`

Ordering Sun Documentation

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at:

`http://www.fatbrain.com/documentation/sun`

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can email your comments to Sun at:

`docfeedback@sun.com`

Please include the part number of your document in the subject line of your email.

CHAPTER 1

Introduction

The Sun HPC Cluster Runtime Environment (CRE) is a program execution environment that provides basic job launching and load-balancing capabilities.

This manual provides information needed to administer Sun HPC clusters on which MPI programs run under CRE. The topics covered are organized in the following manner:

- Chapter 2 provides quick-start instructions for getting an MPI job running on a Sun HPC cluster with newly installed Sun HPC ClusterTools software.
- Chapter 3 provides an introduction to configuring a Sun HPC cluster, using both the administration command interface, `mpadmin`, and the cluster configuration file, `hpc.conf`.
- Chapter 4 explains how to set up and manage PFS file systems. If your cluster does not implement PFS file systems, ignore this chapter.
- Chapter 5 discusses various considerations that can influence how a Sun HPC cluster should be configured.
- Chapter 6 provides a more comprehensive description of `mpadmin` features.
- Chapter 7 provides a more comprehensive description of the `hpc.conf` configuration file.
- Chapter 8 provides guidelines for performing routine maintenance and for recognizing and troubleshooting error conditions.
- Appendix A describes the Cluster Console Manager (CCM), a set of cluster administration tools.
- Appendix B describes the procedure for installing Sun HPC ClusterTools software from the command line rather than by means of the graphical installation interface supplied with the ClusterTools software.
- Appendix C describes plugins that enable HPC ClusterTools software to run on the Load-Sharing Facility (LSF) Suite from Platform Computing Corporation.
- Appendix D describes the procedure for establishing a loose integration between HPC ClusterTools and Sun Grid Engine (SGE), such that MPI programs can execute under SGE control.

Note – If your Sun HPC cluster is configured to use the LSF suite instead of CRE, please consult the *LSF Batch System Administration's Guide* (supplied by Platform Computing), along with Appendix C of this manual.

The balance of this chapter provides an overview of the Sun HPC ClusterTools software and the Sun HPC cluster hardware on which it runs.

Sun HPC Clusters

A Sun HPC hardware configuration can be a single Sun SMP (symmetric multiprocessor) server or multiple SMPs interconnected into a cluster. HPC ClusterTools software supports parallel jobs of up to 2048 processes per job running on clusters of up to 64 nodes.

Note – An individual SMP server within a Sun HPC cluster is referred to as a *node*.

Sun HPC clusters can also be built using any Sun-supported TCP/IP interconnect, such as Ethernet, high-speed Ethernet, Gigabit Ethernet, ATM OC-3, ATM OC-12, FDDI, and HiPPI.

Any Sun HPC node that is connected to a disk storage system can be configured as a Parallel File System (PFS) I/O server.

Cluster Runtime Environment

CRE comprises two sets of daemons—the master daemons and the nodal daemons. These two sets of daemons work cooperatively to maintain the state of the cluster and manage program execution.

The master daemons consist of the daemons `tm.rdb`, `tm.mpmd`, and `tm.watchd`. They run on one node exclusively, which is called the *master node*. There are two nodal daemons, `tm.omd` and `tm.spmd`. They run on all the nodes.

Sun HPC ClusterTools Software

Sun HPC ClusterTools software is an integrated suite of parallel development tools that extend Sun's network computing solutions to high-end distributed-memory applications.

The Sun HPC ClusterTools products can be used either in the Cluster Runtime Environment or with the LSF Suite, Platform Computing Corporation's resource management software, extended with parallel support. To determine which resource manager is used in an existing installation, execute the command `hpc_rte`. It displays `lsf`, or `cre`, or `none` (if neither set of daemons is running).

Sun HPC ClusterTools components run under the Solaris 8 (32-bit or 64-bit) Operating Environment.

Sun MPI and MPI I/O

Sun MPI is a highly optimized version of the Message-Passing Interface (MPI) communications library. Sun MPI implements all the MPI 1.2 standard as well as a significant subset of the MPI 2.0 feature list. In addition, Sun MPI provides extensions such as support for multithreaded programming, MPI I/O support for parallel file I/O, and others as detailed in the Sun MPI documentation.

Sun MPI provides full F77, C, and C++ support and basic F90 support.

Loadable Protocol Modules

The Sun MPI library is capable of providing high-performance communications over several different protocols. HPC ClusterTools makes three protocols available to MPI programs: Shared Memory (SHM), Transport Control Protocol (TCP), and Remote Shared Memory (RSM).

Protocols are provided as dynamically loaded library modules, separate from the MPI library. The cluster administrator determines which protocols are available on a cluster and their relative priorities. The user need not be concerned with the details of any protocol underlying MPI communications.

Parallel File System

Sun HPC ClusterTools software's Parallel File System (PFS) component provides high-performance file I/O for multiprocess applications running in a cluster-based, distributed-memory environment.

PFS file systems closely resemble UFS file systems, but provide significantly higher file I/O performance by striping files across multiple PFS I/O server nodes.

Prism Environment

The Prism™ graphical programming environment allows you to develop, execute, debug, profile, and visualize data in message-passing programs.

The Prism environment can be used with applications written in F77, F90, C, and C++.

Sun S3L

The Sun Scalable Scientific Subroutine Library (Sun S3L) provides a set of parallel and scalable functions and tools that are used widely in scientific and engineering computing. It is built on top of MPI.

Sun S3L routines can be called from applications written in F77, F90, C, and C++.

Related Tools

Sun HPC ClusterTools provides or makes use of several related tools, including Sun compilers and the Cluster Console Manager.

Sun Compilers

The Sun HPC ClusterTools 4 release supports the Sun Forte™ 6 Developer compilers. These compilers support Fortran 77, Fortran 90, C, and C++.

Cluster Console Manager

The Cluster Console Manager is a suite of applications (`cconsole`, `ctelnet`, and `crlogin`) that simplify cluster administration by enabling you to initiate commands on all nodes in the cluster simultaneously. Any command entered in the CCM's master window is broadcast to all the nodes in the cluster.

These applications are described in Appendix A "Cluster Console Manager Tools" of this manual.

CHAPTER 2

Getting Started

This chapter introduces CRE and the basic procedures required to get a Sun HPC cluster ready for use. These basic procedures include starting the CRE daemons and testing the cluster's readiness. This chapter also describes the procedure for shutting down CRE.

The topics covered in this chapter include

This chapter assumes that the ClusterTools software, including CRE, has been correctly installed and configured, as described in the *Sun HPC ClusterTools Installation Guide*.

Notice that the system verification procedures outlined here are identical to the "post-install" procedures recommended for CRE-based clusters in the *Sun HPC ClusterTools Installation Guide*.

Fundamental CRE Concepts

This section introduces some important concepts that you should understand in order to administer the Sun HPC ClusterTools software with CRE.

Cluster of Nodes

As its name implies, the Sun Cluster Runtime Environment is intended to operate in a Sun HPC cluster—that is, in a collection of Sun symmetric multiprocessor (SMP) servers that are connected by any Sun-supported TCP/IP-capable interconnect. An SMP attached to the cluster network is referred to as a *node*.

CRE manages the launching and execution of both serial and parallel jobs on the cluster nodes, which are grouped into logical sets called *partitions*. (See the next section for more information about partitions.) For serial jobs, its chief contribution is to perform load-balancing in shared partitions, where multiple processes may be competing for the same node resources. For parallel jobs, CRE provides:

- A single job-monitoring and control point
- Load-balancing for shared partitions
- Information about node connectivity
- Support for spawning of MPI processes
- Support for Prism interaction with parallel jobs

Note – A "cluster" can consist of a single Sun SMP server. However, executing MPI jobs on even a single-node cluster requires CRE to be running on that cluster.

CRE supports parallel jobs of up to 2048 processes running on clusters of up to 64 nodes.

Security

A Sun HPC cluster may be protected from unauthorized use by means of the standard Solaris authentication `AUTH_SYS` or by installing a third-party authentication product. HPC ClusterTools 4 supports both Kerberos Version 5 and Data Encryption System (DES).

In addition to one (or none) of the above authentication methods, CRE provides basic security by means of a cluster password. It operates by checking the credentials of programs that request access.

The system administrator establishes the cluster password on each node of the cluster and on any outside nodes that may access the cluster. The password should be customized immediately after installing ClusterTools software, as described in the installation instructions.

Partitions

The system administrator configures the nodes in a Sun HPC cluster into one or more logical sets, called *partitions.* A job is always launched on a predefined partition that is currently *enabled*, or accepting jobs. A job will run on one or more nodes in that partition, but not on nodes in any other enabled partition.

Note – The CPUs in certain Sun high-end servers, such as the Enterprise™ 10000 server, can be configured into logical "nodes," referred to as *domains*. These domains can be logically grouped to form partitions, which CRE uses in the same way it deals with partitions containing other types of Sun HPC nodes.

Partitioning a cluster allows multiple jobs to execute concurrently, without the risk that jobs on different partitions will interfere with each other. This ability to isolate jobs can be beneficial in various ways. For example:

- If one job requires exclusive use of a set of nodes but other jobs need to execute at the same time, the availability of two partitions in a cluster allows both needs to be satisfied.
- If a cluster contains a mix of nodes whose characteristics differ—such as having different memory sizes, CPU counts, or levels of I/O support—the nodes can be grouped into partitions that have similar resources. Jobs that require particular resources can then be run on suitable partitions, while jobs that are less resource-dependent can be relegated to less specialized partitions.

The system administrator can selectively enable and disable partitions. Jobs can be executed only on enabled partitions. This restriction makes it possible to define many partitions in a cluster but have only a few active at any one time.

In addition to enabling and disabling partitions, the system administrator can set and unset other partition attributes that influence various aspects of how the partition functions.

It is possible for nodes in a cluster not to belong to a currently enabled partition. If a user logs in to one of these "independent" nodes and does not request a particular partition for a job, CRE launches that user's job on the cluster's *default partition*. It is also possible for a node to belong to more than one partition, so long as only one is enabled at a time.

Note – Although a job cannot be run across partition boundaries, it can be run on a partition plus independent nodes. See the *Sun HPC ClusterTools User's Guide* for information.

Load Balancing

CRE load-balances programs that execute in partitions where multiple jobs are running concurrently.

When a user launches a job in such a shared partition, CRE first determines what criteria (if any) have been specified for the node or nodes on which that program is to run. It then determines which nodes within the partition meet these criteria. If more nodes meet the criteria than are required to run the program, CRE starts the program on the node or nodes that are least loaded. It examines the one-minute load averages of the nodes and ranks them accordingly.

Jobs and Processes

When a serial program executes on a Sun HPC cluster, it becomes a Solaris process with a Solaris *process ID*, or *pid*.

When CRE executes a distributed message-passing program it spawns multiple Solaris processes, each with its own pid.

CRE also assigns a *job ID*, or *jid*, to the program. If it is an MPI job, the jid applies to the overall job. Job IDs always begin with a `j` to distinguish them from pids. Many CRE commands take jids as arguments. For example, you can issue an `mpkill` command with a signal number or name and a jid argument to send the specified signal to all processes that make up the job specified by the jid.

Communication Protocols

The communication protocol modules to be used by MPI jobs are loaded at job startup. HPC ClusterTools software is provided with a default configuration of the protocols SHM, TCP, and RSM, along with their relative preference rankings. The

cluster administrator need not take any action to make protocols available. (The default configuration can be changed by editing the HPC ClusterTools configuration file `hpc.conf`.)

Parallel File System

From the user's perspective, PFS file systems closely resemble UNIX file systems. PFS uses a conventional inverted-tree hierarchy, with a `root` directory at the top and subdirectories and files branching down from there. The fact that individual PFS files are distributed across multiple disks managed by multiple I/O servers is transparent to the programmer. The way that PFS files are actually mapped to the physical storage facilities is determined by the configuration information the administrator has supplied in the Sun ClusterTools configuration file `hpc.conf`.

Starting the CRE Daemons

After installation or after a system shut-down, you need to start the CRE daemons manually to bring up the ClusterTools software. Start the CRE daemons first on the cluster's master node and then on all the other nodes of the cluster.

To find which node is the cluster's master node, look in the `hpc_config` file at the line `MASTER_NODE="`*hostname*`"`. This file was used in the ClusterTools installation process and resides in a directory selected by the administrator at install time.

Note – The `hpc_config` file is distinct from the `hpc.conf` file. `hpc_config` is created at installation time. It stores the configuration information that the installer has specified, including the name of the CRE master node. `hpc.conf` is provided with the software and contains the settings of various system parameters.

- **On the master node as superuser, start the CRE master daemons.**

  ```
  # /etc/init.d/sunhpc.cre_master start
  ```

- **On all the other nodes in the cluster as superuser, start the CRE nodal daemons.**

  ```
  # /etc/init.d/sunhpc.cre_node start
  ```

  You may want to use one of the Cluster Console Manager (CCM) tools for this step, if available. CCM enables you to broadcast a command to all the nodes from a single entry. See Appendix A for instructions on using the CCM tools.

Verifying Basic Functionality

To test the cluster's ability to perform basic operations, you should check that all daemons are running, set the cluster password (unless it was already set at install time), create a default partition, and run a simple job. This section explains how to perform these steps.

Note – You need to have `/opt/SUNWhpc/bin` in your path for many of the following procedures.

Check That Nodes Are Up

Run `mpinfo -N` to display information about the cluster nodes. The following is an example of `mpinfo -N` output for a two-node system:

```
% mpinfo -N
NAME   UP  PARTITION  OS     OSREL  NCPU  FMEM   FSWP   LOAD1  LOAD5  LOAD15
host1  y   -          SunOS  5.8    1      7.17  74.76  0.03   0.04   0.05
host2  y   -          SunOS  5.8    1     34.70  38.09  0.06   0.02   0.02
```

If any nodes are missing from the list or do not have a `y` (yes) entry in the `UP` column, restart their nodal daemons as described in "Starting the CRE Daemons" on page 11.

Create a Default Partition

A partition is a logical group of nodes that cooperate in executing an MPI program. You can create a cluster-wide partition by running the initialization script named `part_initialize` on any node in the cluster. This superuser script resides by default in `/opt/SUNWhpc/bin`.

```
# /opt/SUNWhpc/bin/part_initialize
```

This action creates a single partition named `all`, which includes all the nodes in the cluster as members. The `all` partition can be used in the subsequent verification tests.

Then, run `mpinfo -N` again to verify the successful creation of `all`. See below for an example of `mpinfo -N` output when the `all` partition is present.

```
# /opt/SUNWhpc/bin/part_initialize
# mpinfo -N
NAME   UP  PARTITION  OS    OSREL  NCPU  FMEM   FSWP   LOAD1   LOAD5   LOAD15
node1  y   all        SunOS 5.8    1     7.17   74.76  0.03    0.04    0.05
node2  y   all        SunOS 5.8    1     34.69  38.08  0.00    0.00    0.01
```

Verify That CRE Executes Jobs

Verify that CRE can launch jobs on the cluster. For example, use the `mprun` command to execute the program `hostname` on all the nodes in the cluster, as shown below:

```
# mprun –Ns –np 0 hostname
node1
node2
```

`mprun` is the CRE command that launches jobs. The combination of `–Ns` and `–np 0` ensures that CRE will start one `hostname` process on each node. See the `mprun` man page for descriptions of `–Ns`, `–np`, and the other `mprun` options. In this example, the cluster contains two nodes, `node1` and `node2`, each of which returns its host name.

Note – CRE does not sort or rank the output of `mprun` by default, so host name ordering may vary from one run to another.

Verifying MPI Communications

You can verify MPI communications by running a simple MPI program.

The MPI program must have been compiled by one of the compilers supported by Sun HPC ClusterTools software (listed in “Sun Compilers” on page 4).

Two simple Sun MPI programs are available in `/opt/SUNWhpc/examples/mpi`:

- `connectivity.c` – A C program that checks the connectivity among all processes and prints a message when it finishes

- `monte.f` – A Fortran program in which each process participates in calculating an estimate of π using a Monte-Carlo method

See the `Readme` file in the same directory; it provides instructions for using the examples. The directory also contains the make file, `Makefile`. The full text of both code examples is also included in the *Sun MPI Programming and Reference Guide*.

Stopping and Restarting CRE

If you want to shut down the entire cluster with the least risk to your file systems, use the Solaris `shutdown` command.

However, if you prefer to stop and restart CRE without shutting down the entire cluster, CRE supports a pair of scripts that simplify this process:

- `sunhpc.cre_master` – Use this command to stop or start the CRE master daemons.
- `sunhpc.cre_node` – Use this command to stop or start the CRE nodal daemons.

Stopping CRE

To shut down the CRE daemons without shutting down the Solaris Operating Environment, execute the following commands as superuser:

- **Stop all the nodal daemons by executing the following on all the nodes.**

  ```
  # /etc/init.d/sunhpc.cre_node stop
  ```

  You can simplify this step by using one of the CCM tools (`cconsole`, `ctelnet`, or `crlogin`) to broadcast the following command entered on the master node to all the other nodes. See Appendix A for information about the CCM tools.

- **Stop the master daemons by executing the following on the master node.**

  ```
  # /etc/init.d/sunhpc.cre_master stop
  ```

Restarting CRE

To restart CRE without rebooting the Solaris operating environment, execute the following commands on the master node:

- **Start the master daemons.**

  # **/etc/init.d/sunhpc.cre_master start**

- **Start the nodal daemons.**

  # **/etc/init.d/sunhpc.cre_node start**

Note – Always bring up the master daemons first (before the nodal daemons). Otherwise, the nodal daemons will not be initialized properly and CRE will not work.

When the `sunhpc.cre_master` and `sunhpc.cre_node` programs are executed with `start` commands, they both initiate a `stop` command on all currently running daemons before starting the CRE daemons.

CHAPTER 3

Overview of Administration Controls

The Sun HPC cluster's default configuration supports execution of MPI applications. In other words, if you have started the CRE daemons on your cluster and created a default partition, as described in Chapter 2, users can begin executing MPI jobs.You may, however, want to customize the cluster's configuration to the specific requirements of your site.

This chapter provides a brief overview of the features that control a cluster's configuration and behavior. These are:

- The CRE daemons - introduced here and described more fully in their respective man pages
- The RSM daemon
- The `mpadmin` command - introduced here and described more fully in Chapter 6, as well as in its man page
- The cluster configuration file `hpc.conf` – introduced here and described more fully in Chapter 7, as well as in its man page
- Authentication methods

The CRE Daemons

CRE comprises three master daemons and two nodal daemons:

- CRE master daemons
  - `tm.rdb`
  - `tm.mpmd`
  - `tm.watchd`

- CRE nodal daemons
  - `tm.omd`
  - `tm.spmd`

This section present brief descriptions of the CRE daemons. For complete information on the daemons, see their respective man pages.

Master Daemon `tm.rdb`

`tm.rdb` is the *resource database daemon*. It runs on the master node and implements the resource database used by the other parts of CRE. This database represents the state of the cluster and the jobs running on it.

If you make changes to the cluster configuration, for example, if you add a node to a partition, you must restart the `tm.rdb` daemon to update the CRE resource database to reflect the new condition.

Master Daemon `tm.mpmd`

`tm.mpmd` is the *master process-management daemon*. It runs on the master node and services user (client) requests made via the `mprun` command. It also interacts with the resource database via calls to `tm.rdb` and coordinates the operations of the nodal client daemons.

Master Daemon `tm.watchd`

`tm.watchd` is the cluster *watcher daemon*. It runs on the master node and monitors the states of cluster resources and jobs and, as necessary:

- Marks individual nodes as online or offline by periodically executing remote procedure calls (RPCs) to all the nodes.
- Clears stale resource database (`rdb`) locks.
- If the `-Yk` option has been enabled, aborts jobs that have processes on nodes determined to be down. This option is disabled by default.

Nodal Daemon `tm.omd`

`tm.omd` is the *object-monitoring daemon*. It runs on all the nodes in the cluster, including the master node, and continually updates the CRE resource database with dynamic information concerning the nodes, most notably their load. It also initializes the database with static information about the nodes, such as their host names and network interfaces, when CRE starts up.

The environment variable `SUNHPC_CONFIG_DIR` specifies the directory in which the CRE resource database files are to be stored. The default is `/var/hpc`.

Nodal Daemon `tm.spmd`

`tm.spmd` is the *slave process-management daemon*. It runs on all the compute nodes of the cluster and, as necessary:

- Handles spawning and termination of nodal processes per requests from the `tm.mpmd`.
- In conjunction with `mprun`, handles multiplexing of `stdio` streams for nodal processes.
- Interacts with the resource database via calls to `tm.rdb`.

RSM Daemon

The `hpc_rsmd` daemon provides access services to Remote Shared Memory (RSM) resources and manages RSM communications paths on behalf of MPI jobs. If your cluster is not RSM-enabled, you can ignore this section.

An instance of `hpc_rsmd` runs on each RSM-enabled node of a cluster and:

- Provides remote shared-memory resources to MPI client processes.
- Monitors and manages the RSM paths between node pairs.
- Returns cluster configuration information to CRE.

`hpc_rsmd` is started when the cluster is booted. After boot, the daemon can be stopped and restarted manually by means of the following script (executed by superuser). This script can also be used to clean up files and System V shared memory segments left behind when an `hpc_rsmd` instance exits abnormally.

```
# /etc/init.d/sunhpc.hpc_rsmd  [ start | stop | clean ]
```

mpadmin: Administration Interface

CRE provides an interactive command interface, mpadmin, which you can use to administer your Sun HPC cluster. It can only be invoked by the superuser.

This section introduces mpadmin and shows how to use it to perform several administrative tasks:

- List the names of all nodes in the cluster
- Enable nodes
- Create and enable partitions
- Customize some aspects of cluster administration

mpadmin offers many more capabilities than are described in this section. See Chapter 6 for a more comprehensive description of mpadmin.

Introduction to mpadmin

The mpadmin command has the following syntax:

mpadmin [**-c** *command*] [**-f** *filename*] [**-h**] [**-q**] [**-s** *cluster_name*] [**-V**]

When you invoke mpadmin with no options, it goes into interactive mode, displaying an mpadmin prompt. It also goes into interactive mode when invoked with the options -f, -q, or -s. In this mode, you can execute any number of mpadmin subcommands to perform operations on the cluster or on nodes or partitions.

When you invoke mpadmin with the -c, -h, or -V options, it performs the requested operation and returns to the shell level.

The mpadmin command-line options are summarized in TABLE 3-1.

TABLE 3-1 mpadmin Options

Option	Description
-c *command*	Execute single specified command.
-f *file-name*	Take input from specified file.
-h	Display help/usage text.

TABLE 3-1 `mpadmin` Options

Option	Description
`-q`	Suppress the display of a warning message when a non-root user attempts to use restricted command mode.
`-s` *cluster-name*	Connect to the specified Sun HPC cluster.
`-V`	Display `mpadmin` version information.

Commonly Used `mpadmin` Options

This section describes the `mpadmin` options `-c`, `-f`, and `-s`.

`-c` *command – Execute a Single Command*

Use the `-c` option when you want to execute a single `mpadmin` command and return upon completion to the shell prompt. For example, the following use of `mpadmin -c` changes the location of the CRE log file to `/home/wmitty/cre_messages`:

```
# mpadmin -c set logfile="/home/wmitty/cre_messages"
#
```

Most commands that are available via the interactive interface can be invoked via the `-c` option. See Chapter 6 for a description of the `mpadmin` command set and a list of which commands can be used as arguments to the `-c` option.

`-f` *file-name – Take Input From a File*

Use the `-f` option to supply input to `mpadmin` from the file specified by the *file-name* argument. The source file is expected to consist of one or more `mpadmin` commands, one command per line.

This option can be particularly useful in the following ways:

- It can be used following use of the `mpadmin` command `dump`, which outputs all or part of a cluster's configuration in the form of an `mpadmin` script. If the `dump` output is stored in a file, `mpadmin` can, at a later time, read the file via the `-f` option, thereby reconstructing the configuration that had been saved in the `dump` output file.
- The `-f` option can also be used to read `mpadmin` scripts written by the system administrator—scripts designed to simplify other cluster management tasks that involve issuing a series of `mpadmin` commands.

`-s` *cluster-name – Connect to Specified Cluster*

Use the `-s` option to connect to the cluster specified by the *cluster-name* argument. A cluster's name is the host name of the cluster's master node.

The `mpadmin` commands apply to a certain cluster, determined as follows:

- First, the cluster specified on the command line with the `-s` option.
- If no such option is specified, the command uses the value of the environment variable `SUNHPC_CLUSTER`.
- If this environment variable is not set, the command uses the cluster name supplied (usually at installation time) in `/etc/hpc_system`.

Understanding Objects, Attributes, and Contexts

To use `mpadmin`, you need to understand the concepts of *object*, *attribute*, and *context* as they apply to `mpadmin`.

Objects and Attributes

From the perspective of `mpadmin`, a Sun HPC cluster consists of a system of objects, which include

- The cluster itself
- Each node contained in the cluster
- Each partition (logical group of nodes) defined in the cluster

Each type of object has a set of attributes, which control various aspects of their respective objects. For example, a node's `enabled` attribute can be

- `set` to make the node available for use
- `unset` to prevent it from being used

Some attribute values can be operated on via `mpadmin` commands.

Note – CRE sets many attributes in a cluster to default values each time it starts up. You should not change attribute values, except for the attribute modifications described here and in Chapter 6.

Contexts

`mpadmin` commands are organized into three *contexts*, which correspond to the three types of `mpadmin` objects. These contexts are summarized below and illustrated in FIGURE 3-1.

- Cluster – These commands affect cluster attributes.
- Node – These commands affect node attributes.
- Partition – These commands affect partition attributes.

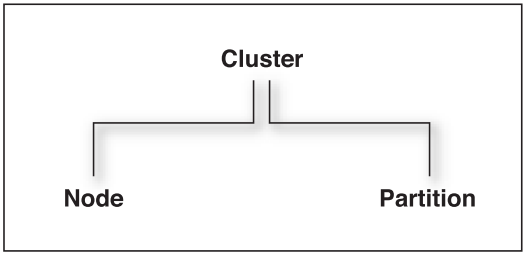

FIGURE 3-1 `mpadmin` Contexts

`mpadmin` Prompts

In interactive mode, the `mpadmin` prompt contains one or more fields that indicate the current context. TABLE 3-2 shows the prompt format for each of the possible `mpadmin` contexts.

TABLE 3-2 `mpadmin` Prompt Formats

Prompt Formats	Context
[*cluster-name*]::	Current context = Cluster
[*cluster-name*]Node::	Current context = Node, but not a specific node
[*cluster-name*]N(*node-name*)::	Current context = a specific node
[*cluster-name*]Partition::	Current context = Partition, but not a specific partition
[*cluster-name*]P(*partition-name*)::	Current context = a specific partition

Performing Sample `mpadmin` Tasks

To introduce the use of `mpadmin`, this section steps through some common tasks the administrator may want to perform. These tasks are:

- List names of nodes
- Enable nodes so that MPI jobs can run on them
- Create and enable partitions so that MPI jobs can run on them
- Customize some cluster attributes

List Names of Nodes

`mpadmin` provides various ways to display information about the cluster and many kinds of information that can be displayed. However, the first information you are likely to need is a list of the nodes in your cluster.

Use the `list` command in the `Node` context to display this list. In the following example, `list` is executed on `node1` in a four-node cluster.

```
node1# mpadmin
[node0]:: node
[node0] Node:: list
    node0
    node1
    node2
    node3
[node0] Node::
```

The `mpadmin` command starts up an `mpadmin` interactive session in the `Cluster` context. This is indicated by the `[node0]::` prompt, which contains the cluster name, `node0`, and no other context information.

Note – A cluster's name is assigned by CRE and is always the name of the cluster's master node.

The `node` command on the example's second line makes `Node` the current context. The `list` command displays a list of all the nodes in the cluster.

Once you have this list of nodes, you have the information you need to enable the nodes and to create a partition. However, before moving on to those steps, you might want to try listing information from within the cluster context or the partition context. In either case, you would follow the same general procedure as for listing nodes.

If this is a newly installed cluster and you have not already run the `part_initialize` script (as described in "Create a Default Partition" on page 8), the cluster contains no partitions. If, however, you *did* run `part_initialize` and have thereby created the partition `all`, you might want to perform the following test.

```
node1# mpadmin
[node0]:: partition
[node0] Partition:: list
    all
[node0] Partition::
```

To see what nodes are in partition `all`, make `all` the current context and execute the `list` command. The following example illustrates this; it begins in the `Partition` context (where the previous example ended).

```
[node0] Partition:: all
[node0] P[all]:: list
    node0
    node1
    node2
    node3
[node0] P[all]::
```

Enabling Nodes

A node must be in the enabled state before MPI jobs can run on it.

Note that enabling a partition automatically enables all its member nodes, as described in the next section.

To enable a node manually, make that node the current context and set its `enabled` attribute. Repeat for each node that you want to be available for running MPI jobs.

The following example illustrates this, using the same four-node cluster used in the previous examples.

```
node1# mpadmin
[node0]:: node0
[node0] N[node0]:: set enabled
[node0] N[node0]:: node1
[node0] N[node1]:: set enabled
[node0] N[node1]:: node2
[node0] N[node2]:: set enabled
[node0] N[node2]:: node3
[node0] N[node3]:: set enabled
[node0] N[node3]::
```

Note the use of a shortcut to move directly from the `Cluster` context to the `node0` context without first going to the general `Node` context. You can explicitly name a particular object as the target context in this way so long as the name of the object is unambiguous—that is, it is not the same as an `mpadmin` command.

`mpadmin` accepts multiple commands on the same line. The previous example could be expressed more succinctly as:

```
node1# mpadmin
[node0]:: node0 set enabled node1 set enabled node2 set enabled node3
set enabled
[node0] N[node3]::
```

To disable a node, use the `unset` command in place of the `set` command.

Creating and Enabling Partitions

You must create at least one partition and enable it before you can run MPI programs on your Sun HPC cluster. Even if your cluster already has the default partition `all` in its database, you will probably want to create other partitions with different node configurations to handle particular job requirements.

There are three essential steps involved in creating and enabling a partition:

- Use the `create` command to assign a name to the partition.
- Set the partition's `nodes` attribute to a list of the nodes you want to include in the partition.
- Set the partition's `enabled` attribute, which automatically enables all the partition's member nodes.

Once a partition is created and enabled, you can run serial or parallel jobs on it. A serial program runs on a single node of the partition. Parallel programs are distributed to as many nodes as CRE determines appropriate for the job.

Note – There are no restrictions on the number or size of partitions, so long as no node is a member of more than one enabled partition.

Example: Creating a Two-Node Partition

The following example creates and enables a two-node partition named `part0`. It then lists the member nodes to verify the success of the creation.

```
node1# mpadmin
[node0]:: partition
[node0] Partition:: create part0
[node0] P[part0]:: set nodes=node0 node1
[node0] P[part0]:: set enabled
[node0] P[part0]:: list
    node0
    node1
[node0] P[part0]::
```

Example: Two Partitions Sharing a Node

The next example shows a second partition, `part1`, being created. One of its nodes, `node1`, is also a member of `part1`.

```
[node0] P[part0]:: up
[node0] Partition:: create part1
[node0] P[part1]:: set nodes=node1 node2 node3
[node0] P[part1]:: list
    node1
    node2
    node3
[node0] P[part1]::
```

Because `node1` is shared with `part0`, which is already enabled, `part1` is not being enabled at this time. This illustrates the rule that a node can be a member of more than one partition, but only one of those partitions can be enabled at a time.

Note the use of the `up` command. The `up` command moves the context up one level, in this case, from the context of a particular partition (that is, from `part0`) to the general `Partition` context.

Example: Shared versus Dedicated Partitions

CRE can configure a partition to allow multiple MPI jobs to be running on it concurrently. Such partitions are referred to as *shared* partitions. CRE can also configure a partition to permit only one MPI job to run at a time. These are called *dedicated* partitions.

In the following example, the partition `part0` is configured to be a dedicated partition and `part1` is configured to allow shared use by up to four processes.

```
node1# mpadmin
[node0]:: part0
[node0] P[part0]:: set max_total_procs=1
[node0] P[part0]:: part1
[node0] P[part1]:: set max_total_procs=4
[node0] P[part1]::
```

The `max_total_procs` attribute defines how many processes can be active on each node in the partition for which it is being set. In this example, it is set to `1` on `part0`, which means only one process can be running at a time. It is set to `4` on `part1` to allow up to four processes to run on that partition.

Note again that the context-changing shortcut (introduced in "Enabling Nodes" on page 20) is used in the second and fourth lines of this example.

Customizing Cluster Attributes

Two cluster attributes that you may wish to modify are `logfile` and `administrator`.

Changing the `logfile` *Attribute*

The `logfile` attribute allows you to log CRE messages in a separate file from all other system messages. For example, if you enter

```
[node0]:: set logfile=/home/wmitty/cre-messages
```

CRE will output its messages to the file `/home/wmitty/cre-messages`. If `logfile` is not set, CRE messages will be passed to `syslog`, which will store them with other system messages in `/var/adm/messages`.

Note – A full path name must be specified when setting the `logfile` attribute.

Changing the `administrator` *Attribute*

You can set the `administrator` attribute to specify, say, the email address of the system administrator. To do this:

```
[node0]:: set administrator="root@example.com"
```

Note the use of double quotes.

Quitting `mpadmin`

Use either the `quit` or `exit` command to quit an `mpadmin` interactive session. Either causes `mpadmin` to terminate and return control to the shell level.

For example:

```
[node0]:: quit
node1#
```

Cluster Configuration File `hpc.conf`

When CRE starts up, it updates portions of the resource database according to the contents of a configuration file named `hpc.conf`. This file is organized into functional sections, which are summarized here and illustrated in TABLE 3-3.

- `ShmemResource` section defines certain shared memory attributes.
- `MPIOptions` section defines certain MPI parameters.
- `CREOptions` section controls the logging of system events, the handling of daemon core files, and the authentication options.
- `PMODULES` section names and locates the protocol modules that are available for communication on the cluster.
- `PM` sections list and rank the protocol modules that are available for communication on the cluster.
- `PFSFileSystem` section names and defines all parallel file systems in the cluster.
- `PFSServers` section names and defines all parallel file system servers in the cluster.
- `HPCNodes` section is not used by CRE. It applies only in an LSF-based runtime environment (as described in Appendix C).

You can change any of these aspects of your cluster's configuration by editing the corresponding parts of the `hpc.conf` file. Default settings are in effect if you make no changes to the `hpc.conf` file as provided.

To illustrate the process of customizing the `hpc.conf` file, this section explains how to:

- Prepare for editing `hpc.conf`
- Create one or more I/O servers for the PFS file systems
- Create PFS file systems
- Control MPI communication attributes
- Update the CRE database

Note – The `hpc.conf` file is provided with the Sun HPC ClusterTools software. It is distinct from the `hpc_config` file, which is created by the installer at installation time.

Note – You may never need to make any changes to `hpc.conf`. If you do wish to make changes beyond those described in this section, see Chapter 7 for a fuller description of this file.

TABLE 3-3 General Organization of the `hpc.conf` File

```
# Begin ShmemResource
# ...
# End ShmemResource

# Begin MPIOptions Queue=
# ...
# End MPIOptions

# Begin CREOptions Server=
# ...
# End CREOptions

# Begin PFSFileSystem=
# ...
# End PFSFileSystem

# Begin PFSServers
# ...
# End PFSServers

# Begin HPCNodes
# ...
# End HPCNodes

Begin PMODULES
...
End PMODULES

Begin PM=shm
...
End PM

Begin PM=rsm
...
End PM

Begin PM=tcp
...
End PM
```

Preparing to Edit hpc.conf

Perform the steps described below to stop the CRE daemons and copy the hpc.conf template.

Stop the CRE Daemons

Stop the CRE nodal and master daemons (in that order). The nodal daemons must be stopped on each node, including the master node.

Use the following scripts to stop the CRE nodal and master daemons:

```
# /etc/init.d/sunhpc.cre_node stop
# /etc/init.d/sunhpc.cre_master stop
```

You can use one of the Cluster Console Manager tools (cconsole, ctelnet, or crlogin) to broadcast the nodal stop command to all the nodes from a single command entered on the master node.

Note – If you edit the hpc.conf file at a later time and make any changes to the PFSServers section or PFSFileSystem section, you will need to also unmount any PFS file systems and stop the PFS daemons on the PFS I/O servers before making the changes.

Copy the hpc.conf Template

The Sun HPC ClusterTools distribution includes an hpc.conf template, which is stored, by default, in /opt/SUNWhpc/examples/rte/hpc.conf.template.

Copy the template from its installed location to /opt/SUNWhpc/etc/hpc.conf and edit it as described below in this section.

When you have finished editing hpc.conf, you need to update the CRE database with the new configuration information. This step is described in "Updating the CRE Database" on page 37.

Creating PFS I/O Servers

Decide which cluster nodes that you want to have function as PFS I/O servers. To be of value as PFS I/O servers, these nodes must be connected to one or more disk storage devices that have enough capacity to handle the PFS file systems you expect will be stored on them.

Note – The disk storage units should include some level of RAID support to protect the file systems against failure of individual storage devices.

Once you know which nodes you want as I/O servers, list their host names on separate lines in the `PFSServers` section of `hpc.conf`. TABLE 3-4 shows a sample `PFSServers` section that includes three PFS I/O server nodes.

TABLE 3-4 `PFSServers` Section Example

```
Begin PFSServers
NODE            OPTIONS
node4           nbufs=150
node5           nbufs=150
node6           nbufs=300
End PFSServers
```

The left column lists the host names of the PFS I/O server nodes.

The second column in this example shows an option specifying the amount of memory the PFS I/O daemon will have available for buffering transfer data. This value is specified in units of 32-Kbyte buffers. The optimal buffer size varies, depending on system type and load. Buffer sizes in the range of 128 to 512 provide reasonable performance on most Sun HPC clusters.

Note – You can use `pfsstat` to get reports on buffer cache hit rates. Knowing buffer cache hit rates can be useful for evaluating how well suited the buffer size is to the cluster's current I/O activity.

You can specify other options for PFS servers in addition to number of buffers, including the number of client threads to spawn and whether to control congestion on either the client side or the server side. For details, see Chapter 4.

Creating PFS File Systems

Add a separate `PFSFileSystem` section for each PFS file system you want to create. Include the following information in each `PFSFileSystem` section:

- The name of the parallel file system
- The host name of each server node in the parallel file system
- The name of the storage device attached to each server node
- The number of PFS I/O threads spawned to support each PFS storage device

- Other options, such as the number and size data buffers for each disk and the optimal alignment for disk accesses (see Chapter 4)

TABLE 3-5 shows sample `PFSFileSystem` sections for two parallel file systems, `pfs0` and `pfs1`.

TABLE 3-5 PFSFileSystem Section Example

```
Begin PFSFileSystem=pfs0
NODE            DEVICE              OPTIONS
node4           /dev/rdsk/c0t1d0s2  threads=1
node5           /dev/rdsk/c0t1d0s2  threads=1
End PFSFileSystem

Begin PFSFileSystem=pfs1
NODE            DEVICE              OPTIONS
node5           /dev/rdsk/c0t2d0s2  threads=1
node6           /dev/rdsk/c0t1d0s2  threads=1
End PFSFileSystem
```

Parallel File System Name

Specify the name of the PFS file system on the first line of the section, to the right of the = symbol.

Apply the same naming conventions to PFS files as are used for serial Solaris files.

Server Node Host names

The `NODE` column lists the hos tnames of the nodes that function as I/O servers for the parallel file system being defined. The example configuration in TABLE 3-5 shows two parallel file systems:

- `pfs0` – two server nodes: `node4` and `node5`
- `pfs1` – two server nodes: `node5` and `node6`

Note that I/O server `node5` is used by both `pfs0` and `pfs1`. This is possible because `node5` is attached to at least two storage devices, one of which is assigned to `pfs0` and the other to `pfs1`.

Storage Device Names

In the `DEVICE` column, specify the name of the device that will be used by the file system. Solaris device naming conventions apply.

Thread Limits

The `OPTIONS` column in this example specifies the number of threads a PFS I/O daemon will spawn for the disk storage device or devices it controls. The number of threads needed by a given PFS I/O server node depends primarily on the performance capabilities of its disk subsystem.

- For a storage object with a single disk or a small storage array, one thread may be enough to exploit the storage unit's maximum I/O potential.
- For a more powerful storage array, two or more threads may be needed to make full use of the available bandwidth.

Other PFS File System Options

You may also specify other file system options, including the number and size data buffers for each disk and the optimal alignment for disk accesses. These options are described in Chapter 4.

Specifying MPI Options

The `MPIOptions` section provides a set of options that control MPI communication behavior in ways that are likely to affect message-passing performance. It contains a template showing some general-purpose option settings, plus an example of alternative settings for maximizing performance. These examples are shown in TABLE 3-6.

- **G**eneral-purpose, multiuser settings – The template in the `MPIOptions` section is designed for general-purpose use at times when multiple message-passing jobs will be running concurrently.
- Performance settings – The second example is designed to maximize the performance of message-passing jobs when only one job is allowed to run at a time.

Note – The first line of the template contains the phrase "`Queue=hpc`." This is because the queue-based LSF workload management run-time environment uses the same `hpc.conf` file as does CRE. For LSF, the settings apply only to the specified queue. For CRE, the settings apply across the cluster.

The options in the general-purpose template are the same as the default settings for the Sun MPI library. In other words, you do not have to uncomment the general-purpose template to have its option values be in effect. This template is provided in the `MPIOptions` section so that you can see what options are most beneficial when operating in a multiuser mode.

TABLE 3-6 `MPIOptions` Section Example

```
# The following is an example of the options that affect the run
# time environment of the MPI library.  The listings below are
# identical to the default settings of the library. The "Queue=hpc"
# phrase makes this an LSF-specific entry, and only for the Queue
# named hpc. These options are a good choice for a multiuser queue.
# To be recognized by CRE, the "Queue=hpc" needs to be removed.
#
# Begin MPIOptions Queue=hpc
# coscheduling  avail
# pbind        avail
# spindtimeout   1000
# progressadjust   on
# spin  off
#
# shm_numpostbox           16
# shm_shortmsgsize        256
# rsm_maxsegsize      1048576
# rsm_numpostbox           15
# rsm_shortmsgsize        401
# rsm_maxstripe            2
# rsm_links          wrsm0,1
# maxprocs_limit   2147483647
# maxprocs_default       4096
#
# End MPIOptions

# The listing below is a good choice when trying to get maximum
# performance out of MPI jobs that are running in a Queue that
# allows only one job to run at a time.
#
# Begin MPIOptions Queue=performance
# coscheduling             off
# spin                      on
# End MPIOptions
```

If you want to use the `performance` settings, do the following:

- Delete the comment character (#) from the beginning of each line of the performance example, including the `Begin MPIOptions` and `End MPIOptions` lines.
- On CRE-based clusters, delete the "`Queue=performance`" phrase from the `Begin MPIOptions` line.

The resulting template should appear as follows:

```
Begin MPIOptions
coscheduling off
spin         on
End MPIOptions
```

The significance of these options is discussed in Chapter 7 "`hpc.conf` Configuration File".

Updating the CRE Database

When you have finished editing `hpc.conf`, update the CRE database with the new information. To do this, restart the CRE master and nodal daemons as follows:

```
# /etc/init.d/sunhpc.cre_master start
# /etc/init.d/sunhp.cre_node start
```

The nodal daemons must be restarted on all the nodes in the cluster, including the master node.

Authentication and Security

CRE provides basic security by means of a cluster password, which is stored in a key file on each node.

In addition, you can set up further methods of guarding the cluster against access by unauthorized users or programs. CRE supports UNIX system authentication (via `rhosts`), as well as two third-party mechanisms for authentication: Data Encryption Standard (DES) and Kerberos Version 5.

Authentication in the PFS file system is established separately from authentication in CRE. PFS supports third-party authentication only via Kerberos 5 (not DES). The setup procedure for PFS is described in Chapter 4.

Setting the CRE Cluster Password

CRE uses a root-read-only key file to control access to the cluster. The key file must exist on every node of the cluster, and the contents of all the key files must be identical. In addition, the key file must be placed on any node outside the cluster that might access the cluster (that is, on any node that may execute the command `mprun -c` *cluster_name*).

The key resides in `/etc/hpc_key.`*cluster_name* on each node.

The installation procedure creates a default key file on each node of the cluster. A strongly recommended step in the post-installation procedure is to customize the key immediately after installing the ClusterTools software. The key should be 10-20 alphanumeric characters.

The administrator can change the key at any time. As superuser, run the `set_key` script on each node in the cluster and on any nodes outside the cluster that may access the cluster:

```
# /opt/SUNWhpc/etc/set_key
```

It is preferable to stop the CRE daemons before changing the key, since a current MPI job might fail.

To guarantee that the cluster key is set identically on every node, you should use the Cluster Console Manager tools (described in Appendix A) to update all the key files at once.

Note – The cluster password security feature exists in addition to the "current" authentication method, as specified in the `hpc.conf` file and described below.

Establishing the Current Authentication Method

Authentication is established in the configuration file `hpc.conf` in the section `CREOptions`.

TABLE 3-7 `CREOptions` Section Example

```
Begin CREOptions
 ...
auth_opt       none
End CREOptions
```

The value of the `auth_opt` option is one of:

- `none` – UNIX system authentication (via `rhosts`), the default
- `des` – DES-based authentication
- `krb5` - Kerberos 5 authentication

To change authentication methods, stop all CRE daemons, edit the `hpc.conf` file, and then restart the CRE daemons. See "Preparing to Edit `hpc.conf`" on page 32.

Note – Since authentication methods limit the time that can elapse between the initiation of a remote procedure call (RPC) and the system's response, administrators should ensure that the nodes of the Sun HPC cluster and the machines from which users submit jobs are closely synchronized. For example, you can synchronize the machines by setting all system clocks to the same time using the Solaris `date` command.

Setting Up the Default Authentication

When authentication option `none` is in use, any CRE operation (such as `mpadmin` or `mprun`) attempted by the superuser will be allowed only if three items:

- The requesting host
- The master CRE host
- The hosts on which any `mprun` operation will execute

appear in one of the following files:

- The `/etc/sunhpc_rhosts` file, if it has been installed (the default)
- The default `.rhosts` file (if the `sunhpc_rhosts` file is not created at installation, or if it has been deleted)

The `sunhpc_rhosts` file's contents are visible only to the superuser.

To allow superuser access from hosts outside the cluster, the node name must be added to the `/etc/sunhpc_rhosts` file.

If the `/etc/sunhpc_rhosts` file is not used (or has been removed), the `.rhosts` file on each node must be updated to include the name of every node in the cluster. Using `.rhosts` assumes trusted hosts. For information on trusted hosts, see the man page for `hosts.equiv`.

Setting Up DES Authentication

In order to use DES authentication with CRE, host keys must exist for each host in the cluster and `/etc/.rootkey` must exist for each node of the cluster. User keys must exist on all hosts that will be used to communicate with the cluster using CRE commands, as well as on each node of the cluster (including the master), for each user who is to access the cluster. Inconsistent key distribution will prevent correct operation.

To set up DES authentication, you must ensure that all hosts in the system, and all users, have entries in both the `publickey` and `netname` databases. Furthermore, the entries in `/etc/nsswitch.conf` for both `publickey` and `netid` databases must point to the correct place. For further information, see the Solaris man pages for `publickey(4)`, `nsswitch.conf(4)`, and `netid(4)`.

After all new keys are in place, you need to restart the DES keyserver `keyserv`. You must also establish `/etc/.rootkey` on each node, as described in the man page `keylogin(1)`.

When the DES setup is complete, restart the CRE daemons (see "Starting the CRE Daemons" on page 11).

It is recommended that you use one of the Cluster Console Manager tools (`cconsole`, `ctelnet`, or `crlogin`) to issue identical commands to all the nodes at the same time. For information about the Cluster Console Manager, see Appendix A.

Note – While DES authentication is in use, users must issue the `keylogin` command before issuing any commands beginning with `mp`, such as `mprun` or `mpps`.

Setting Up Kerberos Authentication

To set up Kerberos 5 authentication, the administrator registers a host principal (`host`) and a Sun CRE (`sunhpc`) principal with an instance for each node that is to be used as a CRE client. In addition, each host must have `host` and `principal` entries in the appropriate `keytabs`.

For example: consider a system consisting of three nodes (`node0`, `node1`, and `node2`), in Kerberos realm `example.com`. Nodes `node0` and `node1` will be used as CRE servers and all three nodes will be used as CRE clients. The database should include the following principals as well as principals for any end-users of CRE services, created using the `addprinc` command in `kadmin`:

```
sunhpc/node0@example.com
sunhpc/node1@example.com
```

```
sunhpc/node2@example.com
host/node0@example.com
host/node1@example.com
host/node2@example.com
```

The `sunhpc` and `host` principals should have entries in the default `keytab` (created using the `ktadd` command in `kadmin`).

Any user who wishes to use CRE to execute programs must first obtain a ticket granting ticket via `kinit`.

For further information on Kerberos version 5, see the Kerberos documentation.

CHAPTER 4

PFS Configuration and Operations

As its name implies, the distinguishing characteristic of a parallel file system is the parallel layout of its files. Unlike traditional file systems, such as UFS, which perform file I/O on a single device, a PFS file system can span over multiple disks. Using multiple I/O streams, it can then access file data in parallel.

Standard Solaris file system commands can be used to access and manipulate PFS files. However, the high-performance I/O capabilities of PFS can be fully exploited through calls to MPI I/O library routines. In addition, PFS provides a number of commands which the administrator can use to create, mount, and monitor the status of PFS file systems.

This chapter covers the following topics:

- A basic description of PFS
- The PFS components
- PFS commands
- Creating a PFS
- Maintaining a PFS
- Further notes on configuring a PFS

PFS Basics

A PFS storage system consists of a PFS I/O server and the disk storage device(s) attached to it. A PFS I/O server is simply a Sun HPC node with these characteristics:

- It has disk storage systems attached.
- It has been defined as a PFS I/O server in the `hpc.conf` file.
- It is running a PFS I/O daemon.

A simple PFS storage system is shown in FIGURE 4-1. Here, a six-node cluster has three of its nodes configured as PFS servers, each with a single disk array as a storage device. PFS is parallel on both the server side (three servers in this case) and on the client side (up to six client/compute nodes in this case).

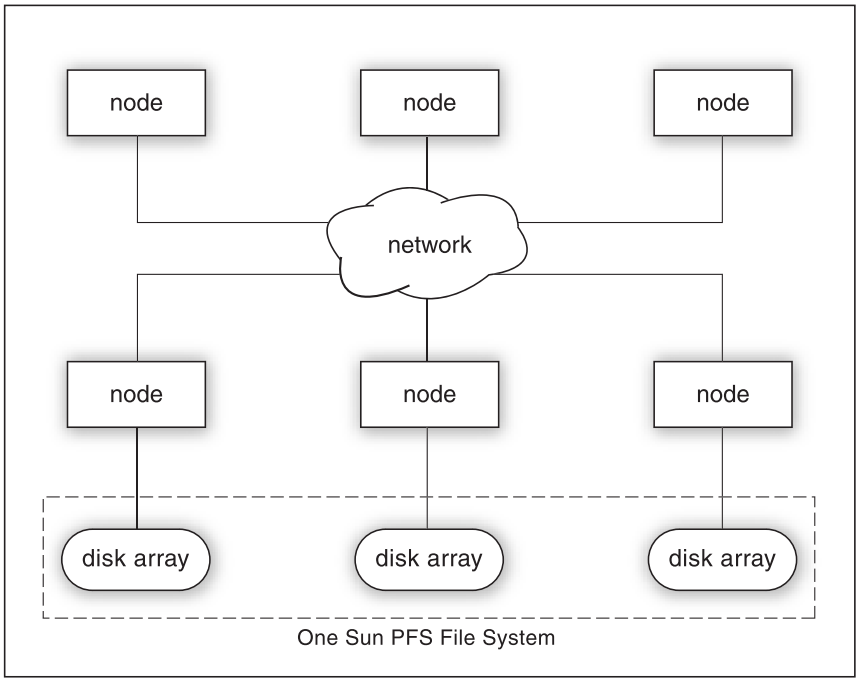

FIGURE 4-1 Sun HPC Cluster With a Single Sun PFS File System

A more complex configuration is shown below in an eight-node cluster:

- Four nodes function as compute servers only – CS0, CS1, CS2, and CS4.
- Three nodes function as PFS I/O servers only – IOS0, IOS 1, and IOS2.
- One node operates as both compute server and PFS I/O server – CS3-IOS3.

All four PFS I/O servers have disk storage subsystems attached. PFS I/O servers IOS0 and IOS3 each have a single disk storage unit, while IOS1 and IOS2 are each connected to two disk storage units.

The PFS configuration example in FIGURE 4-2 shows two PFS file systems, `pfs-demo0` and `pfs-demo1`. Each PFS file system is distributed across three PFS storage systems. This means an individual file in either file system will be divided into three subsets of blocks, which will be written to and read from its storage subsystems in three parallel data streams.

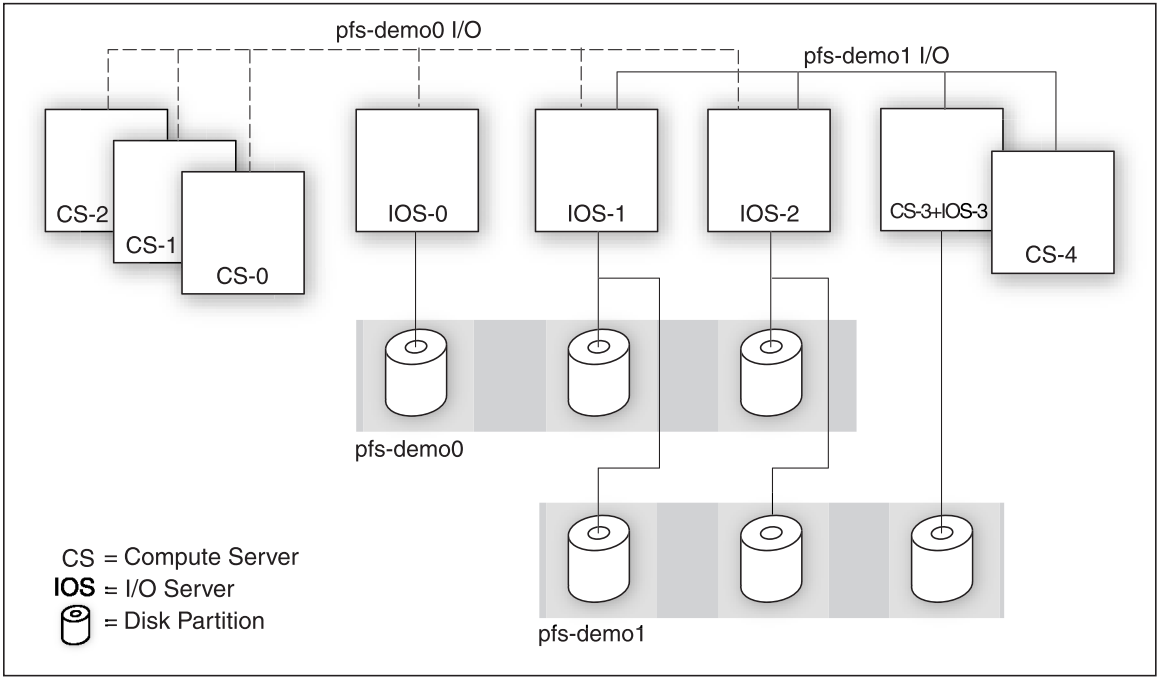

FIGURE 4-2 Sun HPC Cluster With Two Sun PFS File Systems

The dashed lines labeled "pfs-demo0 I/O" indicate paths between the compute processes on CS-0, CS-1, and CS-2 and the three I/O servers on the PFS file system `pfs-demo0`. Likewise, the solid lines labeled "pfs-demo1 I/O" represent I/O paths for the PFS file system `pfs-demo1`. Notice that I/O servers IOS-1 and IOS-2 are part of both PFS file systems. Also, one node is serving as both a compute server (CS-3) and as an I/O server (IOS-3).

This method of laying out PFS files introduces some file system configuration issues not encountered with UFS and other serial file systems. These issues are discussed in the balance of this section.

PFS Components

There are four main components in a Sun PFS file system:

- I/O daemon
- kernel module

- proxy daemon
- runtime library

The relationships among the PFS components are shown in FIGURE 4-3. The HPC ClusterTools user accesses PFS via a UNIX utility or an MPI I/O routine within an MPI job. The system administrator needs to make sure that the four components in the non-shaded boxes (the ones listed above) are set up properly. This section describes these four components.

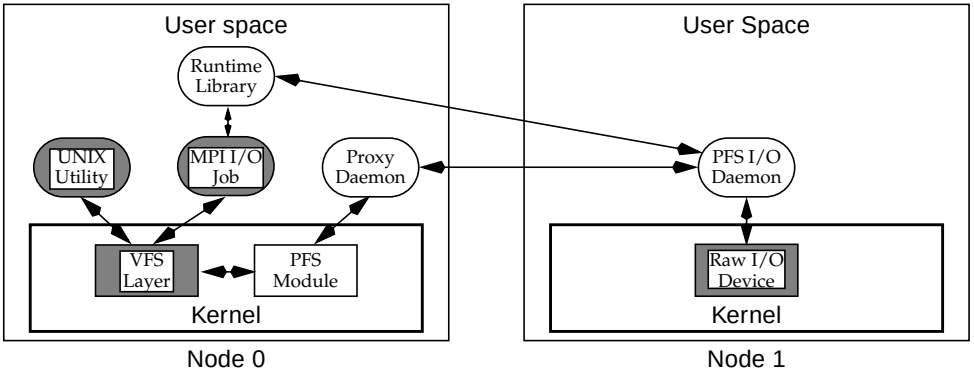

FIGURE 4-3 Relationships of PFS Components

I/O Daemon

A single multithreaded Sun PFS I/O daemon (IOD) runs on each node that is to be used as a Sun PFS server. The IOD is responsible for reading from and writing to disk on behalf of an application's multiple processes.

A PFS may be configured such that one or more threads are available to service client requests. By multiplexing the requests among the threads, the IOD may simultaneously service many more requests than it has threads. By default, an IOD launches five client threads. The administrator can control this in the `hpc.conf` file.

Each disk is assigned at least one dedicated thread. By default, every device is assigned a single thread, but the administrator may assign additional threads (in the `hpc.conf` file), if they are needed to achieve the full performance of a device.

Each I/O daemon maintains a pool of buffers, which are used as a staging area between the disk and the network. It is from these buffers that data is scattered or gathered. The number of buffers allocated by an IOD is determined by the system administrator in the `hpc.conf` file.

For complete information on the PFS I/O daemon, see its man page, `pfs.iod(1M)`.

Kernel Module

Despite the complexity of the underlying Sun PFS architecture, applications and users are able to interface with Sun PFS file systems as they would any other file system. Sun PFS supplies a Solaris kernel module that implements the Virtual File System (VFS) interface. The VFS interface makes the Sun PFS file system appear in the UNIX namespace.

When an application issues a file system-related system call, the VFS layer identifies the particular file system involved, and passes the request on to the kernel module responsible for that file system type.

When the Sun PFS kernel module is passed a file system request, occasionally the request can be satisfied with information available locally. Given the distributed nature of Sun PFS, it is more likely that one or more I/O daemons will need to be contacted to service any given request. In that case, the kernel module marshals the arguments of the request, and passes it on to the Sun PFS proxy daemon.

After installation and activation, the 32-bit PFS kernel module resides in `/usr/kernel/fs`, and the 64-bit PFS kernel module resides in `/usr/kernel/fs/sparcv9/`.

To check whether the module is loaded, use:

```
# modinfo | grep pfs
```

If the kernel module is loaded, you see output similar to this:

```
86 10291d64   b0bc  18   1  pfs (pfs filesystem)
```

Proxy Daemon

There is a single proxy daemon running on each node that intends to access Sun PFS file systems. The proxy daemon receives requests from the PFS kernel module. This daemon then determines which IODs are required to satisfy the request, works with the IODs to satisfy the request, and passes the result back to the kernel module. The proxy daemon is multithreaded, and the number of threads may be tuned by the administrator.

For complete information on the PFS proxy daemon, see its man page, `pfs.proxy(1M)`.

Runtime Library

Users do not interact directly with the Sun PFS runtime library. Instead, applications use the MPI I/O features in the Sun MPI library to access PFS files. Sun MPI I/O works with either Sun PFS or any serial file system in the Solaris Operating Environment. At runtime, the Sun MPI I/O library determines which type of file system is being used. For serial file systems, Sun MPI I/O uses standard Solaris system calls to read and write data. For parallel file systems, Sun MPI I/O uses the Sun PFS runtime library to derive better data-access performance. This behavior allows a user to run the same application on both UFS and Sun PFS without any modifications.

PFS File System Commands

The commands available to the PFS administrator include:

- Standard Solaris file system commands `mkfs`, `mount/umount`, and `fsck`
- PFS-specific commands `pfsmount`, `pfsumount`, `pfsstart`, `pfsstop`, and `pfsstat`

Solaris File System Commands

The Solaris file system commands `mkfs`, `mount/umount`, and `fsck` behave with the PFS file system just as they do with other Solaris file systems. There are two ways to apply these commands to a PFS file system:

- Supply the command-line option `-F pfs`.
- Add an entry for each PFS file system in the file `/etc/vfstab` (as shown in TABLE 4-5 on page 53).

For your convenience, Sun PFS provides man pages for the Solaris commands under the names `fsck_pfs(1)`, `mkfs_pfs(1)`, and `mount_pfs(1)`.

PFS-Specific File System Commands

Sun PFS provides special variants of several file system commands that operate on all nodes of a cluster when invoked on any one node. These commands all reside in `/opt/SUNWhpc/bin`. They are:

- `pfsmount/pfsumount`

- `pfsstart/pfsstop`
- `pfsstat`

These commands act on multiple nodes at once. See their respective man pages for detailed information.

Creating a Parallel File System

This section details the steps you perform to create a PFS file system and make it available for use. These steps are:

- Configure the file system and its servers in the `hpc.conf` file.
- Start the PFS I/O daemons on the servers and the PFS proxy daemons on the clients.
- Make and mount the file system.
- Verify that the file system is mounted.

Configure PFS File Systems and Servers

The administrator configures the PFS in the Sun HPC ClusterTools configuration file `hpc.conf`. This file has two sections that pertain to PFS: the `PFSFileSystem` section and the `PFSServers` section.

If your cluster is configured to use the Sun Cluster Runtime Environment (CRE), the `hpc.conf` file resides at:

```
/opt/SUNWhpc/etc/hpc.conf
```

The `hpc.conf` file is described in more detail in Chapter 7 of this manual and in the man page `hpc.conf(4)`.

PFS File Systems

In the `hpc.conf` section named `PFSFileSystem`, you supply a name of a PFS file system (in the example below, `pfs-demo0`) and list the host name of each of its server nodes and the name of the associated storage device. For each disk, you may also specify a comma-separated list of options. In this example, raw disk partitions are used as the storage devices.

TABLE 4-1 Sample `PFSFileSystem` Entry in hpc.conf File

```
Begin PFSFileSystem=pfs-demo0
# NODE    DEVICE              OPTIONS
hpc-io0 /dev/rdsk/c0t1d0s2  threads=1
hpc-io1 /dev/rdsk/c0t1d0s2  threads=1,bufcnt=8,bufsize=1m
End PFSFileSystem
```

The `PFSFileSystem` options are shown in TABLE 4-2.

TABLE 4-2 `PFSFileSystem` Setup Options

Option	Description
`threads=`	The number of client threads to be spawned in the PFS I/O daemon to manage that storage device. Default is 1.
`bufcnt=`	The number of data buffers to be allocated for servicing requests to that storage device. Default is 4.
`bufsize=`	The size of the data buffers for that storage device. The value is rounded down, if necessary, to the nearest multiple of 32Kb (PFS's basic block size). Default is 1Mb.
`align=`	The alignment for disk accesses (designed for RAIDs). The value is rounded down, if necessary, to the nearest multiple of 32Kb (PFS's basic block size). Default is 32Kb.

PFS Servers

Each node listed in the `PFSFileSystem` section of the configuration file (shown above) must also be listed in the next section, `PFSServers`, along with any desired options. For example:

TABLE 4-3 Sample `PFSServers` Entry in `hpc.conf` File

```
Begin PFSServers
# NODE         OPTIONS
hpc-io0      nbufs=32
hpc-io1      nbufs=32
End PFSServers
```

The `PFSServers` options are shown in TABLE 4-4.

TABLE 4-4 `PFSServers` Setup Options

Option	Description
`threads=`	The number of threads to be spawned to handle requests from compute processes. Default is 5.
`nbufs=`	The amount of memory the PFS I/O daemon will have for buffering data, specified in units of 32-KB buffers. Default is 64.
`auth={krb5 | none}`	The authorization method to be used by PFS. Options are none (the default) or Kerberos Version 5.
`clntcong={yes | no}`	Turn on (or turn off) congestion control on the client (compute server) side. This option and the next cause PFS to regulate carefully the load it places on the network. They are intended to be used with networks (or network interface cards) that do not degrade gracefully under heavy load. In most cases, there is a performance penalty for setting this option to yes. Default is no.
`servcong={yes | no}`	Turn on (or turn off) congestion control on the server side. This option and the previous cause PFS to regulate carefully the load it places on the network and on particular interfaces. They are intended to be used with networks (or network interface cards) that do not degrade gracefully under heavy load. In most cases, there is a performance penalty for setting this option to yes. Default is no.

More information about PFS authorization is provided in the section "Authentication With Kerberos Version 5" on page 56.

Start the PFS Daemons

A PFS I/O daemon must be running on each PFS I/O server, and a PFS proxy daemon must be running on each client/compute node that will access PFS file systems. These daemons start automatically when you boot the node(s) on which they reside.

This section describes the procedures for starting these daemons manually. To stop the daemons, see the section “Stopping the PFS File System” on page 54.

Starting an I/O Daemon

To start an I/O daemon on a newly created I/O server without rebooting, use the following command line (as root) to start the daemon manually on that node.

```
# /etc/init.d/sunhpc.pfs_server start
```

Starting a Proxy Daemon

To start PFS proxy daemons manually, use the following command line on each node that requires it.

```
# /etc/init.d/sunhpc.pfs_client start
```

Starting the I/O and Proxy Daemons All at Once

As an alternative to starting the PFS I/O daemons and proxy daemons individually, as shown above, you can start both the PFS client and server daemons on every node in the cluster with one command. Execute this command on any one node in the cluster:

```
# pfsstart
```

Create and Mount PFS File Systems

As with UFS file systems, you can use the Solaris utilities `mkfs` and `mount` to create and mount PFS file systems.

You may want to add an entry for each PFS file system in the file `/etc/vfstab`. This makes it unnecessary to include the `-F` option when making and mounting the file systems to specify the file system’s type.

TABLE 4-5 shows how a PFS file system entry should look in the `/etc/vfstab` file.

TABLE 4-5 Sample PFS Entry in /etc/vfstab File

#device to mount	device to fsck	mount point	FS type	fsck pass	mount at boot	mount options
pfs-demo0	-	/pfs_demo0	pfs	-	no	-

Create the File System

For example, the following creates a 64-Mbyte PFS file system named `pfs-demo0`. Execute it on any client or server node.

```
# mkfs -F pfs pfs-demo0 64M
```

The option `-F pfs` specifies the file system's type. It may be omitted if you have added an entry for file system `pfs-demo0` in the file `/etc/vfstab`.

Mount the File System

Next, mount the file system on each client node that has a PFS proxy daemon. The file system must exist (from the action of the `mkfs` command) before it can be mounted.

```
hpc-client0# mount –F pfs pfs-demo0 /pfs_demo0
hpc-client1# mount –F pfs pfs-demo0 /pfs_demo0
```

Alternatively, you can execute the following on a single node. This causes the PFS file system to be mounted on all nodes in the cluster.

```
# /opt/SUNWhpc/bin/pfsmount pfs-demo0
```

Verify That the PFS File System Is Mounted

Before anyone attempts to use a newly created PFS file system, it is a good idea to verify that it is correctly mounted. This can easily be done (as superuser) by invoking `df –F pfs` on any node that has a PFS proxy daemon. The example below shows `df` output with the PFS file system `pfs-demo0` included.

```
# df –F pfs
/pfs-demo0        (pfs-demo0    ):66589632 blocks    65426 files
```

Alternatively, if you execute the `pfsmount` command without any arguments, it will list every mounted PFS file system in the cluster. For eaxmple:

```
# /opt/SUNWhpc/bin/pfsmount
Mounted PFS filesystems:
   hpc-node4:
       pfs-demo0

   hpc-node5:
       pfs-demo0
```

The PFS file system `pfs-demo0` is now ready to use. You can use Solaris utilities to create and delete PFS files and directories in the directory `/pfs-demo0`, just as you would in any UFS file system. To achieve best performance, however, applications should access PFS data via MPI I/O read and write calls.

Maintaining a PFS File System

In general, the administrator needs to perform maintenance actions only in the event of a machine crash or when changing the configuration of the cluster.

Before bringing down any machine that is used as a Sun PFS server, any Sun PFS file systems assigned to that machine must be unmounted on all client nodes.

Stopping the PFS File System

Stopping PFS is a two-step process: unmount the file system and then stop the daemons.

PFS file systems are automatically unmounted and PFS I/O daemons are automatically stopped when nodes are shut down or rebooted. This section shows how to perform these steps manually.

Unmount the File System

You can unmount a PFS file system from the nodes individually or all at once.

To unmount the PFS file system `pfs-demo0` from all nodes of a cluster, use the command `pfsumount` on any node:

```
# /opt/SUNWhpc/bin/pfsumount pfs-demo0
```

Alternatively, use the command `umount` on each node:

```
# umount pfs-demo0
```

Stop the I/O Daemons

Once you've unmounted the PFS file system, stop the I/O daemons on each PFS server. You do this by running the following command on each server node:

```
# /etc/init.d/sunhpc.pfs_server stop
```

Stop the PFS Proxy Daemons

To manually stop a PFS proxy daemon on an individual node, use the following command:

```
# /etc/init.d/sunhpc.pfs_client stop
```

Stop the PFS I/O and Proxy Daemons All at Once

As an alternative to stopping the PFS I/O daemons and proxy daemons separately, as shown above, you can stop both the PFS client and server daemons on every node in the cluster with one command. Execute this command on any one node in the cluster:

```
# /opt/SUNWhpc/bin/pfsstop
```

Recovering from Node or Daemon Failure

If a Sun PFS server crashes or is shut down without unmounting the PFS file systems it serves, those file systems will be marked as "dirty." This indicates that the server may have had data stored in its buffer cache, and is unable to guarantee that all data written to that file system was committed to stable storage before the failure.

In this event, Sun PFS will refuse to mount those file systems until the administrator has run `fsck`. This utility examines a file system's on-disk data structures to help ensure that the file system itself is in a stable and consistent state. If there are inconsistencies, `fsck` will attempt to resolve them to enable the file system to be used without fear of accidentally damaging existing data.

Notes on PFS Configuration

This section provides further information on configuring PFS file systems with Kerberos 5 authentication and on deciding whether to run applications and I/O daemons on the same PFS I/O servers or to segregate them onto separate nodes.

Authentication With Kerberos Version 5

PFS supports Kerberos Version 5 for authentication of users and programs.

Authentication is specified as an option to the PFS daemons `pfs.iod` and `pfs.proxy`. For the I/O daemon, you specify the authentication method (Kerberos 5 or none) in the `PFSServers` section of the file `hpc.conf`. For the proxy daemons, you specify the authentication method as a command-line argument. For complete information, see the man pages for `pfs.iod(1M)`, `pfs.proxy(1M)`, and `hpc.conf(4)`.

To ensure that authentication is enabled following a system reboot, you must modify the PFS startup script, `/etc/init.d/sunhpc.pfs_server`.

To set up Kerberos version 5 authentication, `root` must have an entry in the Kerberos database. The administrator must register a principal name for Sun PFS (`sunhpc-pfs`) with an instance for each node that is to be used as a PFS client. Appropriate `keytab` files must be stored on each host that is to be used as a PFS server.

For example: consider a system consisting of three nodes (`node0`, `node1`, and `node2`), in Kerberos realm `example.com`. Nodes `node0` and `node1` will be used as PFS servers and all three nodes will be used as PFS clients. The database should have three principals (one for each client):

```
sunhpc-pfs/node0@example.com
sunhpc-pfs/node1@example.com
sunhpc-pfs/node2@example.com
```

The `keytab` file for `node0` must have an entry for `sunhpc-pfs/node0@example.com`, and the `keytab` file for `node1` must have an entry for `sunhpc-pfs/node1@example.com`.

Before starting the PFS I/O daemons on a node, you must obtain a ticket-granting ticket (`tgt`) on that node. The `tgt`'s for the PFS daemons may expire, so they have to be refreshed periodically.

For further information on Kerberos Version 5, see your Kerberos documentation.

Setting System Parameters on TCP Clusters

Very large clusters connected with TCP interconnects may require adjustments to certain system parameters if they run PFS jobs with large numbers of processes. With the default TCP parameter settings, it is possible that the TCP connections might time out before the PFS I/O daemon can establish all the needed connections or service the processes' data requests.

In particular, the following changes to TCP parameters have been observed to increase the amount of time available and thus make job timeout less likely:

```
tcp_ip_abort_cinterval  - increase to 2400000
tcp_time_wait_interval  - lower to 60000
tcp_rexmit_interval_max - increase to 120000
tcp_ip_abort_interval   - increase to 1200000
```

The following parameters have not been observed to affect timeout on TCP clusters, but they might have an impact in certain cluster configurations and system setups:

```
tcp_conn_req_max_q      - increase to 256
tcp_conn_req_max_q0     - increase to 2048
```

These values are general guidelines. The actual values needed to eliminate all timeouts and yield the best PFS performance will vary among configurations.

Applications and I/O Processes, Collocate or Run Separately?

If you plan to configure only a subset of the nodes on a cluster as PFS I/O servers, you will have the option of either collocating applications and I/O daemons on the same PFS I/O servers or segregating them onto separate nodes. If, however, you configure all the nodes in a cluster as PFS I/O servers, you will of necessity collocate applications and PFS I/O daemons.

Guidelines for making this choice are provided below.

Conditions That Favor Collocating

Each of the following conditions favors collocating applications with PFS I/O daemons.

- Large nodes (many CPUs per node)
- Fast disk-storage devices (storage arrays, for example) on each node
- Lower-performance cluster interconnect, such as 10- or 100-BASE-T Ethernet
- Small number of applications competing for node resources

When these conditions exist in combination, the network is more likely to be a performance-limiting resource than the relatively more powerful nodes. Therefore, it becomes advantageous to locate applications on the PFS I/O servers to decrease the amount of data that must be sent across the network.

Conditions That Favor Separating Applications and IODs

You should avoid running applications on I/O server nodes when some or all of the following conditions exist.

- Smaller nodes
- Slow disk storage devices (single disks, for example) on each node
- Relatively high-performance cluster interconnect
- Large number of applications competing for node resources

In this case, the competition for memory, bus bandwidth, and CPU cycles may offset any performance advantages local storage would provide.

Effect of Cluster Size

By itself, the size of a cluster (number of nodes) does not favor either collocating or not collocating applications and PFS I/O daemons. Larger clusters do, however, attenuate the benefits of collocating. This is because the amount by which collocating reduces network traffic can be expressed roughly as

$$T_c = T_s - T_s/N$$

where T_c is the level of network traffic using collocating, T_s is the level of network traffic without collocating, and N is the number of nodes in the cluster. In other words, collocating reduces network traffic by 1/number-of-nodes. The more nodes there are in the cluster, the smaller the effect of collocating.

CHAPTER 5

Cluster Configuration Notes

This chapter examines various issues that may have some bearing on choices you make when configuring your Sun HPC cluster. The discussion is organized into three general topic areas:

- Nodes
- Interconnects
- Parallel file systems

Nodes

Configuring a Sun SMP or cluster of SMPs to use Sun HPC ClusterTools software involves many of the same choices seen when configuring general-purpose compute servers. Common issues include the number of CPUs per machine, the amount of installed memory, and the amount of disk space reserved for swapping.

Because the characteristics of the particular applications to be run on any given Sun HPC cluster have such a large effect on the optimal settings of these parameters, the following discussion is necessarily general in nature.

Number of CPUs

Since Sun MPI programs can run efficiently on a single SMP, it can be advantageous to have at least as many CPUs as there are processes used by the applications running on the cluster. This is not a necessary condition since Sun MPI applications can run across multiple nodes in a cluster, but for applications with very large interprocess communication requirements, running on a single SMP may result in significant performance gains.

Memory

Generally, the amount of installed memory should be proportional to the number of CPUs in the cluster, although the exact amount depends significantly on the particulars of the target application mix.

Computationally intensive Sun HPC applications that process data with some amount of locality of access often benefit from larger external caches on their processor modules. Large cache capacity allows data to be kept closer to the processor for longer periods of time.

Swap Space

Because Sun HPC applications are, on average, larger than those typically run on compute servers, the swap space allocated to Sun HPC clusters should be correspondingly larger. The amount of swap should be proportional to the number of CPUs and to the amount of installed memory. Additional swap should be configured to act as backing store for the shared memory communication areas used by Sun HPC ClusterTools software in these situations.

Sun MPI jobs require large amounts of swap space for shared memory files. The sizes of shared memory files scale in stepwise fashion, rather than linearly. For example, a two-process job (with both processes running within the same SMP) requires shared memory files of approximately 35 Mbytes. A 16-process job (all processes running within the same SMP) requires shared memory files of approximately 85 Mbytes. A 256-process job (all processes running within the same SMP) requires shared memory files of approximately 210 Mbytes.

Interconnects

One of the most fundamental issues to be addressed when configuring a cluster is the question of how to *connect* the nodes of the cluster. In particular, both the type and the number of networks should be chosen to complement the way in which the cluster is most likely to be used.

Note – For the purposes of this discussion, the term *default network* refers to the network associated with the standard host name. The term *parallel application network* refers to an optional second network, operating under the control of CRE.

In a broad sense, a Sun HPC cluster can be viewed as a standard LAN. Operations performed on nodes of the cluster will generate the same type of network traffic that is seen on a LAN. For example, running an executable and accessing directories and files will cause NFS traffic, while remote login sessions will cause network traffic. This kind of network traffic is referred to here as *administrative* traffic.

Administrative traffic has the potential to tax cluster resources. This can result in significant performance losses for some Sun MPI applications, unless these resources are somehow protected from this traffic. Fortunately, CRE provides enough configuration flexibility to allow you to avoid many of these problems.

The following sections discuss some of the factors that you should consider when building a cluster for Sun HPC applications.

ClusterTools Internode Communication

Several Sun HPC ClusterTools components generate internode communication. It is important to understand the nature of this communication in order to make informed decisions about network configurations.

Administrative Traffic

As mentioned earlier, a Sun HPC cluster generates the same kind of network traffic as any UNIX-based LAN. Common operations like starting a program can have a significant network impact. The impact of such administrative traffic should be considered when making network configuration decisions.

When a simple serial program is run within a LAN, network traffic typically occurs as the executable is read from a NFS-mounted disk and paged into a single node's memory. In contrast, when a 16- or 32-process parallel program is invoked, the NFS server is likely to experience approximately simultaneous demands from multiple nodes—each pulling pages of the executable to its own memory. Such requests can often result in large amounts of network traffic. How much traffic occurs will depend on various factors, such as the number of processes in the parallel job, the size of the executable, and so forth.

CRE-Generated Traffic

CRE uses the cluster's default network interconnect to perform communication between the daemons that perform resource management functions. CRE makes heavy use of this network when Sun MPI jobs are started, with the load being roughly proportional to the number of processes in the parallel jobs. This load is in

addition to the start-up load described in the previous section. CRE will generate a similar load during job termination as the CRE database is updated to reflect the expired MPI job.

There is also a small amount of steady traffic generated on this network as CRE continually updates its view of the resources on each cluster node and monitors the status of its components to guard against failures.

Sun MPI Interprocess Traffic

Parallel programs use Sun MPI to move data between processes as the program runs. If the running program is spread across multiple cluster nodes, then the program generates network traffic.

Sun MPI uses the network that CRE instructs it to use, which can be set by the system administrator. In general, CRE instructs Sun MPI to use the fastest network available so that message-passing programs obtain the best possible performance.

If the cluster has only one network, then message-passing traffic shares bandwidth with administrative and CRE functions. This results in performance degradation for all types of traffic, especially if one of the applications is performing significant amounts of data transfer, as message-passing applications often do. You should understand the communication requirements associated with the types of applications to be run on the Sun HPC cluster in order to decide whether the amount and frequency of application-generated traffic warrants the use of a second, dedicated network for parallel application network traffic. In general, a second network will significantly assist overall performance.

Prism Traffic

The Prism graphical programming environment is used to tune, debug, profile, and visualize data from Sun MPI programs running within the cluster. As the Prism program itself is a parallel program, starting it generates the same sort of CRE traffic that invocation of other applications generates.

Once the Prism environment has been started, two kinds of network traffic are generated during a debugging session. The first, which has been covered in preceding sections, is traffic created by running the Sun MPI code that is being debugged. The second kind of traffic is generated by the Prism program itself and is routed over the default network along with all other administrative traffic. In general, the amount of traffic generated by the Prism program itself is small, although viewing performance analysis data on large programs and visualizing large data arrays can cause transiently heavy use of the default network.

Parallel I/O Traffic

Sun MPI programs can make use of the parallel I/O capabilities of Sun HPC ClusterTools, but not all such programs do so. You need to understand how distributed multiprocess applications that are run on the Sun HPC cluster will make use of parallel I/O to understand the ramifications for network load.

Applications can use parallel I/O in two different ways, and the choice is made by the application developer. Applications that use parallel I/O to read from and write to standard UNIX file systems can generate NFS traffic on the default network, on the network being used by the Sun MPI component, or some combination of the two. The type of traffic that is generated depends on the type of I/O operations being used by the applications. Collective I/O operations generate traffic on the Sun MPI network, while most other types of I/O operations involve only the default network.

Applications that use parallel I/O to read from and write to PFS file systems use the network specified by CRE. In a one-network cluster, this means that parallel I/O traffic is routed over the same network used by all other internode traffic. In a two-network cluster, where an additional network has been established for use by parallel applications, you would normally configure CRE so that this type of parallel I/O would be routed over the parallel application network.

Network Characteristics

Bandwidth, latency, and performance under load are all important network characteristics to consider when choosing interconnects for a Sun HPC cluster. These are discussed in this section.

Bandwidth

Bandwidth should be matched to expected load as closely as possible. If the intended message-passing applications have only modest communication requirements and no significant parallel I/O requirements, then a fast, expensive interconnect may be unnecessary. On the other hand, many parallel applications benefit from *large pipes* (high-bandwidth interconnects). Clusters that are likely to handle such applications should use interconnects with sufficient bandwidth to avoid communication bottlenecks. Significant use of parallel I/O would also increase the importance of having a high-bandwidth interconnect.

It is also a good practice to use a high-bandwidth network to connect large nodes (nodes with many CPUs) so that communication capabilities are in balance with computational capabilities.

An example of a low-bandwidth interconnect is the 10-Mbit/s Ethernet. Examples of higher-bandwidth interconnects include ATM and switched FastEthernet.

Latency

The *latency* of the network is the sum of all delays a message encounters from its point of departure to its point of arrival. The significance of a network's latency varies according to the communication patterns of the application.

Low latency can be particularly important when the message traffic consists mostly of small messages—in such cases, latency accounts for a large proportion of the total time spent transmitting messages. Transmitting larger messages can be more efficient on a network with higher latencies.

Parallel I/O operations are less vulnerable to latency delays than small-message traffic because the messages transferred by parallel I/O operations tend to be large (often 32 Kbytes or larger).

Performance Under Load

Generally speaking, better performance is provided by switched network interconnects, such as ATM and Fibre Channel. Network interconnects with collision-based semantics should be avoided in situations where performance under load is important. Unswitched 10-Mbit/s and 100-Mbit/s Ethernet are the two most common examples of this type of network. While 10-Mbit/s Ethernet is almost certainly not adequate for any HPC application, a switched version of 100-Mbit/s Ethernet may be sufficient for some applications.

Notes on RSM Setup

The `hpc_rsmd` daemon, which is thread-safe , receives requests from client processes by means of door calls.While running, the `hpc_rsmd` creates and destroys RSM backing store using System V shared memory. The directory `/var/hpc/rsm` is reserved specifically for `hpc_rsmd` usage and should not have any user files stored in it. This directory will contain the `hpc_rsmd_door` file, which is used by RSM protocol modules (PMs) to contact the daemon.

The daemon `hpc_rsmd` has a polling thread that it uses to determine the state of known RSM paths and to determine if unallocated segments should be reaped. The periodicity of the path-monitor thread is 60 seconds.

The RSM PM is activated only by a call into the MPI library made by an application. It has an MT-safe version to be used by the `libmpi_mt.so` library. Its memory usage is dynamic, dependent on the number of processes it has set up to communicate with and the memory it has exported for use in the communications.

When Sun HPC ClusterTools software is installed, certain system parameters in `/etc/system` are set to enable RSM communication. You might wish to change these values to suit other requirements of your site. The default settings established at installation time are the following:

```
set shmsys:shminfo_shmseg=0x800
set shmsys:shminfo_shmmni=0x1000
set shmsys:shminfo_shmmax=0xf00000
```

If you decrease any of these values, then the system might not be able to run MPI jobs using RSM. Also, you may need to increase these values beyond the defaults if other applications that use System V shared memory will be running in tandem with MPI jobs.

Storage and the Parallel File System

The performance of distributed multiprocess applications can be enhanced by using PFS file systems. How much value PFS contributes depends on how storage and I/O are configured on your Sun HPC cluster.

PFS on SMPs and Clusters

Although a PFS file system can be used in a single SMP, PFS is more beneficial to a cluster of SMPs. A high-performance serial file system is likely to provide better I/O performance on a single SMP.

Note – Applications written to use MPI I/O for file I/O can easily be moved from single SMPs with high-speed local file systems to cluster environments with PFS file systems.

PFS Using Individual Disks or Storage Arrays

Since PFS distributes file data and file system metadata across all the storage devices in a file system, the failure of any single device will result in the loss of access to the file system. For that reason, the underlying storage devices in the PFS should be storage arrays with some form of RAID support.

Although PFS may be configured to manage each disk in a storage array individually, for the purposes of safety and performance some form of volume manager should be used to manage the individual disks. PFS should then be used to manage the volumes across multiple servers.

PFS and Storage Placement

In broad terms, you can choose between two models for locating I/O servers in a Sun HPC cluster:

- Use separate nodes for program execution and I/O support.
- Use the same nodes for both program execution and I/O.

Traditionally, administrators have assigned a subset of nodes in a cluster to the role of I/O server and have reserved the remainder for computational work. Often this strategy was based on the assumption that individual nodes were relatively underpowered. Given the computational power and I/O bandwidth of Sun's SMP nodes, this assumption is less likely to be true—consequently, the benefits of segregating I/O and computation are less compelling than was once the case.

In theory, colocating computation and I/O support on the same nodes can improve I/O performance by reducing the amount of I/O traffic going over the network. In reality, the performance gains provided by an increase in local I/O may be small. When N nodes in a cluster are configured as PFS I/O servers, N-1/N of the I/O traffic will go off-node. When N=2, half the I/O traffic will be on-node and half off. This is the best efficiency that can be expected when mixing computation and I/O on the same servers. For larger numbers of I/O servers, the percentage of I/O traffic that will go off-node increases asymptotically toward 100%.

Separate Functions

If nodes act as either compute servers or as I/O servers, but not as both, all parallel I/O operations will generate network traffic, and the node's network interface will determine the limit of the performance of a parallel file system. In such cases, the total number of processing nodes being used to run the processes of a parallel job will set an upper limit on the aggregate throughput available. The absolute limit will be set by the bandwidth limitations of the network interconnect itself.

For example, if a 16-process job is scheduled on four SMP nodes, then the limiting factor will be the four network adapters that the SMPs will use for communicating with the remote storage objects of the parallel file system.

In such cases, the best rule of thumb is to match (as closely as possible) the number of compute nodes to the number of I/O nodes so that consumer bandwidth is roughly matched to producer bandwidth within the limitations of the cluster's network bandwidth.

Mixed Functions

When nodes act as both compute servers and PFS I/O servers, the same network bandwidth considerations discussed above apply. However, some performance gains may be realized by having a portion of the I/O operations access local disks. The likely limits of such gains are also discussed in "PFS and Storage Placement" on page 68.

In order to maximize efficiency in the mixed-use mode, applications should be examined to determine the most efficient mapping of their processes onto cluster nodes. Then, the PFS file system should be set up to complement this placement with storage objects being installed on those nodes.

For example, a 16-process application may run best on a given cluster when four processes are scheduled onto each of four-CPU SMPs. In this case, the parallel file system should be configured with storage objects on each of the four SMPs.

Balancing Bandwidth for PFS Performance

When deciding where to place storage devices, it is important to balance the bandwidth of the storage device with the bandwidth of the network interface. For example, in a cluster running on switched Fast Ethernet, the bandwidth out of any node is limited to 100 Mbits/s.

CHAPTER 6

mpadmin: Detailed Description

This chapter describes the CRE cluster administration command interface, `mpadmin`. Topics covered include:

- `mpadmin` syntax, the subcommands it supports, and other aspects of `mpadmin` functionality
- How to use `mpadmin` to perform various cluster administration tasks

mpadmin Syntax

The `mpadmin` command has six optional arguments, as follows:

```
# mpadmin [-c command] [-f filename] [-h] [-q] [-s cluster_name] [-V]
```

When you invoke `mpadmin` with the `-q` or `-s` option or no option, `mpadmin` goes into the interactive mode, displaying the `mpadmin` prompt. In this mode, you can execute any number of `mpadmin` subcommands until you quit the interactive session.

Note – In the rest of this discussion, `mpadmin` subcommands are referred to as `mpadmin` commands or simply as commands.

When you invoke `mpadmin` with the `-c`, `-f`, `-h`, or `-V` option, `mpadmin` performs the requested operation and then returns to the shell level. For command arguments, you can specify most of the subcommands that are available within the `mpadmin` interactive environment.

Command-Line Options

TABLE 6-1 provides summary definitions of the `mpadmin` command-line options. This section describes their use.

TABLE 6-1 `mpadmin` Options

Option	Description
`-c` *command*	Execute single specified command
`-f` *file-name*	Take input from specified file
`-h`	Display help/usage text
`-q`	Suppress the display of a warning message when a non-superuser attempts to use restricted command mode
`-s` *cluster-name*	Connect to the specified Sun HPC cluster
`-V`	Display `mpadmin` version information

`-c` *command* – Single Command Option

Use the `-c` option when you want to execute a single `mpadmin` command and return automatically to the shell prompt. For example, the following use of `mpadmin -c` changes the location of the CRE log file to `/home/wmitty/cre_messages`:

```
# mpadmin -c set logfile="/home/wmitty/cre_messages"
#
```

Note – Most commands that are available via the interactive interface can be invoked via the `-c` option. See "`mpadmin` Command Overview" on page 73 for an overview of the `mpadmin` command set and a list of which commands can be used as arguments to the `-c` option.

`-f` *file-name* – Take Input From a File

Use the `-f` option to supply input to `mpadmin` from the file specified by the *file-name* argument.

-h – Display Help

The -h option displays help information about mpadmin.

-q – Suppress Warning Message

Use the -q option to suppress a warning message when a non-root user attempts to invoke a restricted command.

-s *cluster-name* – Connect to Specified Cluster

Use the -s option to connect to the cluster specified by the *cluster-name* argument.

-V – Version Display Option

Use the -V option to display the version of mpadmin.

mpadmin Objects, Attributes, and Contexts

Before examining the set of mpadmin commands further, it is useful to understand three concepts that are central to the mpadmin interface: *objects*, *attributes*, and *contexts*.

mpadmin Objects and Attributes

From the perspective of mpadmin, a Sun HPC cluster consists of a system of objects, which include

- The cluster itself
- Each node contained in the cluster
- Each partition (logical group of nodes) defined in the cluster

Each type of object has a set of attributes whose values can be operated on via `mpadmin` commands. These attributes control various aspects of their respective objects, such as: whether a node is enabled or disabled (that is, whether it can be used or not), the names of partitions, and which nodes a partition contains.

Note – CRE sets most cluster object attributes to default values each time it boots up. With few exceptions, *do not* change these system-defined values.

`mpadmin` Contexts

`mpadmin` commands are organized into three *contexts*, which correspond to the three types of `mpadmin` objects. These contexts are illustrated in FIGURE 6-1 and summarized below.

- `Cluster` – These commands affect cluster attributes.
- `Partition` – These commands affect partition attributes.
- `Node` – These commands affect node attributes.

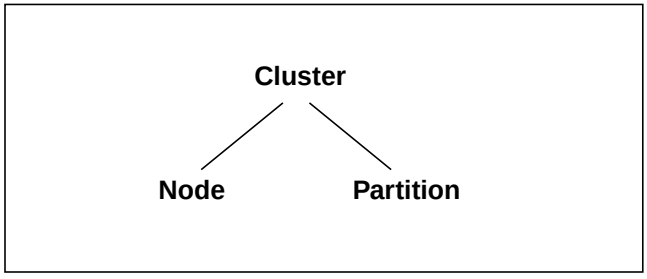

FIGURE 6-1 The `mpadmin` Contexts

Except for `Cluster`, each context is nested in a higher context: `Node` within `Cluster` and `Partition` within `Cluster`.

The mpadmin prompt uses one or more fields to indicate the current context. TABLE 6-2 shows the prompt format for each of the possible mpadmin contexts.

TABLE 6-2 mpadmin Prompt Formats

Prompt Formats	Context
[*cluster-name*]::	Current context = Cluster
[*cluster-name*]Node::	Current context = Node, but not a specific node
[*cluster-name*]N(*node-name*)::	Current context = a specific node
[*cluster-name*]Partition::	Current context = Partition, but not a specific partition
[*cluster-name*]P(*partition-name*)::	Current context = a specific partition

mpadmin Command Overview

This section describes the subcommands that mpadmin provides.

Types of mpadmin Commands

mpadmin provides commands for performing the following operations:

- Configuration control – These commands are used to create and delete mpadmin objects (nodes and partitions).
- Attribute control – These commands are used to set and reset attribute values.
- Context navigation – These commands are used to change the current context to a different context.
- Information retrieval – These commands are used to display object and attribute information.
- Miscellaneous.

Configuration Control

A Sun HPC cluster contains one or more named partitions. Each partition contains some number of specific nodes.

CRE automatically creates the cluster and node objects based on the contents of the `hpc.conf` file. Partitions are the only kind of object that you are required to create and manage.

Use the `delete` command to remove partitions, but no other types of cluster objects. You remove nodes from a Sun HPC cluster by editing the `hpc.conf` file. (See "Adding/Removing Nodes in an Existing Cluster" on page 169.)

create

Usage:

`:: create` *object-name*

Available In: Node, Partition

The `create` command creates a new object with the name *object-name* and makes the new object the current context. Partitions can only be created from within the Partition context.

The following example creates the partition `part0`.

```
[node0] Partition:: create part0
[node0] P(part0)::
```

As the second line in the example shows, `part0` becomes the new context.

delete

Usage:

`:: delete` [*object-name*]

Available In: Node, Partition

The `delete` command deletes the object specified by the *object-name* argument. The object being deleted must either be contained in the current context or must be the current context. The first example shows a partition contained in the current context being deleted.

```
[node0] Partition:: delete part0
[node0] Partition::
```

If the current context is the object to be deleted, the *object-name* argument is optional. In this case, the context reverts to the next higher context level.

```
[node0] P(part0):: delete
[node0] Partition::
```

Attribute Control

Each `mpadmin` object has a set of attributes that can be modified. Use the `set` command to specify a value for a given attribute. Use `unset` to delete an attribute.

Note – CRE requires most attributes to have their default values. Be certain to limit your attribute changes to those described in this chapter.

`set`

Usage:

`:: set` *attribute*`[=`*value*`]`

Available In: Cluster, Node, Partition

The `set` command sets the specified attribute of the current object.

You must be within the context of the target object to set its attributes. For example, to change an attribute of a specific partition, you must be in that partition's context.

To set a literal or numeric attribute, specify the desired value. The following example sets the node attribute for partition `part0`. Setting a partition's node attribute identifies the set of nodes that are members of that partition.

```
[node0] P(part0):: set nodes=node1 node2
[node0] P(part0)::
```

To change the value of an attribute that has already been set, simply set it again. The following example adds `node3` to partition `part0`.

```
[node0] P(part0):: set nodes=+node3
[node0] P(part0)::
```

As shown by this example, if the value of an attribute is a list, items can be added to or removed from the list using the + and - symbols, without repeating items that are already part of the list.

To set a Boolean attribute, specify the name of the Boolean attribute to be activated. Do not include `=`*value* in the expression. The following example enables partition `part0`.

```
[node0] P(part0):: set enabled
[node0] P(part0)::
```

Note – If you mistakenly set a Boolean attribute to a value—that is, if you follow a Boolean attribute's name with the =*value* field, `mpadmin` ignores the value assignment and simply considers the attribute to be active.

`unset`

Usage:

`:: unset` *attribute*

Available In: Cluster, Node, Partition

The `unset` command deletes the specified attribute from the current object. You must be within the context of an object to unset any of its attributes.

The following example disables the partition `part0` (that is, makes it unavailable for use).

```
[node0] P(part0):: unset enabled
[node0] P(part0)::
```

Note – Remember, you cannot use the `set` command to set Boolean attributes to the logical 0 (inactive) state. You must use the `unset` command.

Context Navigation

By default, `mpadmin` commands affect objects that are in the current context—that is, objects that are in the same context in which the command is invoked. For example, if the command `list` is invoked in the Node context, `mpadmin` lists all the nodes in the cluster. If `list` is invoked in the Partition context, it lists all the partitions in the cluster, as shown below:

```
[node0] Partition:: list
          part0
          part1
          part2
[node0] Partition::
```

`mpadmin` provides several context navigation commands that enable you to operate on objects and attributes outside the current context.

current

Usage:

`:: current` *object-name*

Available In: Cluster, Node, Partition

The `current` command changes the current context to the context of the object specified by *object-name*. The target object must exist. That is, if it is a partition, you must already have used the `create` command to create it. If the target object is a cluster or node, it must have been created by CRE.

The following example changes the current context from the general Node context to the context of a specific node, `node1`.

```
[node0] Node:: current node1
[node0] N(node1)::
```

If the name of the target object does not conflict with an `mpadmin` command, you can omit the `current` command. This is illustrated by the following example, where `node1` is the name of the target object.

```
[node0] Node:: node1
[node0] N(hpc-node1)::
```

This works even when the object is in a different context.

```
[node0] Partition:: node1
[node0] N(node1)::
```

Note – The `current` command must be used when the name of the object is the same as an `mpadmin` command. For example, if you have a partition named `Partition`, its name conflicts with the command `Partition`. In this case, to make the object `Partition` the current context, you would need to include the `current` command to make it clear that the `Partition` term refers to the object and is not an invocation of the command.

---

top

Usage:

```
:: top
```

Available In: Node, Partition

The `top` command moves you to the Cluster context. The following example moves from the Partition context to the Cluster context.

```
[node0] Partition:: top
[node0]::
```

up

Usage:

```
:: up
```

Available In: Node, Partition

The `up` command moves you up one level from the current context. The following example moves from the Node context to the context of Cluster. (Since there are only two levels in the object hierarchy, the action of the `up` command is the same as that of the `top` command.)

```
[node0] N[node2] Node:: up
[node0] ::
```

node

Usage:

```
:: node
```

Available In: Cluster

The `node` command moves you from the Cluster context to the Node context.

```
[node0]:: node
[node0] Node::
```

partition

Usage:

```
:: partition
```

Available In: Cluster, Node

The `partition` command moves you from the Cluster or Node context to the Partition context.

```
[node0]:: partition
[node0] Partition::
```

Information Retrieval

The information retrieval commands display information about

- The specified object
- If no object is specified, the current context

`dump`

Usage:

`:: dump` [*object-name*]

Available In: Cluster, Node, Partition

The `dump` command displays the current state of the attributes of the specified object or of the current context. The object can be

- The entire cluster
- A specific partition
- All partitions in the cluster
- A specific node
- All nodes in the cluster

The `dump` command outputs objects in a specific order that corresponds to the logical order of assignment when a cluster is configured. For example, nodes are output before partitions because, when a cluster is configured, nodes must exist before they can be assigned to a partition.

The `dump` command executes in this hierarchical manner so it can be used to back up cluster configurations in a format that allows them to be easily restored at a later time.

The following example shows the `dump` command being used in this way. In this example, it is invoked using the `-c` option on the `mpadmin` command line, with the output being directed to a backup file.

```
# mpadmin -c dump > sunhpc.configuration
```

Later, when it was time to restore the configuration, `mpadmin` could read the backup file as input, using the `-f` option.

```
# mpadmin -f sunhpc.configuration
```

If you wanted to modify the configuration, you could edit the backup file before restoring it.

The following example shows the `dump` command being used to output the attribute states of the partition `part0`.

```
[node0] Partition:: dump part0
        set nodes = node1 node2 node3
        set max_total_procs = 4
        set name = part0
        set enabled
        unset no_login
[node0] Partition::
```

Note – Each attribute is output in the form of a `set` or `unset` command so that the `dump` output functions as a script.

If you are within the context of the object whose attributes you want to see, you do not have to specify its name.

```
[node0] P(part0):: dump
        set nodes = node1 node2 node3
        set max_total_procs = 4
        set enabled
        set name = part0
[node0] P(part0)::
```

list

Usage:

```
:: list
```

Available In: Cluster, Node, Partition

The `list` command lists all of the defined objects in the current context. The following example shows that there are three partitions defined in the Partition context.

```
[node0] Partition:: list
        part0
        part1
        part2
[node0] Partition::
```

show

Usage:

:: show [*object-name*]

Available In: Cluster, Node, Partition

The `show` command displays the current state of the attributes of the specified object *object-name*, which must be in the current context. The following example displays the attributes for the partition `part0`.

```
[node0] Partition:: show part0
        set nodes = node0 node1 node2 node3
        set max_total_procs = 4
        set name = part0
        set enabled
        unset no_login
[node0] Partition::
```

If, in the above example, you attempted to `show node1`, the operation fails because `node1` is not in the current context.

Miscellaneous Commands

This section describes the `mpadmin` commands `connect`, `echo`, `help`, and `quit/exit`.

connect

Usage:

:: connect *cluster-name*

Available In: Cluster

In order to access any objects or attributes in a Sun HPC cluster, you must be *connected* to the cluster.

However, connecting to a cluster ordinarily happens automatically, so you are not likely to ever need to use the `connect` command.

The environment variable `SUNHPC_CLUSTER` names a default cluster. If no other action is taken to override this default, any `mpadmin` session will connect to the cluster named by this environment variable.

If you issue the `mpadmin` command on a node that is part of a cluster, you are automatically connected to that cluster, regardless of the `SUNHPC_CLUSTER` setting.

If you are not logged in to the cluster you want to use and you do not want to use the default cluster, you can use the `mpadmin -s` option, specifying the name of the cluster of interest as an argument to the option.

Note – When CRE creates a cluster, it always names it after the hostname of the cluster's master node—that is, the node on which the master daemons are running. Therefore, whenever you need to specify the name of a cluster, use the hostname of the cluster's master node.

The following example shows the `connect` command being used to connect to a cluster whose master node is `node0`.

```
[hpc-demo]:: connect node0
[node0]::
```

echo

Usage:

`:: echo` *text–message*

Available In: Cluster, Node, Partition

The `echo` command prints the specified text on the standard output. If you write a script to be run with `mpadmin -f`, you can include the `echo` command in the script so that it will print status information as it executes.

```
[node0]:: echo Enabling part0 and part1
Enabling part0 and part1
[node0]::
```

help

Usage:

:: help [*command*]

Available In: Cluster, Node, Partition

When invoked without a command argument, the `help` command lists the `mpadmin` commands that are available within the current context. The following example shows `help` being invoked at the `Cluster` level.

```
[node0]:: help
connect <cluster-name>connect to a Sun HPC cluster
set <attribute>[=value]set an attribute in the current context
unset <attribute>delete an attribute in the current context
show          show attributes in current context
dump          show all objects on the cluster
node          go to the node context
partition     go to the partition context
echo ...      print the rest of the line on standard output
quit          quit mpadmin
help [command]show information about command command
? [command]   show information about command command
[node0]::
```

To get a description of a particular command, enter the command name as an argument to `help`.

If you specify a context command (`node` or `partition`), `mpadmin` lists the commands available within that context.

```
[node0]:: help node
current <node>set the current node for future commands
create <node> create a new node with the given name
delete [node] delete a node
list          list all the defined nodes
show [node]   show a node's attributes
dump [node]   show attributes for a node
set <attribute>[=value]  set the current node's attribute
unset <attribute>delete the current node's attribute
up            go up to the Cluster level command prompt
top           go up to the Cluster level command prompt
echo ...      print the rest of the line on standard output
help [command]show information about command command
? [command]   show information about command command
[node0]::
```

The "?" character is a synonym for `help`.

quit/exit

Usage:

```
:: quit
:: exit
```

Available In: Cluster, Node, Partition

Entering either `quit` or `exit` causes `mpadmin` to terminate and return you to the shell level.

Example:

```
[node0]:: quit
#
```

Example:

```
[node0] N(node2):: exit
#
```

Additional `mpadmin` Functionality

This section describes other functionality provided by `mpadmin`.

Multiple Commands on a Line

Because `mpadmin` interprets its input, if you issue more than one command on a line, `mpadmin` will execute them sequentially in the order they appear.

The following example shows how to display a list of nodes when not in the Node context. The `node` command switches to the Node context and the `list` command generates a list for that context.

```
[node0]:: node list
        node0
        node1
        node2
        node3
[node0] Node::
```

The following example sets the `enabled` attribute on partition `part1`. The `part1` entry acts as a command that switches the context from `part0` to `part1` and the `set` command turns on the `enabled` attribute.

```
[node0] P[part0]:: part0 set enabled
[node0] P(part0)::
```

Command Abbreviation

You can abbreviate commands to the shortest string of at least two letters so long as it is still unique within the current context.

```
[node0] Node:: pa
[node0] Partition:: li
      part0
      part1
      part2
      part3
[node0] Partition:: part2
[node0] P(part2):: sh
      set enabled
      set max_total_procs = 4
      set name = part2
      set nodes = node0 node1
[node0] P(part2)::
```

Note – The names of objects cannot be abbreviated.

Using `mpadmin`

This section explains how to use `mpadmin` to perform the principal administrative tasks involved in setting up and maintaining a Sun HPC cluster. It describes the following tasks:

- Logging in to the cluster
- Customizing cluster-level attributes
- Managing nodes
- Managing partitions

Note on Naming Partitions and Custom Attributes

You can assign names to partitions and to *custom attributes*. Custom attributes are attributes that are not part of the default CRE database; they are discussed in "Setting Custom Attributes" on page 101.

Names must start with a letter and are case sensitive. The following characters can be used:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789–_.

The only limit to name length is the limit imposed by Solaris on host names—it is ordinarily set at 256 characters.

Note – Do not begin an attribute name with the characters `mp_`. This starting sequence is reserved by CRE.

Nodes and partitions have separate name spaces. Thus, you can have a partition named `Parallel` that contains a node named `Parallel`.

Logging Into the Cluster

It is assumed that you are logged in to a node that is part of the cluster you want to set up. If the node you are logged in to is not part of any cluster, set the `SUNHPC_CLUSTER` environment variable to the name of the target cluster. For example,

```
# setenv SUNHPC_CLUSTER node0
```

makes `node0` the default cluster. Remember, a cluster's name is the same as the host name of its master node.

Once you are connected to the cluster, you can start using `mpadmin` to perform the administrative tasks described below.

When you start up an `mpadmin` interactive session, you begin at the Cluster level. TABLE 6-3 lists the `mpadmin` commands that can be used in the Cluster context.

TABLE 6-3 Cluster-Level `mpadmin` Commands

Command	Synopsis
`connect` *cluster-name*	Connect to a Sun HPC cluster named *cluster-name*. You will not need to use this command.
`show`	Show cluster attributes.
`dump`	Show all objects in the Sun HPC cluster.
`set` *attribute[=value]*	Set a cluster-level attribute.
`unset` *attribute*	Delete a cluster-level attribute.
`node`	Enter the `Node` context.
`partition`	Enter the `Partition` context.
`echo` ...	Print the rest of the line on the standard output.
`quit / exit`	Quit `mpadmin`.
`help` [*command*] `/ ?`	Show information about commands.

Customizing Cluster-Level Attributes

This section describes various Cluster-level attributes that you may want to modify. TABLE 6-4 lists the attributes that can be changed in the Cluster context.

TABLE 6-4 Cluster-Level Attributes

Attribute	Kind	Description
`default_interactive_partition`	Value	Specifies the default partition.
`logfile`	Value	Specifies an optional output file for logging CRE daemon error messages.
`administrator`	Value	Specifies an email address for the system administrator(s).

`default_interactive_partition`

This attribute specifies the default partition for running MPI jobs. Its value is used by the command `mprun`, which is described in the *Sun HPC ClusterTools User's Guide*.

For example, to make a partition named `part0` the default partition, enter the following in the Cluster context:

```
[node0]:: set default_interactive_partition=part0
```

When a user executes a program via `mprun`, CRE decides where to run the program, based on the following criteria:

1. Check for the command-line `-p` option. If a partition is specified, execute the program in that partition. If the specified partition is invalid, the command fails.
2. Check to see if the `MPRUN_FLAGS` environment variable specifies a default partition. If so, execute the program in that partition. If the specified partition is invalid, the command fails.
3. Check to see if the `SUNHPC_PART` environment variable has a value set. If it specifies a default partition, execute the program in that partition. If the specified partition is invalid, then check to see if the user is logged in to any partition. If so, execute the program in that partition.
4. Check to see if the user is logged in to a partition. Execute the program in that partition.
5. If none of these checks yields a partition name, check for the existence of the `default_interactive_partition` attribute. If it specifies a partition, execute the program in that partition.

logfile

The `logfile` attribute allows you to log CRE messages in a file separate from all other system messages. For example, if you enter:

```
[node0]:: set logfile=/home/wmitty/cre-messages
```

CRE outputs its messages to the file `/home/wmitty/cre-messages`. If `logfile` is not set, CRE messages are passed to `syslog`, which stores them with other system messages in `/var/adm/messages`.

Note – A full path name must be specified when setting the `logfile` attribute.

---

administrator

Set the `administrator` attribute to identify the system administrator. For example, to specify the email address of the system administrator:

```
[node0]:: set administrator="root@example.com"
```

Note the use of double quotation marks.

Managing Nodes

Ordinarily, the only administrative action that you need to take with nodes is to enable them for use. Or, if you want to temporarily make a node unavailable for use, disable it.

Other node-related administrative tasks—such as naming the nodes, identifying the master node, setting memory and process limits, and setting the node's partition attribute—are either handled by CRE automatically or are controlled via partition-level attributes.

Node Commands

The table below lists the `mpadmin` commands that can be used at the Node level.

TABLE 6-5 Node-Level `mpadmin` Commands

Command	Synopsis
`current` *node*	Set the context to the specified node for future commands.
`create` *node*	Create a new node with the given name.
`delete` [*node*]	Delete a node.
`list`	List all the defined nodes.
`show` [*node*]	Show a node's attributes.
`dump` [*node*]	Show the attributes of the node.
`set` *attribute*[=*value*]	Set the specified attribute of the current node.
`unset` *attribute*	Delete the specified attribute of the current node.
`up`	Move to the next higher level (`Top`) command context.
`top`	Move to the `Top` level command context.
`echo` ...	Print the rest of the line on the standard output.
`help` [*command*]	Show information about commands (`?`).

Node Attributes

Nodes are defined by many attributes, most of which are not accessible to `mpadmin` commands. Although you are not able to affect these attributes, it can be helpful to know of their existence and meaning; hence, they are listed and briefly described in TABLE 6-6.

TABLE 6-7 lists the Node-level attributes that *can* be set via `mpadmin` commands. However, `enabled` and `max_total_procs` are the only node attributes that you can safely modify.

TABLE 6-6 Node Attributes That Cannot Be Set by the System Administrator

Attribute	Kind	Description
`cpu_idle`	Value	Percent of time CPU is idle.
`cpu_iowait`	Value	Percent of time CPU spent in I/O wait state.
`cpu_kernel`	Value	Percent of time CPU spent in kernel state.
`cpu_type`	Value	Type of CPU, for example, `sparc`.
`cpu_user`	Value	Percent of time CPU spends running user's program.
`load1`	Value	Load average for the past minute.
`load5`	Value	Load average for the past five minutes.
`load15`	Value	Load average for the past 15 minutes.
`manufacturer`	Value	Manufacturer of the node, e.g., `Sun_Microsystems`.
`mem_free`	Value	Node's available RAM (in Mbytes).
`mem_total`	Value	Node's total physical memory (in Mbytes).
name	Value	Name of the node; this is predefined and must not be set via `mpadmin`.
`ncpus`	Value	Number of CPUs in the node.
`offline`	Boolean	Set automatically by the system if the `tm.spmd` daemon on the node stops running or is unresponsive; if set, prevents jobs from being spawned on the node.
`os_arch_kernel`	Value	Node's kernel architecture (same as output from `arch -k`, for example, `sun4u`).
`os_name`	Value	Name of the operating system running on the node.
`os_release`	Value	Operating system's release number.
`os_release_maj`	Value	Operating system's major release number.
`os_release_min`	Value	Operating system's minor release number,.
`serial_number`	Value	Hardware serial number or host id.
`swap_free`	Value	Node's available swap space (in Mbytes).
`swap_total`	Value	Node's total swap space (in Mbytes).
`update_time`	Value	When this information was last updated.

TABLE 6-7 Node Attributes That Can Be Set by the System Administrator

Attribute	Kind	Description
`enabled`	Boolean	Set if the node is enabled, that is, if it is ready to accept jobs.
`master`	Boolean	Specify node on which the master daemons are running as an argument to `mprun`.
`max_locked_mem`	Value	Maximum amount of shared memory allowed to be locked down by Sun MPI processes (in Kbytes).
`max_total_procs`	Value	Maximum number of Sun HPC processes per node.
`min_unlocked_mem`	Value	Minimum amount of shared memory not to be locked down by Sun MPI processes (in Kbytes).
`partition`	Value	Partition of which node is a member.
`shmem_minfree`	Value	Fraction of swap space kept free for non-MPI use.

`enabled`

The attribute `enabled` is set by default when CRE daemons start on a node. Unsetting it prevents new jobs from being spawned on the node.

A partition can list a node that is not enabled as a member. However, jobs will execute on that partition as if that node were not a member.

`master`

Note – You must not change this node attribute. CRE automatically sets it to the hostname of the node on which the master CRE daemons are running. This happens whenever the CRE daemons start.

`max_locked_mem` *and* `min_unlocked_mem`

Note – You should not change these node attributes. They are described here so that you can interpret their values when node attributes are displayed via the `dump` or `show` commands.

The `max_locked_mem` and `min_unlocked_mem` attributes limit the amount of shared memory available to be locked down for use by Sun MPI processes. Locking down shared memory guarantees maximum speed for Sun MPI processes by

eliminating delays caused by swapping memory to disk. However, locking physical memory can have undesirable side effects because it prevents that memory from being used by other processes on the node.

The Solaris software provides two related tunable kernel parameters:

- `tune_t_minasmem`, which is similar to `min_unlocked_mem`
- `pages_pp_maximum`, which is similar to `max_locked_mem`

The CRE parameters impose limits only on MPI programs, while the kernel parameters limit all processes. Also, the kernel parameter units are pages rather than Kbytes. Refer to your Solaris documentation for more information about `tune_t_minasmem` and `pages_pp_maximum`.

max_total_procs

You limit the number of `mprun` processes allowed to run concurrently on a node by setting this attribute to an integer.

```
[node0] P(part0):: set max_total_procs=10
[node0] P(part0)::
```

If `max_total_procs` is set at the node level, that value overrides any value set at the partition level.

By default, `max_total_procs` is `unset`. CRE does not impose any limit on the number of processes allowed on a node.

partition

Note – There is no need to set this attribute. CRE sets it automatically if the node is included in any partition configuration(s).

A node can belong to multiple partitions, but only one of those partitions can be enabled at a time. No matter how many partitions a node belongs to, the `partition` attribute shows only one partition name—that name is always the name of the enabled partition, if one exists for that node.

shmem_minfree

Note – You should not change this node attribute. It is described here so that you can interpret its value when node attributes are displayed via the `dump` or `show` commands.

The `shmem_minfree` attribute reserves some portion of the `/tmp` file system for non-MPI use.

For example, if `/tmp` is 1 Gbyte and `shmem_minfree` is set to 0.2, any time free space on `/tmp` drops below 200 Mbytes (1 Gbyte * 0.2), programs using the MPI shared memory protocol will not be allowed to run.

This attributeexists on both nodes and partitions. If they are set to different values, the node attribute overrides the partition attribute.

Deleting Nodes

If you permanently remove a node from the Sun HPC cluster, you should then delete the corresponding node object from the CRE resource database.

Recommendations

Before deleting a node:

- Remove it from any enabled partition by unsetting its `partition` attribute (automatically removing the node from the partition's `nodes` attribute list), or by removing it from the partition's `nodes` attribute list. See "Partition Attributes" on page 97 for details.
- Wait for any jobs running on it to terminate, or stop them using the `mpkill` command, which is described in the *Sun HPC ClusterTools User's Guide*.

Using the `delete` *Command*

To delete a node, use the `delete` command within the context of the node you want to delete.

```
[node0] N(node3):: delete
[node0] Node::
```

Managing Partitions

Partitions are logical collections of nodes that work cooperatively to run programs on the Sun HPC cluster. An MPI job can run on a single partition or on the combination of a single partition and one or more nodes that are not members of any partition. MPI jobs cannot run in multiple partitions.

You must create a partition and enable it before you can run MPI programs on your Sun HPC cluster. Once a partition is created, you can configure it to meet the specific needs of your site and enable it for use.

Once a partition is created and enabled, you can run serial or parallel jobs on it. Serial programs run on a single node of a partition. Parallel programs run on any number of nodes of a partition in parallel.

CRE performs load balancing on shared partitions. When you use `mprun` to execute a program on a shared partition, CRE automatically runs it on the least-loaded nodes that satisfy any specified resource requirements.

Partitions are mutable. That is, after you create and configure a partition, you can change it if your site requirements change. You can add nodes to a partition or remove them. You can change a partition's attributes. Also, since you can enable and disable partitions, you can have many partitions defined and use only a few at a time according to current needs.

There are no restrictions on the number or size of partitions, so long as no node is a member of more than one enabled partition.

Partition Commands

TABLE 6-8 lists the mpadmin commands that can be used within the partition context.

TABLE 6-8 Partition-Level `mpadmin` Commands

Command	Synopsis
`current` *partition*	Set the context to the specified partition for future commands.
`create` *partition*	Create a new partition with the given name.
`delete` [*partition*]	Delete a partition.
`list`	List all the defined partitions.
`show` [*partition*]	Show a partition's attributes.
`dump` [*partition*]	Show the attributes of a partition.
`set` *attribute*[=*value*]	Set the current partition's attribute.
`unset` *attribute*	Delete the current partition's attribute.
`up`	Move up one level in the context hierarchy.
`top`	Move to the top level in the context hierarchy.
`echo` ...	Print the rest of the line on the standard output.
`help` [*command*]	Show information about the command *command*.
`?` [*command*]	Show information about the command *command*.

Viewing Existing Partitions

Before creating a new partition, you might want to list the partitions that have already been created. To do this, use the `list` command from within the Partition context.

```
[node0] Partition:: list
        part0
        part1
[node0] Partition::
```

Creating a Partition

To create a partition, use the `create` command, followed by the name of the new partition. "Note on Naming Partitions and Custom Attributes" on page 86 discusses the rules for naming partitions.

For example:

```
[node0] Partition:: create part0
[node0] P(part0)::
```

The `create` command automatically changes the context to that of the new partition.

At this point, your partition exists by name but contains no nodes. You must assign nodes to the partition before using it. You can do this by setting the partition's `nodes` attribute. Note these prerequisites:

- Nodes have to exist in the CRE database before you can add them to partitions.
- A node must be enabled for it to be an active member of a partition. If a node is configured as a partition member, but is not enabled, it will not participate in jobs that run on that partition.

Configuring Partitions

You can configure partitions by setting and deleting their attributes using the `set` and `unset` commands. TABLE 6-9 shows the commonly used partition attributes.

You can combine these attributes in any way that makes sense for your site.

TABLE 6-9 Common Partitions and Their Attributes

Partition Type	Relevant Attributes	Recommended Value
Login	`no_logins`	not set
Shared	`max_total_procs`	not set or set greater than 1
Dedicated	`max_total_procs`	=1
	`no_logins`	set
Serial	`no_mp_jobs`	set
Parallel	`no_mp_jobs`	not set

Partitions, once created, can be enabled and disabled. This lets you define many partitions but use just a few at a time. For instance, you might want to define a number of shared partitions for development use and dedicated partitions for executing production jobs, but have only a subset available for use at a given time.

Partition Attributes

TABLE 6-10 lists the predefined partition attributes. To see their current values, use the `mpadmin show` command.

TABLE 6-10 Predefined Partition Attributes

Attribute	Kind	Description
`enabled`	Boolean	Set if the partition is enabled, that is, if it is ready to accept logins or jobs.
`max_locked_mem`	Value	Maximum amount of shared memory allowed to be locked down by MPI processes (in Kbytes).
`max_total_procs`	Value	Maximum number of simultaneously running processes allowed on each node in the partition.
`min_unlocked_mem`	Value	Minimum amount of shared memory that may not be locked down by MPI processes (in Kbytes).
`name`	Value	Name of the partition.
`no_logins`	Boolean	Disallow logins.
`no_mp_tasks`	Boolean	Disallow multiprocess parallel jobs.
`nodes`	Value	List of nodes in the partition.
`shmem_minfree`	Value	Fraction of swap space kept free for non-MPI use.

`enabled`

Set the `enabled` attribute to make a partition available for use.

By default, the `enabled` attribute is *not* set when a partition is created.

`max_locked_mem` *and* `min_unlocked_mem`

You should not change these partition attributes.

`max_total_procs`

To limit the number of simultaneously running `mprun` processes allowed on each node in a partition, set the `max_total_procs` attribute in a specific Node context or in the Partition context.

```
[node0] P(part0):: set max_total_procs=10
[node0] P(part0)::
```

You can set `max_total_procs` if you want to limit the load on a partition. For example, if `max_total_procs` is set to 3 and there are three nodes in the partition, then the maximum `mprun -np` value for programs running in that partition is 9. By default, `max_total_procs` is `unset`.

If `max_total_procs` is set at the node level, that value overrides any value set at the partition level.

CRE does not impose any limit on the number of processes allowed on a node.

name

The `name` attribute is set when a partition is created. To change the name of a partition, set its `name` attribute to a new name.

```
[node0] P(part0):: set name=part1
[node0] P(part1)::
```

See "Note on Naming Partitions and Custom Attributes" on page 86 for partition naming rules.

no_logins

To prohibit users from logging in to a partition, set the `no_logins` attribute.

```
[node0] P(part1):: set no_logins
[node0] P(part1)::
```

no_mp_jobs

To prohibit multiprocess parallel jobs from running on a partition—that is, to make a serial partition—set the `no_mp_jobs` attribute.

```
[node0] P(part1):: set no_mp_jobs
[node0] P(part1)::
```

nodes

To specify the nodes that are members of a partition, set the partition's `nodes` attribute.

```
[node0] P(part1):: set nodes=node1
[node0] P(part1):: show
        set nodes = node1
        set enabled
[node0] P(part1)::
```

The value you give the nodes attribute defines the entire list of nodes in the partition. To add a node to an already existing node list without retyping the names of nodes that are already present, use the + (plus) character.

```
[node0] P(part1):: set nodes=+node2 node3
[node0] P(part1):: show
        set nodes = node0 node1 node2 node3
        set enabled
[node0] P(part1)::
```

Similarly, you can use the – (minus) character to remove a node from a partition.

To assign a range of nodes to the nodes attribute, use the : (colon) syntax. This example assigns to part0 all nodes whose names are alphabetically greater than or equal to node0 and less than or equal to node3:

```
[node0] P(part1):: set nodes = node0:node3
[node0] P(part1)::
```

Setting the nodes attribute of an enabled partition has the side effect of setting the partition attribute of the corresponding nodes. Continuing the example, setting the nodes attribute of part1 affects the partition attribute of node2:

```
[node0] P(part1):: node node2
[node0] N(node2):: show
        set partition = part1
[node0] N(node2)::
```

A node cannot be a member of more than one enabled partition. If you try to add a node that is already in an enabled partition, mpadmin returns an error message.

```
[node0] P(part1):: show
        set nodes = node0 node1 node2 node3
        set enabled
[node0] P(part1):: current part0
[node0] P(part0):: set enabled
[node0] P(part0):: set nodes=node1
mpadmin: node1 must be removed from part1 before it can be added to
part0
```

Unsetting the nodes attribute of an enabled partition has the side effect of unsetting the partition attribute of the corresponding node.

Unsetting the nodes attribute of a disabled partition removes the nodes from the partition but does not change their partition attributes.

shmem_minfree

Use the shmem_minfree attribute to reserve some portion of the /tmp file system for non-MPI use.

For example, if /tmp is 1 Gbyte and shmem_minfree is set to 0.2, any time free space on /tmp drops below 200 Mbytes (1 Gbyte * 0.2), programs using the MPI shared memory protocol will not be allowed to run.

This attribute exists on both nodes and partitions. If both are set, the node's shmem_minfree attribute overrides the partition's shmem_minfree attribute.

Enabling Partitions

A partition must be enabled before users can run programs on it. Before enabling a partition, you must disable any partitions that share nodes with the partition that you are about to enable.

To enable a partition, set its enabled attribute.

```
[node0] P(part0):: set enabled
[node0] P(part0)::
```

Now the partition is ready for use.

Enabling a partition has the side effect of setting the partition attribute of every node in that partition.

If you try to enable a partition that shares a node with another enabled partition, mpadmin prints an error message.

```
[node0] P(part1):: show
        set nodes = node1 node2 node3
        set enabled
[node0] P(part1):: current part2
[node0] P(part2):: show
        set nodes = node1
[node0] P(part2):: set enabled
mpadmin: part1/node1: partition resource conflict
```

Disabling Partitions

To disable a partition, unset its enabled attribute.

```
[node0] P(part0):: unset enabled
[node0] P(part0)::
```

Now the partition can no longer be used.

Any jobs that are running on a partition when it is disabled will continue to run. After disabling a partition, you should either wait for any running jobs to terminate or stop them using the `mpkill` command. This is described in the *Sun HPC ClusterTools User's Guide*.

Deleting Partitions

Delete a partition when you do not plan to use it anymore.

Note – Although it is possible to delete a partition without first disabling it, you should disable the partition by unsetting its `enabled` attribute before deleting it.

To delete a partition, use the `delete` command in the context of the partition you want to delete.

```
[node0] P(part0):: delete
[node0] Partition::
```

Setting Custom Attributes

Sun HPC ClusterTools software does not limit you to the attributes listed. You can define new attributes as desired.

For example, if a node has a special resource that will not be flagged by an existing attribute, you may want to set an attribute that identifies that special characteristic. In the following example, node `node3` has a frame buffer attached. This feature is captured by setting the custom attribute `has_frame_buffer` for that node.

```
[node0] N(node3):: set has_frame_buffer
[node0] N(node3)::
```

Users can then use the attribute `has_frame_buffer` to request a node that has a frame buffer when they execute programs. For example, use the following `mprun` command lines to select a node with or without a frame buffer, respectively:

```
% mprun -R "has_frame_buffer"
% mprun -R "\!has_frame_buffer"
```

See "Note on Naming Partitions and Custom Attributes" on page 86 for restrictions on attribute names.

CHAPTER 7

hpc.conf Configuration File

This chapter discusses the Sun HPC ClusterTools configuration file `hpc.conf`, which defines various attributes of a Sun HPC cluster. A single `hpc.conf` file is shared by all the nodes in a cluster. It resides in `/opt/SUNWhpc/etc`.

Note – This configuration file is also used on LSF-based clusters, but it resides in a different location. See Appendix C.

`hpc.conf` is organized into functional sections, which are summarized below and illustrated in TABLE 7-1.

- `ShmemResource` section defines certain shared memory attributes.
- `MPIOptions` section defines certain MPI parameters.
- `CREOptions` section controls the logging of system events, the handling of daemon core files, and the authentication options.
- `PMODULES` section names and locates the protocol modules that are available for communication on the cluster.
- `PM` sections list and rank the protocol modules that are available for communication on the cluster.
- `PFSFileSystem` section names and defines all parallel file systems in the cluster.
- `PFSServers` section names and defines all parallel file system servers in the cluster.
- `HPCNodes` section is not used by CRE. It applies only in an LSF-based runtime environment (as described in Appendix C).

Sun HPC ClusterTools software is distributed with an `hpc.conf` template, which resides by default in `/opt/SUNWhpc/examples/rte/hpc.conf.template`. If you wish to customize the configuration settings, you should copy this template file to `/opt/SUNWhpc/etc/hpc.conf` and edit it.

Each configuration section is bracketed by a `Begin/End` keyword pair and, when a parameter definition involves multiple fields, the fields are separated by spaces.

TABLE 7-1 General Organization of the `hpc.conf` File

```
# Begin ShmemResource
# ...
# End ShmemResource

# Begin MPIOptions Queue=
# ...
# End MPIOptions

# Begin CREOptions Server=
# ...
# End CREOptions

# Begin PFSFileSystem=
# ...
# End PFSFileSystem

# Begin PFSServers
# ...
# End PFSServers

# Begin HPCNodes
# ...
# End HPCNodes

Begin PMODULES
...
End PMODULES

Begin PM=shm
...
End PM

Begin PM=rsm
...
End PM

Begin PM=tcp
...
End PM
```

Note – When any changes are made to `hpc.conf`, the system should be in a quiescent state. To ensure that it is safe to edit `hpc.conf`, shut down the nodal and master CRE daemons as described in "Stopping and Restarting CRE" on page 14. If you change the `PFSFileSystem` or `PFSServers` sections, you must also unmount any PFS file systems first. See Chapter 4 for details. If you change the `PMODULES` or `PM=rsm` sections, you must also stop the RSM daemon `hpc_rsmd`. See "RSM Daemon" on page 19.

`ShmemResource` Section

The `ShmemResource` section provides the administrator with two parameters that control allocation of shared memory and swap space: `MaxAllocMem` and `MaxAllocSwap`. This special memory allocation control is needed because some Sun HPC ClusterTools components use shared memory.

TABLE 7-2 shows the `ShmemResource` template that is in the `hpc.conf` file that is shipped with Sun HPC ClusterTools software.

TABLE 7-2 `ShmemResource` Section Example

```
#Begin ShmemResource
#MaxAllocMem  0x7fffffffffffffff
#MaxAllocSwap 0x7fffffffffffffff
#End ShmemResource
```

To set `MaxAllocMem` and/or `MaxAllocSwap` limits, remove the comment character (#) from the start of each line and replace the current value, `0x7fffffffffffffff`, with the desired limit.

Guidelines for Setting Limits

The Sun HPC ClusterTools internal shared memory allocator permits an application to use swap space, the amount of which is the smaller of:

- The value (in bytes) given by the `MaxAllocSwap` parameter
- 90% of available swap on a node

If `MaxAllocSwap` is not specified, or if zero or a negative value is specified, 90% of a node's available swap is used as the swap limit.

The `MaxAllocMem` parameter can be used to limit the amount of shared memory that can be allocated. If a smaller shared memory limit is not specified, the shared memory limit is 90% of available physical memory.

The following Sun HPC ClusterTools components use shared memory:

- CRE uses shared memory to hold cluster and job table information. Its memory use is based on cluster and job sizes and is not controllable by the user. Shared memory space is allocated for CRE when it starts up and is not affected by `MaxAllocMem` and `MaxAllocSwap` settings. This ensures that CRE can start up no matter how low these memory-limit variables have been set.
- MPI uses shared memory for communication between processes that are on the same node.
- Sun S3L uses shared memory for storing data. An MPI application can allocate parallel arrays whose subgrids are in shared memory. This is done with the utility `S3L_declare_detailed()`.

Note – Sun S3L supports a special form of shared memory known as *Intimate Shared Memory* (ISM), which reserves a region in physical memory for shared memory use. What makes ISM space special is that it is not swappable and, therefore, cannot be made available for other use. For this reason, the amount of memory allocated to ISM should be kept to a minimum.

Note – Shared memory and swap space limits are applied per-job on each node.

If you have set up your system for dedicated use (only one job at a time is allowed), you should leave `MaxAllocMem` and `MaxAllocSwap` undefined. This allows jobs to maximize use of swap space and physical memory.

If, however, multiple jobs will share a system, you may want to set `MaxAllocMem` to some level below 50% of total physical memory. This reduces the risk of having a single application lock up physical memory. How much below 50% you choose to set it depends on how many jobs you expect to be competing for physical memory at any given time.

Note – When users make direct calls to `mmap`(2) or `shmget`(2), they are not limited by the `MaxAllocMem` and `MaxAllocSwap` variables. These utilities manipulate shared memory independently of the `MaxAllocMem` and `MaxAllocSwap` values.

MPIOptions Section

The MPIOptions section provides a set of options that control MPI communication behavior in ways that are likely to affect message-passing performance. It contains a template showing some general-purpose option settings, plus an example of alternative settings for maximizing performance. These examples are shown in TABLE 7-3.

- **G**eneral-purpose, multiuser settings – The template in the MPIOptions section is designed for general-purpose use at times when multiple message-passing jobs will be running concurrently.
- Performance settings – The second example is designed to maximize the performance of message-passing jobs when only one job is allowed to run at a time.

Note – The first line of the template contains the phrase "Queue=hpc." This line indicates a queue in the LSF batch runtime environment, which uses the same hpc.conf file as CRE. For LSF, the settings apply only to the specified queue. For CRE, the settings apply across the cluster.

The options in the general-purpose template are the same as the default settings for the Sun MPI library. In other words, you do not have to uncomment the general-purpose template to have its option values be in effect. This template is provided in the MPIOptions section so that you can see what options are most beneficial when operating in a multiuser mode.

If you want to use the performance settings, do the following:

- Delete the comment character (#) from the beginning of each line of the performance example, including the Begin MPIOptions and End MPIOptions lines.
- On CRE-based clusters, delete the "Queue=performance" phrase from the Begin MPIOptions line.

The resulting template should appear as follows:

```
Begin MPIOptions
coscheduling  off
spin          on
End MPIOptions
```

TABLE 7-3 `MPIOptions Section` Example

```
# The following is an example of the options that affect the run time
# environment of the MPI library.  The listings below are identical to
# the default settings of the library.  The "Queue=hpc" phrase makes
# this an LSF-specific entry, and only for the Queue named hpc.  These
# options are a good choice for a multiuser Queue.  To be recognized
# by CRE, the "Queue=hpc" needs to be removed.
#
# Begin MPIOptions Queue=hpc
# coscheduling  avail
# pbind        avail
# spindtimeout   1000
# progressadjust   on
# spin  off
#
# shm_numpostbox           16
# shm_shortmsgsize        256
# rsm_maxsegsize      1048576
# rsm_numpostbox           15
# rsm_shortmsgsize        401
# rsm_maxstripe            2
# rsm_links          wrsm0,1
# maxprocs_limit   2147483647
# maxprocs_default       4096
#
# End MPIOptions

# The listing below is a good choice when trying to get maximum
# performance out of MPI jobs that are running in a Queue that
# allows only one job to run at a time.
#
# Begin MPIOptions Queue=performance
# coscheduling             off
# spin                      on
# End MPIOptions
```

TABLE 7-4 provides brief descriptions of the MPI runtime options that can be set in `hpc.conf`. Each description identifies the default value and describes the effect of each legal value.

Some MPI options not only control a parameter directly, they can also be set to a value that passes control of the parameter to an environment variable. Where an MPI option has an associated environment variable, TABLE 7-4 names the environment variable.

TABLE 7-4 MPI Runtime Options

Option	Values		Description
	Default	Other	
coscheduling	`avail`		Allows `spind` use to be controlled by the environment variable `MPI_COSCHED`. If `MPI_COSCHED`=0 or is not set, `spind` is not used. If `MPI_COSCHED`=1, `spind` must be used.
		`on`	Enables coscheduling; `spind` is used. This value overrides `MPI_COSCHED=0`.
		`off`	Disables coscheduling; `spind` is not to be used. This value overrides `MPI_COSCHED=1`.
`pbind`	`avail`		Allows processor binding state to be controlled by the environment variable `MPI_PROCBIND`. If `MPI_PROCBIND=0` or is not set, no processes will be bound to a processor. This is the default. If `MPI_PROCBIND=1`, all processes on a node will be bound to a processor.
		`on`	All processes will be bound to processors. This value overrides `MPI_PROCBIND=0`.
		`off`	No processes on a node are bound to a processor. This value overrides `MPI_PROCBIND=1`.
`spindtimeout`	`1000`		When polling for messages, a process waits 1000 milliseconds for `spind` to return. This equals the value to which the environment variable `MPI_SPINDTIMEOUT` is set.
		integer	To change the default timeout, enter an integer value specifying the number of milliseconds the timeout should be.
`progressadjust`	`on`		Allows user to set the environment variable `MPI_SPIN`.
		`off`	Disables user's ability to set the environment variable `MPI_SPIN`.

TABLE 7-4 MPI Runtime Options *(Continued)*

Option	Values		Description
	Default	Other	
`shm_numpostbox`	`16`		Sets to 16 the number of postbox entries that are dedicated to a connection endpoint. This equals the value to which the environment variable `MPI_SHM_NUMPOSTBOX` is set. (See the *Sun HPC ClusterTools Performance Guide* for details.)
		integer	To change the number of dedicated postbox entries, enter an integer value specifying the desired number.
`shm_shortmsgsize`	`256`		Sets to 256 the maximum number of bytes a short message can contain. This equals the default value to which the environment variable `MPI_SHM_SHORTMSGSIZE` is set.
		integer	To change the maximum-size definition of a short message, enter an integer specifying the maximum number of bytes it can contain.
`rsm_numpostbox`	`15`		Sets to 15 the number of postbox entries that are dedicated to a connection endpoint. This equals the value to which the environment variable `MPI_RSM_NUMPOSTBOX` is set.
		integer	To change the number of dedicated postbox entries, enter an integer specifying the desired number.
`rsm_shortmsgsize`	`401`		Sets to 401 the maximum number of bytes a short message can contain when sent via RSM without using buffers. This equals the value to which the environment variable `MPI_RSM_SHORTMSGSIZE` is set.
		integer	To change the maximum-size definition of a short message, enter an integer specifying the maximum number of bytes it can contain.

TABLE 7-4 MPI Runtime Options *(Continued)*

Option	Values		Description
	Default	Other	
`rsm_maxstripe`	`2`		Sets to 2 the maximum number of interfaces per stripe that can be used. (The default is the number of interfaces in the cluster, with a maximum of 64.) This equals the value to which the environment variable `MPI_RSM_MAXSTRIPE` is set.
		integer	To change the maximum number of stripes that can be used, enter an integer specifying the desired limit.
`rsm_links`	`wrsm0,1`		Defines the controllers/links that can be used on each node for RSM communication.
`rsm_maxsegsize`	(see note at right)		Sets to 17179869184 the maximum size segment that can be created for communication purposes.
		integer	To change the maximum size of an RSM segment, enter an integer specifying the desired limit.
`maxprocs_default`	`4096`		Sets to 4096 the number of processes an MPI process may be connected to at any one time. It includes processes in the same MPI job and processes in jobs that are currently connected to the MPI process. This equals the value to which the environment variable `MPI_MAXPROCS` is set.
		integer	To change the maximum number of processes an MPI process may be connected to at any one time, enter an integer specifying the desired limit. The value may not exceed the setting for the option `maxprocs_limit`.

TABLE 7-4 MPI Runtime Options *(Continued)*

Option	Values		Description
	Default	Other	
`maxprocs_limit`		*integer*	The maximum process table size a user may set `MPI_MAXPROCS` to. If the option `maxprocs_default` is not set, the user is able to specify a value up to `MAX_INT`.
`spin`	`off`		Sets the MPI library spin policy to spin nonaggressively. This equals the value to which the environment variable `MPI_SPIN` is set.
		`on`	Sets the MPI library to spin aggressively.

Setting MPI Spin Policy

An MPI process often has to wait for a particular event, such as the arrival of data from another process. If the process checks (*spins*) for this event continuously, it consumes CPU resources that may be deployed more productively for other purposes.

The administrator can direct that the MPI process instead register events associated with shared memory or remote shared memory (RSM) message passing with the spin daemon `spind`, which can spin on behalf of multiple MPI processes (*coscheduling*). This frees up multiple CPUs for useful computation. The `spind` daemon itself runs at a lower priority and backs off its activities with time if no progress is detected.

The `SUNWrte` package implements the `spind` daemon, which is not directly user callable.

The cluster administrator can control spin policy in the `hpc.conf` file. The attribute `coscheduling`, in the `MPIOptions` section, can be set to `avail`, `on`, or `off`.

- `avail` (the default) means that spin policy is determined by the setting of the environment variable `MPI_COSCHED`. If `MPI_COSCHED` is set to zero or is not set, `spind` is not used. If `MPI_COSCHED` is set to one, `spind` must be used.
- `on` means that `spind` must be used by MPI processes that wish to block on shared-memory communication. This value overrides `MPI_COSCHED=0`.
- off means that `spind` cannot be used by MPI processes. This value overrides `MPI_COSCHED=1`.

The cluster administrator can also change the setting of the attribute `spindtimeout`, indicating how long a process waits for `spind` to return. The default is 1000 milliseconds.

For tips on determining spin policy, see the man page for `MPI_COSCHED`. In general, the administrator may wish to force use of `spind` for heavily used development partitions where performance is not a priority. On other partitions, the policy could be set to `avail`, and users can set `MPI_COSCHED=0` for runs where performance is needed.

`CREOptions` Section

The CREOptions section controls the behavior of CRE in logging system events, handling daemon core files, and authenticating users and programs.

The template `hpc.conf` file contains the default settings for these behaviors for the current cluster. These settings are shown in TABLE 7-5.

TABLE 7-5 `CREOptions` Section Example

```
Begin CREOptions
enable_core       off
corefile_name     core
syslog_facility   daemon
auth_opt          none
End CREOptions
```

Specifying the Cluster

The cluster is specified by appending the tag Server=*master-node-name* to the `Begin CREOptions` line:

`Begin CREOptions Server=`*master-node-name*

If the node name supplied does not match the name of the current master node, then this section is ignored.

It is possible to have two `CREOptions` sections. The section without a tag is always processed first. Then the section with a matching *master-node-name* adds to or overrides the previous settings.

Logging System Events

By default, CRE uses the `syslog` facility to log system events, as indicated by the entry `syslog_facility daemon`. Other possible values are `user`, `local0`, `local1`, `...`, `local7`. See the man pages for `syslog(2)`, `syslogd(8)`, and `syslog.conf(5)` for information on the `syslog` facility.

Note – In rare cases, the CRE daemons may log errors to the default system log. This occurs when an error is generated before the system has read the value of `syslog_facility` in `hpc.conf`.

Enabling Core Files

By default, core files are disabled for CRE daemons. The administrator may enable core files by changing `enable_core off` to `enable_core on`. The administrator may also specify where daemon core files are saved by supplying a value for `corefile_name`. See `coreadm(1M)` for the possible naming patterns. For example:

```
corefile_name /var/hpc/core.%n.%f.%p
```

This would cause any core files to be placed in `/var/hpc` with the name `core` modified by node name (`%n`), executable file name (`%f`), and process ID (`%p`). (Note that only daemon core files are affected by the `CREOptions` section; user programs are not affected.)

Enabling Authentication

Authentication may be enabled by changing the `auth_opt` value from `none` (the default) to either `des` or `krb5`. These values indicate DES or Kerberos Version 5, respectively. See "Authentication and Security" on page 37 for additional steps needed to establish the chosen authentication method.

`PFSFileSystem` Section

The `PFSFileSystem` section describes the parallel file systems that Sun MPI applications can use. This description includes

- The name of the parallel file system

- The host name of each server node in the parallel file system
- The name of the storage device attached to each server node
- One or more options modifying each server node's setup

A separate `PFSFileSystem` section is needed for each parallel file system that you want to create. TABLE 7-6 shows a sample `PFSFileSystem` section with two parallel file systems, `pfs0` and `pfs1`.

TABLE 7-6 `PFSFileSystem` Section Example

```
Begin PFSFileSystem=pfs0
#NODE            DEVICE              OPTIONS
node0           /dev/rdsk/c0t1d0s2
node1           /dev/rdsk/c0t1d0s2
node2           /dev/rdsk/c0t1d0s2
End PFSFileSystem

Begin PFSFileSystem=pfs1
#NODE            DEVICE              OPTIONS
node2           /dev/rdsk/c0t2d0s2
node3           /dev/rdsk/c0t2d0s2
End PFSFileSystem
```

Parallel File System Name

The first line shows the name of the parallel file system. PFS file system names must not include spaces.

Server Node Host Names

The `NODE` column lists the host names of the nodes that function as I/O servers for the parallel file system being defined. The example configuration in TABLE 7-6 shows two parallel file systems:

- `pfs0` – Three server nodes: `node0`, `node1`, and `node2`.
- `pfs1` – Two server nodes: `node2` and `node3`.

Note that I/O server `node2` is used by both `pfs0` and `pfs1`. Note also that host name `node3` represents a node that is used as both a PFS I/O server and as a computation server—that is, it is also used for executing application code.

Storage Device Names

The `DEVICE` column gives the device name associated with each member node. This name follows Solaris device naming conventions.

Options

For each node, the administrator may specify a comma-separated list of options.

TABLE 7-7 `PFSFileSystem` Options

Option	Description
`threads=`	The number of client threads to be spawned in the PFS I/O daemon to manage that disk. Default is 1.
`bufcnt=`	The number of data buffers to be created for that disk. Default is 4.
`bufsize=`	The size of the data buffers for that disk. The value is rounded down, if necessary, to the nearest multiple of 32Kb (PFS's basic block size). Default is 1Mb.
`align=`	The optimal alignment for disk accesses (designed for RAIDs). The value is rounded down, if necessary, to the nearest multiple of 32KB (PFS's basic block size). Default is 32K. .

For example:

```
Begin PFSFileSystem=pfs1
# NODE     DEVICE              OPTIONS
hpc-io0    /dev/rdsk/c0t1d0s2  threads=1
hpc-io1    /dev/rdsk/c0t1d0s2  threads=1,bufcnt=8,bufsize=1MB
End PFSFileSystem
```

`PFSServers` Section

A PFS I/O server is a Sun HPC node that is:

- Connected to one or more disk storage units that are listed in a `PFSFileSystem` section of the `hpc.conf` file
- Listed in the `PFSServers` section of `hpc.conf`, as shown in TABLE 7-8
- Has a PFS I/O daemon running on it.

TABLE 7-8 `PFSServers` Section Example

```
Begin PFSServers
#NODE            OPTIONS
node0
node1
node2
node3
End PFSServers
```

Nodes

The left column lists the host names of the nodes that are PFS I/O servers. In this example, they are `ios0` through `node2` and `node3`.

Options

For each node, the administrator may specify a comma-separated list of options. The options are shown in TABLE 7-9.

TABLE 7-9 `PFSServer` Options

Option	Description
`threads=`	The number of threads to be spawned to handle client requests. Default is 5.
`nbufs=`	The amount of memory the PFS I/O daemon will have for buffering transfer data, specified in units of 32-Kbyte buffers. Default is 64.
`clntcong={yes | no}`	Turn on (or turn off) congestion control on the client side. This option and the next cause PFS to regulate carefully the load it places on the network and on particular interfaces. They are intended to be used with networks that do not degrade gracefully under heavy load. In most cases, there is a performance penalty for setting this option to yes. Default is no.
`servcong={yes | no}`	Turn on (or turn off) congestion control on the server side. This option and the previous cause PFS to regulate carefully the load it places on the network and on particular interfaces. They are intended to be used with networks that do not degrade grace fully under heavy load. In most cases, there is a performance penalty for setting this option to yes. Default is no.

For example:

```
Begin PFSServers
# NODE         OPTIONS
hpc-io0      nbufs=32
hpc-io1      nbufs=32
End PFSServers
```

HPCNodes Section

This section is used only in a cluster that is using LSF as its workload manager, not CRE. CRE ignores the `HPCNodes` section of the `hpc.conf` file.

PMODULES Section

The `PMODULES` section provides the names and locations of the protocol modules (PMs) which the run-time system is to discover and make available for use for communication in the cluster.

When a CRE-based cluster is being started, an instance of the daemon `tm.omd` is started on each node. This daemon is responsible for discovering various information about a node, including which PMs are available for that node. The `tm.omd` daemon looks in the `hpc.conf` file for a list of PMs that may be available. It then opens the PMs, and calls an interface discovery function to find out if the PM has interfaces that are up and running. This information is returned to the `tm.omd` and stored away in the cluster database.

The `PMODULES` section of the `hpc.conf` file lists each PM by name and gives the location (or default location) where it may be found. The template `hpc.conf` looks like this:

```
# PMODULE LIBRARY
Begin PMODULES
shm      ()
rsm      ()
tcp      ()
End PMODULES
```

Three PMs are shipped with Sun HPC ClusterTools and included in the template `hpc.conf` file. These are:

- `shm` PM, used for on-node communication
- `tcp` PM, used (typically) for internode communication on TCP-IP-compatible interconnects
- `rsm` PM, used (typically) for internode communication on a supported interconnect

The template `hpc.conf` file specifies the location of all three PMs as `()`, which indicates the default location. The default location is `/opt/SUNW/hpc/lib` for 32-bit daemons, and `/opt/SUNW/hpc/lib/sparcv9` for 64-bit daemons.

The administrator has the option of putting PM libraries in a location other than the default. This is useful, for instance, when a new user-defined PM is being developed. For PMs located in a directory other than the default, the administrator must put the absolute pathname in the `hpc.conf` file. For example:

```
# PMODULE LIBRARY
Begin PMODULES
tcp      ()
shm      /home/jbuffett/libs
End PMODULES
```

In this example, the `tcp` libraries are located in the default location and the `shm` libraries are located in `/home/jbuffett/libs`. In a 64-bit environment, `sparcv9` is automatically added to the pathname. Thus, this `hpc.conf` entry indicates that the `shm` PM libraries would be found in `/home/jbuffett/libs/sparcv9`.

PM Section

The `hpc.conf` file contains a PM section for each available protocol module. The section gives standard information (name of interface and its preference ranking) for the PM, along with additional information for some types of PMs.

The name of the PM being described appears on the same line as the keyword `PM` with an equal sign and no spaces between them. This example shows the `PM` sections provided for the `shm` and `rsm` PMs.

```
# SHM settings
# NAME  RANK
Begin PM=shm
shm     5
End PM
# RSM settings
# NAME  RANK  AVAIL
Begin PM=rsm
wrsm    20    1
End PM
```

The NAME and RANK columns must be filled in for all PMs. The shm PM requires only these two standard items of information; the rsm PM has an additional field called AVAIL.

NAME Column

The name of the interface indicates the controller type and, optionally, a numbered interface instance. Interface names not ending with a number are wildcards; they specify default settings for all interfaces of that type. The name can be can be between 1 and 32 characters in length.

If interfaces are specified by name after a wildcard entry, the named entries take precedence.

RANK Column

The rank of an interface is the order in which that interface is preferred over other interfaces, with the lowest-ranked interface the most preferred. That is, if an interface with a rank of 0 is available when a communication operation begins, it will be selected for the operation before interfaces with ranks of 1 or greater. Likewise, an available rank 1 interface will be used before interfaces with a rank of 2 or greater.

Note – Because hpc.conf is a shared, cluster-wide configuration file, the rank specified for a given interface will apply to all nodes in the cluster.

Network ranking decisions are usually influenced by site-specific conditions and requirements. Although interfaces connected to the fastest network in a cluster are often given preferential ranking, raw network bandwidth is only one consideration. For example, an administrator might decide to dedicate one network that offers very low latency, but not the fastest bandwidth, to all communication within a cluster and use a higher-capacity network for connecting the cluster to other systems.

Rank can also be specified for interface instances within a PM section. For example, consider a customized hpc.conf entry like this:

```
# RSM Settings
# NAME     RANK  AVAIL
Begin PM=rsm
wrsm       15    1
wrsm0      10    1
wrsm1      20    1
wrsm2      30    1
wrsm3      40    1
End PM=rsm
```

If controllers wrsm0 and wrsm1 could be used to establish connections to the same process, wrsm0 would always be chosen, since it has the lower ranking number.

Note – If multiple interfaces have the same ranking, the rsm PM will use the MPI options rsm_links and rsm_maxstripe to reduce the number of interfaces and use the remaining interface(s) for the connection. If more than one interface is resolved for making the connection, the rsm PM will stripe the connection, using all interfaces resolved.

AVAIL Column

The rsm PM section contains an additional column headed AVAIL. This value indicates whether a controller is (1) or is not (2) available for RSM communication.

Configuring Out Controllers

The AVAIL column can be used to configure out one or more controllers, perhaps for maintenance on the network. If all controllers on the network are going to be unavailable, the administrator can change the availability of the wildcard entry to 0. If only certain controllers will be unavailable, the administrator can leave the

wildcard entry set to 1 but add entries set to 0 availability for named instances. (Remember to stop the CRE daemons and the RSM daemon `hpc_rsmd` when editing the `hpc.conf` file and restart them afterward.)

```
# RSM Settings
# NAME     RANK  AVAIL
Begin PM=rsm
wrsm       15    1
wrsm2      20    0
wrsm3      15    0
End PM=rsm
```

Note – Alternatively, the administrator can use the option `rsm_links` in the `MPIOptions` section to configure out one or more controllers. See "Configuring Out Network Controllers" on page 139.

Enabling Software Striping

Another use for the `AVAIL` column is to enable software-controlled striping of messages over network controllers. When a message is submitted for transmission over the network, the `rsm` PM distributes the message over as many network interfaces as are available with the same preference ranking, up to the limit of 8 links.

Note – Software-controlled message striping is most useful for interconnect technology that does not support hardware-controlled striping. The hardware striping performed by some interconnects is generally preferable to the software-controlled striping described here.

In striped communication, a message is split into smaller packets and transmitted in two or more parallel streams over a set of network controllers that have been logically combined into a *stripe-group*.

The `AVAIL` column allows the administrator to include individual network interfaces in a (software) *stripe-group pool*. Members of this pool are available to be included in logical stripe groups as long as they have the same preference ranking. These stripe groups are formed on an as-needed basis, selecting interfaces from this stripe-group pool.

To include an interface in a stripe-group pool, set its `AVAIL` value to `1`. To exclude an interface from the pool, specify `0`.

Stripe-group membership is optional so you can reserve some network bandwidth for non-striped use (assuming the network has another PM enabled). To do so, simply set `AVAIL` to 0 on the network interface(s) you wish to reserve in this way.

TCP-IP `PM` Section

The `PM` section provided for the `tcp` PM in the template `hpc.conf` file contains the standard `NAME` and `RANK` columns, along with several placeholder columns that are not used at this time. The default TCP settings (and placeholders) are shown in TABLE 7-10.

TABLE 7-10 `PM=tcp` Section Example

```
# TCP settings
# NAME  RANK    MTU     STRIPE  LATENCY BANDWIDTH
Begin PM=tcp
midn     0     16384  0       20      150
idn1     0     16384  0       20      150
wrsmd   25     32768  0       20      150
mscid   30     32768  0       20      150
scid    40     32768  0       20      150
mba     50     8192   0       20      150
ba      60     8192   0       20      150
mfa     70     8192   0       20      150
fa      80     8192   0       20      150
macip   90     8192   0       20      150
acip    100    8192   0       20      150
manfc   110    16384  0       20      150
anfc    120    16384  0       20      150
mbf     130    4096   0       20      150
bf      140    4096   0       20      150
mbe     150    4096   0       20      150
be      160    4096   0       20      150
mqfe    163    4096   0       20      150
qfe     167    4096   0       20      150
mhme    170    4096   0       20      150
hme     180    4096   0       20      150
mle     190    4096   0       20      150
le      200    4096   0       20      150
msmc    210    4096   0       20      150
smc     220    4096   0       20      150
lo      230    4096   0       20      150
End PM
```

The template `hpc.conf` file identifies the network interfaces that are included in the TCP `PM` section. The networks with the prefix "`m`" are for Enterprise 10000 alternate pathing support, and should be used in preference to the underlying interface (thus their lower ranking).

Note – Inclusion of any network interface in this file does not imply that Sun Microsystems supports, or intends to support, that network.

Propagating `hpc.conf` Information

Whenever `hpc.conf` is changed, the CRE database must be updated with the new information. After all required changes to `hpc.conf` have been made, restart the CRE master and nodal daemons as follows:

```
# /etc/init.d/sunhpc.cre_master start
# /etc/init.d/sunhpc.cre_node start
```

Remember to start the CRE master daemons before the nodal daemons.

Also, restart the RSM daemon if used:

```
# /etc/init.d/sunhpc.hpc_rsmd start
```

If PFS file systems were unmounted and PFS I/O daemons were stopped, restart the I/O daemons and remount the PFS file systems. See Chapter 4 for guidance.

CHAPTER 8

Maintenance and Troubleshooting

This chapter describes some procedures you can use for preventive maintenance and troubleshooting. The topics covered are:

Cleaning Up Defunct CRE Jobs

One preventive maintenance practice that can be beneficial is the routine cleanup of defunct jobs. There are several types of such jobs:

- Jobs that have exited, but still appear in `mpps` output
- Jobs that have not terminated, but need to be removed
- Jobs that have orphan processes

Removing CRE Jobs That Have Exited

When a job does not exit cleanly, it is possible for all of a job's processes to have reached a final state, but the job object itself to not be removed from the CRE database. The following are two indicators of such incompletely exited jobs:

- A process (identified by `mpps`) in the `EXIT`, `SEXIT`, `FAIL`, or `CORE` state

- A Prism main window that will not close or exit

If you see a job in one of these defunct states, perform the following steps to clear the job from the CRE database:

1. **Execute `mpps -e` again in case CRE has had time to update the database (and remove the job).**
2. **If the job is still running, kill it, specifying its job ID.**

```
% mpkill jid
```

If `mpps` continues to report the killed job, use the `-C` option to `mpkill` to remove the job object from the CRE database. This must be done as superusee from the master node.

```
# mpkill -C jid
```

Removing CRE Jobs That Have Not Terminated

The second type of defunct job includes jobs that are waiting for signals from processes on nodes that have gone off line. The `mpps` utility displays such jobs in states such as `RUNNING`, `EXITING`, `SEXTNG`, or `CORNG`.

Note – If the job-killing option of `tm.watchd` (`–Yk`) is enabled, CRE handles such situations automatically. This section assumes this option is not enabled.

Kill the job using:

```
% mpkill jid
```

There are several variants of the `mpkill` command, similar to the variants of the Solaris `kill` command. You may also use:

```
% mpkill -9 jid
```

or

```
% mpkill -I jid
```

If these do not succeed, execute `mpps -pe` to display the unresponsive processes. Then, execute the Solaris `ps` command on each of the nodes listed. If those processes still exist on any of the nodes, you can remove them using `kill –9` *pid*.

Once you have eliminated defunct jobs, data about the jobs may remain in the CRE database. As superuser from the master node, use `mpkill –C` to remove this residual data.

Killing Orphaned Processes

When the `tm.watchd –Yk` option has been enabled, the watch daemon marks processes `ORPHAN` if they run on nodes that have gone off line. If the node resumes communication with the CRE daemons, the watch daemon will kill the `ORPHAN` processes. If not, you will have to kill the processes manually using the Solaris `kill` command. Otherwise, such processes will continue to consume resources.

Symptoms of orphaned processes can be detected by examining error log files or `stdout`, if you are running from a terminal. You can also search for such errors as `RPC: cannot connect`, or `RPC: timout`. These errors will appear under `user.err` priority in `syslog`.

Note – If an `mprun` process becomes unresponsive on a system, even where `tm.watchd –Yk` has been enabled, it may be necessary to use `Ctrl-c` to kill `mprun`.

Cleaning Up After RSM Failures

The daemon `hpc_rsmd`, which is started when the cluster is booted, manages access to remote shared memory services on behalf of MPI processes. If an instance of `hpc_rsmd` exits abnormally, you can use the following script to clean up any files and System V shared memory segments that it leaves behind.

```
# /etc/init.d/sunhpc.hpc_rsmd  [ start | stop | clean ]
```

Using Diagnostics

The following sections describe Solaris diagnostics that may be useful in troubleshooting various types of error conditions.

Using Network Diagnostics

You can use `/usr/sbin/ping` to check whether you can connect to the network interface on another node. For example:

```
% ping hpc-node3
```

tests (over the default network) the connection to `hpc-node3`.

You can use `/usr/sbin/spray` to determine whether a node can handle significant network traffic. `spray` indicates the amount of dropped traffic. For example:

```
% spray –c 100 hpc-node3
```

sends 100 small packets to `hpc-node3`.

Checking Load Averages

You can use `mpinfo –N` or, if CRE is not running, `/usr/bin/uptime`, to determine load averages. These averages can help to determine the current load on the machine and how quickly it reached that load level.

Using Interval Diagnostics

The diagnostic programs described below check the status of various parameters. Each accepts a numerical option that specifies the time interval between status checks. If the interval option is not used, the diagnostics output an average value for the respective parameter since boot time. Specify the numerical value at the end of the command to get current information.

Use `/usr/bin/netstat` to check local system network traffic. For example:

```
% netstat –ni 3
```

checks and reports traffic every three seconds.

Use `/usr/bin/iostat` to display disk and system usage. For example:

```
% iostat -c 2
```

displays percentage utilizations every two seconds.

Use `/usr/bin/vmstat` to generate additional information about the virtual memory system. For example:

```
% vmstat -S 5
```

reports on swapping activity every five seconds.

It can be useful to run these diagnostics periodically, monitoring their output for multiple intervals.

Interpreting CRE Error Messages

This section presents sample error messages and their interpretations.

- The following error message usually indicates that all the nodes in a CRE partition are marked down—that is, their node daemons are not running:

  ```
  No nodes in partition satisfy RRS:
  ```

- Under certain circumstances, when a user attempts to kill a job, CRE may log error messages of the following form on the master node:

  ```
  Aug 27 11:02:30 ops2a tm.rdb[462]: Cond_set: unable to
  connect to ops2a/45126: connect: Connection refused
  ```

If these errors can be correlated to jobs being killed, then they can be safely ignored. One way to check this correlation would be to look at the accounting logs for jobs that were signaled during this time.

- The following error message indicates that no partitions have been set up:

  ```
  mprun: unique partition: No such object
  ```

- When there is stale job information in the CRE database, an error message of the following form may occur:

  ```
  Query returned excess results:
  a.out: (TMTL UL) TMRTE_Abort: Not yet initialized
  The attempt to kill your program failed
  ```

  This might happen, for example, when `mpps` shows running processes that are actually no longer running.

  Use the `mpkill -C` *nn* command to clear out such stale jobs.

Note – Before removing the job's information from the database, the `mpkill -C` option verifies that the processes of the job are in fact no longer running.

Anticipating Common Problems

This section presents some guidelines for preventing and troubleshooting common problems.

- When running multiprocess jobs (`-np` equal to 0 or greater than 1) in a cluster with NFS-mounted file systems, you should take steps to limit core dumps to zero. This can be done with the Solaris `limit` command. See the `limit(1)` man page for additional information.
- The CRE resource database daemon (`tm.rdb`) does not remove missing interfaces after a client daemon is restarted. Instead, the `mpinfo -Nv` command will show them marked as `down`.
- The contents of the `/var/adm/messages` file are local to each node. Any daemon messages are logged only on the node where that daemon runs. By default, CRE daemon messages are stored in `/var/adm/messages` along with other messages handled by `syslog`. Alternatively, CRE messages can be written to a file specified by the `mpadmin logfile` command.
- Use shell I/O redirection instead of `mprun` `-I` options whenever possible. Using shell redirection reduces the likelihood of problems involving standard I/O.
- If `mprun` is signaled too soon after it has been invoked, it exits without stopping the job's processes. If this happens, use `mpkill -9` *jid* to kill such a job.
- CRE does not pass supplemental group ID information to remote processes. You must use the `-G` *gid* option with `mprun` to run with the group permissions of that group. You must be a member of that group.
- CRE RPC timeouts in Sun MPI code are logged to `syslog`, but the default `syslog.conf` file causes these messages to be dropped. If you want to see these errors, modify your `/etc/syslog.conf` file so that messages of the priority `user.err` are not dropped. Note that this does not apply to RPC timeouts occurring in the CRE daemons themselves. By default, these are logged to `/var/adm/messages`.

Note – If you have set the Cluster-level attribute `logfile`, all error messages generated by user code will be handled by CRE (not `syslog`) and will be logged in a file specified by an argument to `logfile`.

CRE RPC timeouts in user code are generally not recoverable. The job might continue to run, but processes probably will not be able to communicate with each other. There are two ways to deal with this:

- Enable the `tm.watchd` job killing option (`–Yk`), which will automatically kill jobs when nodes go off line. This will catch most of these cases, since RPC timeouts usually coincide with `tm.watchd` marking the node as off line.
- Monitor RPC errors from user codes by looking for `syslog` messages of priority `user.err`. Then use `mpkill` to kill the associated job manually.
- If a file system is not visible on all nodes, users can encounter a `permission denied` message when attempting to execute programs from such a file system. Watch for errors caused by non-shared file systems like `/tmp`, which exist locally on all nodes. This can show up when users attempt to execute programs from `/tmp`, and the program does not exist in the `/tmp` file systems of all nodes.
- If you execute `mpkill` with the `–C` option (this option is available only to the system administrator), you should look for and remove leftover files on the master node. The file names for large files are of the form:

  `/tmp/.hpcshm_mmap.`*jid*`.*`

  Smaller files will have file names of the form:

  `/tmp/.hpcshm_acf.`*jid*`.*`

The Sun MPI shared memory protocol module uses these files for interprocess communication on the same node. These files consume swap space.

Understanding Protocol-Related Errors

Errors may occur at cluster startup or at program initialization because of problems finding or loading protocol modules. Such errors are not fatal to the runtime environment (that is, to the CRE daemons), but they do mean that the protocol in question is not available for communication on the cluster.

This section describes some error conditions that may occur in relation to protocol modules(PMs) and the RSM daemon `hpc_rsmd`.

Errors When CRE Daemons Load Protocol Modules

The errors below are generated when the CRE daemons first start up. These errors can occur because of problems in the hpc.conf file, or because of problems loading the PMs.

All these errors are considered nonfatal to the daemon, but the PM that causes the error will not be usable. The errors below cause the CRE daemons to generate calls to `syslog` that result in self-explanatory error messages.

- Error: PM listed in `hpc.conf` cannot be found.
- Error: Library that PM depends on could not be found.
- Error: PM name has more than 4 characters.
- Error: Attempting to load 32-bit PM into 64-bit daemon.

The daemons cause a warning to be generated when there are duplicate PM entries in the `PMODULES` section of `hpc.conf`. If there are multiple PM entries with the same name, then only the first one is loaded.

Errors When Protocol Modules Discover Interfaces

The errors below are generated at program startup when a PM attempts interface discovery.

These errors are nonfatal to the CRE daemons, but they may mean that the PM causing the error will not be usable. Appropriate error strings are generated by `syslog`.

- Error: Malformed interface entry in the `hpc.conf` file. This example indicates a missing `RANK` column entry for the `hme` interface for the `tcp` PM.

```
-WARNING- Problem detected initializing tcp PM: PM=tcp entry hme is
missing tokens
```

- Error: Missing interface entry in the `hpc.conf` file. If an entry is missing entirely, then default values are used if available. This example indicates that there is no entry for the `hme` interface for the `tcp` PM.

```
-WARNING- Problem detected initializing tcp PM: Interface hme0 has
missing or broken entry in hpc.conf. Will use: Rank=1000,stripe=0,
mtu=1500 latency=-1, bandwidth=-1
```

- Error: Malformed interface in the `hpc.conf` file. This example indicates that there are too many columns for the `hme` interface for the `tcp` PM.

```
-WARNING- Problem detected initializing tcp PM: PM=tcp entry hme has
extra tokens
```

Errors When the RSM Protocol Module Reads MPI Options

During the startup of an MPI job, the RSM PM interprets values of options in the `MPIOptions` section of `hpc.conf`.

If the `MPIOptions` section does not exist, the RSM PM uses the default values. The PM ignores any option names it does not recognize. If an option is given a value that is either malformed (for instance, a character instead of a number) or out of bounds, the PM uses a default value and causes a message like the following to be generated:

```
-WARNING- ignoring rsm_shortmsgsize=10 in hpc.conf, using default
value of 384 bytes. Please contact system administrator.
```

The `rsm_links` option has other error messages associated with it, indicating an invalid instance number for an interface type or an invalid interface name. In both cases, the invalid item is ignored. The message generated is similar to the following:

```
-WARNING- ignoring rsm_links entry "hpc-comm0wrsm0,1" in hpc.conf...
```

Action of the RSM Daemon

The RSM daemon `hpc_rsmd` writes messages to `syslog` upon detection of unusual events. It writes three different levels of `syslog` messages: error, warning, and information.

Error events that result in the termination of a running MPI job are recorded in the `syslog` at error level. Error events that do not result in the termination or a running MPI job are recorded in the `syslog` at warning level. All other `syslog` messages are written at information level. The `syslogd` can be configured manually to enable or limit the output of these levels.

During `hpc_rsmd` initialization, configuration information is written to `syslog` at information level. An error causes a messages to be written at the `syslog` error level, and the `hpc_rsmd` exits.

After initialization, the `hpc_rsmd` will never spontaneously exit. If the `hpc_rsmd` detects a recoverable error in a request while it is being sent, it will retransmit the request. If the `hpc_rsmd` detects a non-recoverable error while a request is being sent, it will abort the request and return an error to the MPI process that made the

request. This, in turn, will cause the MPI job to abort. However, the `hpc_rsmd` path monitor will periodically ping all paths it believes are connected and, upon failure of a ping, will mark the path not available for use.

The RSM PM does not detect error signals itself. In the event that the RSM PM is initialized when the RSM daemon `hpc_rsmd` is not running, the RSM PM prints out an appropriate error message and abort the MPI job.

Recovering From System Failure

Recovering from system failure involves rebooting CRE and recreating the CRE resource database.

The `cre.master reboot` and `cre.node reboot` commands should be used only as a last resort, if the system is not responding (for example, if programs such as `mprun`, `mpinfo`, or `mpps` hang). Follow these steps to reboot CRE:

1. Run `cre.master reboot` on the master node:

```
# /etc/init.d/cre.master reboot
```

2. Run `cre.node reboot` on all the nodes (including the master node if it is running `tm.spmd` and `tm.omd`):

```
# /etc/init.d/cre.node reboot
```

The procedure attempts to save the system configuration (in the same way as using the `mpadmin dump` command), kill all the running jobs, and restore the system configuration. Note that the Cluster Console Manager applications may be useful in executing commands on all the nodes in the cluster simultaneously. For information about the Cluster Console Manager applications, see Appendix A.

Note – `rte.master reboot` saves the existing `rdb-log` and `rdb-save` files in `/var/hpc/rdb-log.1` and `/var/hpc/rdb-save.1`. The `rdb-log` file is a running log of the resource database activity and `rdb-save` is a snapshot of the database taken at regular intervals.

To recover CRE after a partial failure, that is, when some but not all daemons have failed, it is possible to clean up bad database entries without losing the configuration information. You can flush the dynamic data while preserving the configuration by executing these commands:

```
# /etc/init.d/sunhpc.cre_master start
# /etc/init.d/sunhpc.cre_master reboot
```

(The `start` command is necessary in case some of the daemons are not running.)

Configuring Out Network Controllers

During maintenance or replacement of Sun Fire™ series high-performance cluster interconnect hardware (when available), the cluster administrator may wish to configure out the controller(s) that have become unavailable. This can be done in the `hpc.conf` file in either of two ways: by changing the `PM=rsm` section or by changing the `rsm_links` option in the `MPIOptions` section. This section describes both methods.

Using the `PM` Section

Stop the CRE and RSM daemons when editing a `PM` section of `hpc.conf` and restart them when finished. See "Stopping and Restarting CRE" on page 14 and "RSM Daemon" on page 19.

If all controllers on the same network are going to be unavailable, the administrator can add a new line to the `PM=rsm` section that names the controller instance and sets its `AVAIL` field to 0.

```
# RSM Settings
# NAME     RANK  AVAIL
Begin PM=rsm
wrsm       15    1
wrsm0      15    0
End PM=rsm
```

Once the daemons are restarted, controller 0 will be unavailable on all nodes on the cluster. The wildcard entry `wrsm` is still set to 1, indicating that all controllers on that network other than controller 0 will be available.

Using the MPIOptions Section

By editing the rsm_links option in the MPIOptions section, the administrator can either remove a controller instance across all nodes or remove a controller instance from one or more specified nodes.

Using the MPIOptions method does not require you to stop and restart the daemons. However, any MPI jobs that are currently running are not notified to not use the controller(s) that are being configured out, and this may cause a job to abort.

To remove an instance across all nodes, set the rsm_links option to exclude that instance. For example, suppose that a cluster has controllers wrsm0 and wrsml and you wish to configure out wrsm0. To do so, set the value of rsm_links to wrsm1 only.

```
rsm_links wrsm1
```

To remove a controller instance only on specified nodes, use the syntax *node.controller* to indicate which controllers on the node(s) in question should remain available. For example, consider a three-node cluster with controllers wrsm0 and wrsm1 on all three nodes. The following entry has the effect of removing controller wrsm0 from node1:

```
rsm_links wrsm1 node0.wrsm0 node2.wrsm0
```

This entry specifies that controller wrsm1 is available on all nodes, while controller wrsm0 is available only on nodes node0 and node2.

APPENDIX **A**

Cluster Console Manager Tools

This appendix describes a set of cluster administration tools that are installed with the Sun HPC ClusterTools release.This toolset, called the Cluster Console Manager, allows you to issue commands to all nodes in a cluster simultaneously through a graphical user interface. The CCM offers three modes of operation:

- `cconsole` – This interface provides access to each node's console port through terminal concentrator links. To use this tool, the cluster nodes must be connected to terminal concentrator ports and those node/port connections must be defined in the `hpc_config` file. See the *Sun HPC ClusterTools Installation Guide* for details.
- `ctelnet` – This interface initiates simultaneous `telnet` sessions over the network to all nodes in the cluster. Note that if passwords are required, every node must be able to accept the same password.
- `crlogin` – This interface uses `rlogin` to log you in to every node in the cluster. Note that if you launch `crlogin` while logged in as superuser, all `rlogin` sessions are done as superuser. Likewise, if `crlogin` is launched from an ordinary user prompt, all remote logins are done as user.

Each of these modes creates a command entry window, called the *Common Window*, and a separate console window, called a *Term Window*, for each node. Each command typed in the Common Window is echoed in all Term Windows (but not in the Common Window). Every Term Window displays commands you issue as well as system messages logged by its node.

Note – If the cluster nodes are not connected to a terminal concentrator, only `ctelnet` and `crlogin` can be used, not `cconsole`.

Launching Cluster Console Tools

All Cluster Console tools are launched using the same command-line form:

% *tool_name cluster_name*

where *tool_name* is `cconsole`, `ctelnet`, or `crlogin`, and *cluster_name* is a name given to the cluster. The default cluster name is `hpc_cluster`, which is established automatically when `cluster_tool_setup` is executed. For example,

- Launch `ctelnet` by entering:

```
% ctelnet hpc_cluster
```

- Launch `crlogin` by entering:

```
% crlogin hpc_cluster
```

- Launch `cconsole` by entering:

```
% cconsole hpc_cluster
```

If you want to use `cconsole` to monitor messages generated while rebooting the cluster nodes, you will need to launch it from a machine outside the cluster. If you launch it from a cluster node, it will be disabled when the node from which it is launched reboots.

Note – Because `cconsole` accesses the console ports of every node in the cluster, no other accesses to any console in the cluster will be successful while the `cconsole` session is active.

All three Cluster Console commands take the standard X/Motif command-line arguments.

Common Window

The Common Window is the primary window used by the system administrator to send input to all the nodes. This window has a menu bar with three menus and a text field for command entry. The Common Window is always displayed when the Cluster Console is launched.

The Common Window menu bar has three menus:

- Hosts
- Options
- Help

In this manual, the Cluster Console term *Hosts* refers to Sun HPC cluster nodes.

Hosts Menu

The Hosts menu displays a list of the nodes contained in the cluster, plus two other entries, Select Hosts and Exit. TABLE A-1 describes these menu choices.

TABLE A-1 Cluster Console Menu Entries

Entry	Function
Host toggle buttons	Selects whether or not the host gets input from the Common Window text field. There is a separate toggle button for each node currently connected to the Cluster Console. ON — Enables input from the Common Window text field to the node. OFF — Disables input from the Cluster Console.
Select Hosts	Displays the Select Hosts dialog window.
Exit	Quits the Cluster Console program.

Select Hosts Dialog

The Select Hosts dialog enables you to add or delete nodes during the current Cluster Console session. The scrolled text window in the Select Hosts dialog displays a list of the nodes that are currently connected to the Cluster Console.

There are three Select Hosts dialog buttons, which are described in TABLE A-2.

TABLE A-2 Select Hosts Dialog Buttons

Entry	Function
Insert	Opens a Term Window and establishes a connection to the specified hosts(s). Adds the host(s) specified in the Hostname text field to the list of accessible hosts. The inserted host name(s) are displayed in the hosts list in the scrolled text window and in the Common Window.
Remove	Deletes the host selected in the Hosts list in the scrolled text window.
Dismiss	Closes the Select Hosts dialog.

.

▼ Adding a Single Node

1. **Enter the *hostname* in the Hostname text field.**
2. **Select Insert.**

   Entering a valid host name opens a Term Window for the specified host and establishes a connection to that host. The name of the selected host appears in the scrolled text window and in the hosts list on the Hosts menu in the Common Window.

▼ Adding All Nodes in a Cluster

1. **Enter the *clustername* in the Hostname text field.**
2. **Select Insert.**

   The Cluster Console automatically expands the cluster name into its constituent host names and then opens one Term Window for each node. A connection is established for each of the constituent host names. The Cluster Console automatically displays the names of the hosts in the cluster in the scrolled text window and in the hosts list on the Hosts menu in the Common Window.

▼ Removing a Node

1. **Select the name of the host in the list in the scrolled text window.**
2. **Select Remove.**

   This closes the corresponding Term Window and disconnects the host. The name of the removed host disappears from the scrolled text window and from the hosts list on the Hosts menu in the Common Window.

Options Menu

The Options menu has one entry, Group Term Window; see TABLE A-3 for a description.

TABLE A-3 Group Term Window Entry

Entry	Function
Group Term Windows	This is a toggle button that groups and ungroups the Common Window and the Term Window. ON — Group: the Term Windows follow the Common Window when the Common Window is moved. OFF — Ungroup: the Term Windows and the Common Window move independently.

Help Menu

The Help menu has three entries; see TABLE A-4 for a description.

TABLE A-4 Help Menu

Entry	Function
Help	Displays a Help window—the interface to the Sun online help system.
About	Displays the About box, which contains information on the Cluster Console application, such as version number.
Comments	Displays the Comments box, which allows you to enter comments about the software and send them to the development team.

.

Text Field

The text field is where you enter commands that you want to have executed simultaneously on multiple nodes. The state of the host toggle buttons under the Hosts menu determines which nodes receive this input.

Term Windows

The Term Window is just like a normal terminal window. To type on only one host, move the cursor to the Term Window of the desired host and type directly into it.

The Cluster Console Term Windows are like other terminal programs, such as `xterm`, `cmdtool`, and `shelltool`, except that they can also receive input from the Common Window. The Term Windows use VT220 terminal emulation.

The environment variable `TERM` informs your editor of your terminal type. If you are having display problems from `vi` or any other tools, set the environment variable using the appropriate commands for your shell.

The Term Window contains additional functionality, which you can access by positioning the pointer over the Term Window and pressing the right mouse button. This displays the menu described in TABLE A-5.

TABLE A-5 Term Window Menu Entries

Entry	Function
Disable/Enable Scroll Bar	Toggles the scroll bar display on and off in the Term Window.
Exit This Window	Closes the current Term Window.

.

Using the Cluster Console

To issue commands to multiple nodes simultaneously:

- **Position the cursor in the text field of the Common Window and enter your command.**

  Every keystroke entered in this field is sent to all hosts that are currently selected for input.

To issue commands to a single node:

- **Position the cursor in the corresponding Term Window and enter your command.**

  Alternatively, you can turn off all hosts in the Hosts menu, except the one you want to access. Then issue your commands from the Common Window.

Administering Configuration Files

Two configuration files are used by Cluster Console: `clusters` and `serialports`. These files are created automatically by `cluster_tool_setup`, which places them in `/etc`.

The `clusters` File

The `clusters` configuration file maps a cluster name to the list of hostnames that make up the cluster. Each line in this database corresponds to a cluster. The format is:

clustername hostname–1 hostname–2 `[...]` *hostname–n*

For example:

```
cities chartres izmir tampico inchon essen sydney
```

The `clusters` file is used to map cluster names to host names on the command line and in the Select Hosts dialog.

The `serialports` File

The `serialports` file maps each host name to the terminal concentrator and the terminal concentrator serial port to which it is connected. Each line in this database specifies a separate serial port using the format:

hostname terminal_concentrator serial_port

For example:

```
chartres cities-tc 5002
izmir cities-tc 5003
```

The `serialports` file is used by `cconsole` to determine which terminal concentrator and serial ports to connect to for the various cluster nodes that have been specified on the command line or the Select Hosts dialog.

APPENDIX **B**

Using the Command Line to Install and Remove HPC ClusterTools 4 Software

The easiest way to configure and install Sun HPC ClusterTools 4 software is to use the configuration tool, `install_gui`, as described in the *Sun HPC ClusterTools 4 Installation Guide*. If you can't (or prefer not to) use `install_gui`, you may install the software from the command line as described in this appendix.

The figure below shows the principal steps involved in installing the HPC ClusterTools 4 software, including verifying that system requirements are met before starting the installation and verifying the success of the installation after the installation is complete. The pre- and post-installation tasks are described in the *Sun HPC ClusterTool 4 Installation Guide*, Chapters 2 and 5, respectively.

This appendix includes instructions for

- Editing the cluster configuration file, `hpc_config`
- Using the `cluster-tool` installation method
- Installing the HPC ClusterTools software at the command line
- Removing the HPC ClusterTools software
- Removing and installing individual packages
- Selecting which HPC ClusterTools version will be active, 3.1 or 4
- Adding or removing nodes in an existing cluster

The installation steps are diagrammed in FIGURE B-1.

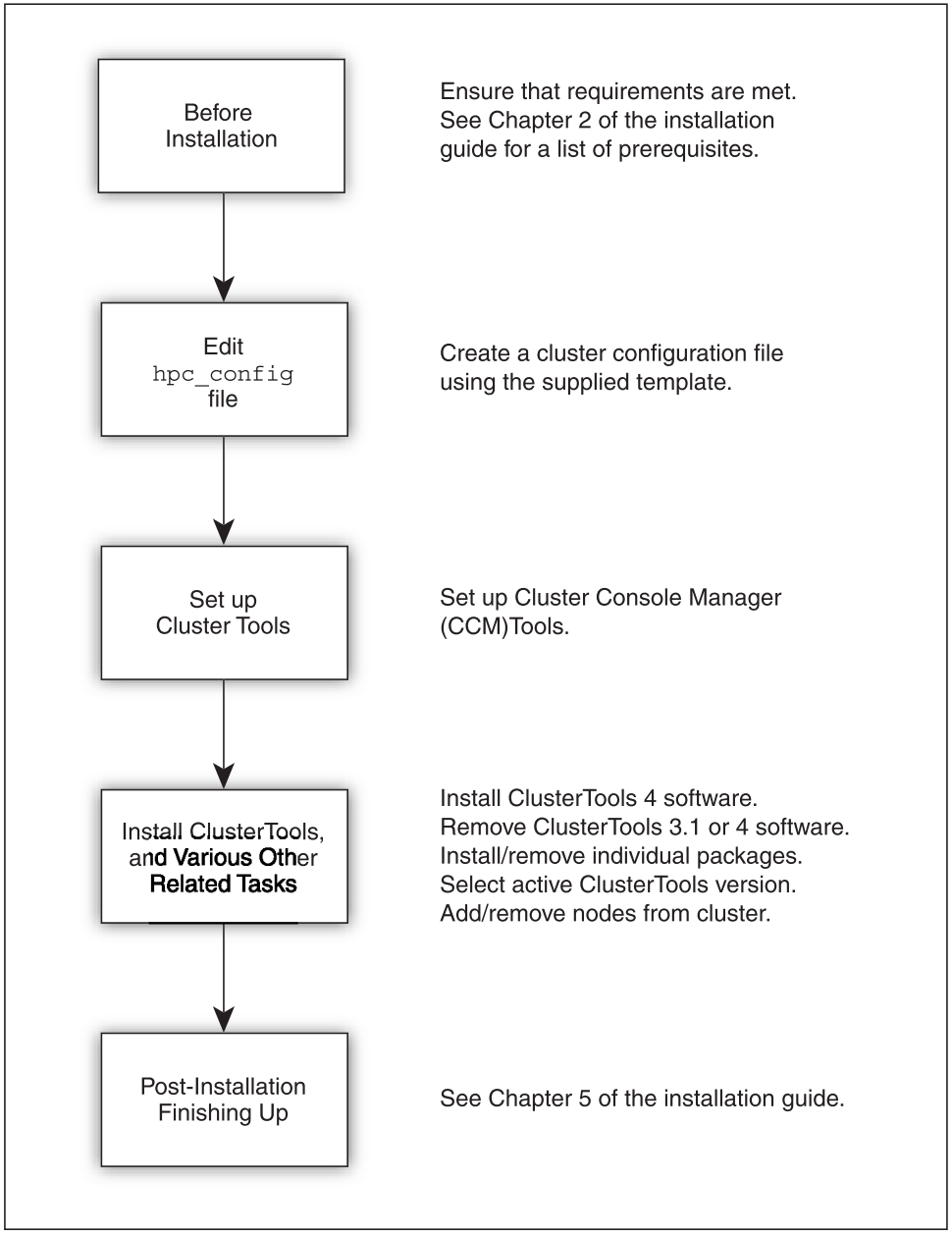

FIGURE B-1 Installing Sun HPC ClusterTools 4 Software, Task Summary

Preparing for Installation

Before installing Sun HPC ClusterTools 4 software, you need to ensure that the hardware and software that make up your cluster meet certain requirements. Requirements are outlined in Chapter 2 of the *Sun HPC ClusterTools 4 Installation Guide*. Review them before proceeding with the instructions in this appendix.

Note – During a local installation, Sun HPC ClusterTools installation software detects whether the server is running Solaris 8 in 32-bit or 64-bit mode. To install both 32-bit and 64-bit binaries, the server must be running Solaris 8 in 64-bit mode. If the server is running Solaris 8 in 32-bit mode, only 32-bit binaries are installed.

Accessing and Editing `hpc_config`

Many aspects of the Sun HPC ClusterTools 4 installation process are controlled by a configuration file called `hpc_config`. Instructions for accessing and editing `hpc_config` are provided in this section.

`hpc_config` Template

A template for `hpc_config` is provided on the Sun HPC ClusterTools distribution CD-ROM to simplify creation of this file. The `hpc_config` template is located in

```
/cdrom/hpc_4_0_ct/Product/Install_Utilities/config_dir/hpc_config
```

Before starting the installation process, you should

- Choose a node to serve as the installation platform
- Copy the `hpc_config` template to a directory on that node
- Edit the template so that it satisfies your site-specific installation requirements

The resulting `hpc_config` file must be located in a directory within a file system that is mounted with read/write/execute permissions (`chmod` = 777) on all the other nodes in the cluster.

Accessing the hpc_config Template

To access hpc_config on the distribution CD-ROM, perform the following steps on the node chosen to be the installation platform:

1. Mount the CD-ROM path on all the nodes in the cluster.
2. Load the Sun HPC ClusterTools 4 distribution CD-ROM.
3. Copy the configuration template onto the node. In the following example, *config_dir_install* represents a directory that you have chosen to be the location where the configuration files will reside. All cluster nodes must be able to read from and write to this directory.

```
# cd config_dir_install
# cp /cdrom/hpc_4_0_ct/Product/Install_Utilities/config_dir/hpc_config .
```

4. Edit the hpc_config file according to the instructions provided in the next section.

Editing hpc_config

FIGURE B-2 shows the basic hpc_config template, but without most of the comment lines provided in the online template. The template is simplified here to make it easier to read and because each section is discussed in detail below. The template comprises six sections:

- *Supported Software Installation* – All installations must complete this first section.
- *General installation information* – All installations must complete this section.
- *For all installations* – All installations must complete this section.
- *Information for NFS installations* – Complete this section *only* if you are installing the software on an NFS server.
- *List of nodes from which HPC ClusterTools software is to be removed* – Use this section when you want to remove HPC ClusterTools software from one or more nodes. List the host names of those nodes in this section.
- *List of nodes to which HPC ClusterTools software is to be added* – Use this section when you want to add HPC ClusterTools software to one or more nodes. List the host names of those nodes in this section.

When you specify the host names of nodes anywhere in the hpc_config file (such as MASTER_NODE, NODES, NFS_SERVER, ADD_NODES, and REMOVE_NODES), use the short name (omit the domain name). The short name must match the output of the Solaris hostname command.

Sun HPC ClusterTools 4 software allows you to specify domain names. However, the domain names are not used during installation.

Sun HPC ClusterTools 4 software requires that all nodes reside in the same domain. Do not combine nodes from different domains in the same installation.

```
#   Section I - Supported Software Information

# Do you want to run HPC 4 ClusterTools software with LSF software?
LSF_SUPPORT="<choice>"

# Part A: Running HPC 4 ClusterTools software with LSF software.

# Do you want to modify LSF parameters to optimize HPC job launches?
MODIFY_LSF_PARAM="<choice>"
# Specify the name of the LSF Cluster.
LSF_CLUSTER_NAME="<clustername>"

# Part B: Running HPC 4 ClusterTools software without LSF software.

# Supply Master Node
MASTER_NODE="<hostname>"
# Authentication
AUTHENTICATION="<choice>"
CREATE_REMOTE_AUTH_FILE="<choice>"

#   Section II - General Installation Information

# Type of Installation Configuration
INSTALL_CONFIG="<choice>"
# Installation Location
INSTALL_LOC="/opt"
# CD-ROM Mount Point
CD_MOUNT_PT="/cdrom/hpc_4_0_ct"

#   Section III - For All Installations

# Installation Method
INSTALL_METHOD="<method>"
# Hardware Information
NODES-"<hostname1> <hostname2> <hostname3>"

#   Section IV - For NFS Installation

# NFS Server
NFS_SERVER=""
# Location of the Software Installed on the NFS Server
INSTALL_LOC_SERVER=""

#   Section V - Removing Nodes

# Enter the name of the nodes from which the ClusterTools will be removed.
REMOVE_NODES=""

#   Section VI - Adding Nodes

# Enter the name of the nodes to which the ClusterTools will be added.
ADD_NODES=""
```

FIGURE B-2 `hpc_config` Template (With Most Comment Lines Removed)

Section I – Supported Software Installation

This section must be completed for all installations.

LSF Support

If you will be using the CRE for resource management, enter `no` and proceed directly to Part B of this section. If you will be using LSF, enter `yes` and then fill in Part A.

```
LSF_SUPPORT="no"
```

Part A: Running HPC ClusterTools With LSF Software

If your cluster will be running LSF resource management software, specify whether you want the installation script to modify certain LSF parameters to optimize HPC job launching. For example:

```
MODIFY_LSF_PARAM="yes"
```

Also, enter the name of the LSF cluster. This is an arbitrary name of your choice.

```
LSF_CLUSTER_NAME="clustername"
```

Part B: Running HPC ClusterTools Without LSF Software

If your cluster will be using CRE for resource management, complete this part.

Specify the host name of the node that you want to serve as master node. The main feature of this node is that it is the platform on which a set of master daemons run.

```
MASTER_NODE="hostname"
```

Also, specify your choice for authentication support. Your options are: DES (`des`), Kerberos 5 (`krb5`) or standard UNIX authentication (`none`).

```
AUTHENTICATION="choice"
```

If you choose not to install any authentication (`none`), you may elect to use a Sun HPC authentication file by entering `yes`, as follows.

```
CREATE_REMOTE_AUTH_FILE="yes"
```

Specifying `yes` prompts the installation scripts to create a file named `sunhpc_rhosts` on the master node, containing a list of remote hosts that are authorized to access the cluster.

If you enter `no`, your `.rhosts` file will be used for authentication. If you set `AUTHENTICATION` to `krb5` or `des`, you *must* enter `no` in this field. For further information about `.rhosts`, see the `rhosts` man page.

Section II – General Installation Information

This section must be completed for all installations.

Type of Installation

Three types of installation are possible for Sun HPC ClusterTools:

- `nfs` – Install the HPC ClusterTools software on an NFS server for remote mounting on client nodes.
- `smp-local` – For single-node clusters only. Install the HPC ClusterTools software locally on this node for use only on this node.
- `cluster-local` – For multinode clusters only. Install the HPC ClusterTools software locally on every node in the cluster.

Specify one of the installation types: `nfs`, `smp-local`, or `cluster-local`. There is no default type of installation.

```
INSTALL_CONFIG="config_choice"
```

Installation Location

The way the `INSTALL_LOC` path is used varies, depending on which type of installation you have chosen.

- For local installations (`smp-local` or `cluster-local`), `INSTALL_LOC` is the path where the packages will actually be installed.
- For NFS installations (`nfs`), `INSTALL_LOC` is the mount point for the software on the NFS *clients*.

You must enter a full path name. The default location is `/opt`. Read/write permissions must be set on all installation locations (or NFS mount points) on all the nodes in the cluster.

```
INSTALL_LOC="/opt"
```

If you choose an installation directory other than the default `/opt`, a symbolic link is created from `/opt/SUNWhpc` to the chosen installation location/mount point.

CD-ROM Mount Point

Specify a mount point for the CD-ROM. This mount point must be mounted on (that is, NFS-accessible to) all the nodes in the cluster. The default mount point is `/cdrom/hpc_4_0_ct`. For example:

```
CD_MOUNT_PT="/cdrom/hpc_4_0_ct"
```

Section III – General Installation Information

This section must be completed for all installations.

Installation Method Options

Specify either `rsh` or `cluster-tool` as the method for propagating the installation to all the nodes in the cluster.

- `cluster-tool` – If you choose the `cluster-tool` option, you will be able to use one of the Cluster Console Manager (CCM) applications, `cconsole`, `ctelnet`, or `crlogin`, to facilitate the installation of the HPC ClusterTools software on all of the nodes in parallel. See Appendix A for information about using CCM tools.
- `rsh` – If you choose the `rsh` method, the software will be installed serially on the cluster nodes in the order in which they are listed in `hpc_config`. The CCM applications cannot be used to install the software in `rsh` mode.

Also note that `rsh` requires that all nodes be trusted hosts—at least during the installation process.

```
INSTALL_METHOD="method"
```

Hardware Information

There are two ways to enter information in this section:

- If the cluster nodes are connected to a terminal concentrator, list each node in the following triplet format.

```
NODES="hostname1/termcon_name/port_id  hostname2/termcon_name/port_id  ..."
```

In each triplet, specify the host name of a node, followed by the host name of the terminal concentrator and the port ID on the terminal concentrator to which that node is connected. Separate the triplet fields with virgules (/). Use spaces between node triplets.

- If the cluster nodes are not connected to a terminal concentrator, simply list the node host names, separated by spaces, as follows.

```
NODES="hostname1  hostname2  hostname3  ..."
```

Section IV – For NFS Installations Only

Complete this section only if you are installing the software on an NFS server.

NFS Server Host Name

The format for setting the NFS server host name is the same as for setting the host names for the nodes in the cluster. There are two ways to define the host name of the NFS server:

- If you have a terminal concentrator, describe the NFS server in the following triplet format.

```
NFS_SERVER="hostname/termcon_name/port_id"
```

- If you do not have a terminal concentrator, simply specify the host name of the NFS server.

```
NFS_SERVER="hostname"
```

The NFS server can be one of the cluster nodes or it can be external (but connected) to the cluster. If the server will be part of the cluster—that is, will also be an execution host for the Sun HPC ClusterTools suite—it must be included in the NODES field. If the NFS server will *not* be part of the cluster, it must be available from all the hosts listed in NODES, but it should not be included in the NODES field.

Location of the Software on the Server

Determine the location where you want to install the software on the NFS server. Specify the directory in INSTALL_LOC_SERVER.

```
INSTALL_LOC_SERVER="directory"
```

Recall that the directory specified in INSTALL_LOC defines the mount point for INSTALL_LOC_SERVER on each NFS client.

Section V – Removing Nodes

When you want to remove HPC ClusterTools 4 software from one or more nodes, list the host names of the nodes in this section.

```
REMOVE_NODES=""
```

Use the same host name format as is used in the NODES="" section; that is, *hostname* or *hostname/termcon_name/portid*.

Detailed instructions for removing nodes from a cluster are provided in "Removing HPC ClusterTools 3.1 or 4 Software" on page 164.

Section VI – Adding Nodes

When you want to add HPC ClusterTools 4 software to one or more nodes, list the host names of the nodes in this section.

```
ADD_NODES=""
```

Use the same host name format as is used in the NODES="" section; that is, *hostname* or *hostname/termcon_name/portid*.

Detailed instructions for adding nodes to a cluster are provided in "Adding Nodes to an Existing Cluster" on page 173.

Note – Use the ADD_NODES field only to add nodes. Do not use it to change NFS servers or CRE master nodes.

Re-Using an `hpc_config` file created by `install_gui`

The `hpc_config` file created using during installation specifies two installation method options. The pair of options differ, depending on whether the `hpc_config` file is created by the installation tool (`install_gui`) or the file is edited by hand to use the manual installation option.

TABLE B-1 Options for the `INSTALL_METHODS` field in the `hpc_config` file

install_gui	manual
rsh	rsh
telnet	cluster-tool

To reuse an `hpc_config` file that has been created by the installation tool for a later (manual) installation, change `telnet` to `cluster-tool` in the `hpc_config` file. If the `INSTALL_METHODS` option is `rsh`, you can leave the option unchanged.

Run `cluster_tool_setup`

If you have chosen the `cluster-tool` installation method option (described in "Installation Method Options" on page 157) and plan to use the CCM tools, you need to run the `cluster_tool_setup` script first. This loads the CCM administration tools onto a node and creates a cluster configuration file that is used by CCM applications. You must be superuser to run `cluster_tool_setup`.

See Appendix A for a description of the three CCM applications, `cconsole`, `ctelnet`, and `crlogin`.

Note – `cconsole` requires the nodes to be connected to a terminal concentrator. The others, `ctelnet` and `crlogin`, do not.

If you want to use `cconsole` to monitor messages generated *while rebooting the cluster nodes*, launch it from a machine *outside* the cluster. If you launch it from a cluster node, it will be disabled when the node from which it is launched reboots.

1. **Go to the `Install_Utilities` directory on the Sun HPC ClusterTools 4 distribution CD-ROM.**

   Note that this directory must be mounted on (accessible by) all nodes in the cluster.

   ```
   % su
   Password: root_password
   # cd /cdrom/hpc_4_0_ct/Product/Install_Utilities/bin
   ```

2. **If you are running on a node within the cluster, perform Step a. If you are running on a machine *outside* the cluster, perform Step b.**

   a. ***Within* the cluster, run `cluster_tool_setup -c`.**

      Run `cluster_tool_setup`; use the `-c` tag to specify the directory containing the `hpc_config` file.

      ```
      # ./cluster_tool_setup -c config_dir_install
      ```

   b. ***Outside* the cluster, run `cluster_tool_setup -c -f`.**

      Run `cluster_tool_setup`. Use the `-c` tag to specify the directory containing the `hpc_config` file and the `-f` tag to force the installation to take place.

      ```
      # ./cluster_tool_setup -c config_dir_install -f
      ```

3. **Set the `DISPLAY` environment variable to the machine on which you will be running the CCM tools.**

   ```
   # setenv DISPLAY hostname:0
   ```

4. **Invoke the CCM tool of your choice. If the nodes are connected to a terminal concentrator, specify `cconsole`. If not, specify either `ctelnet` or `crlogin`.**

   All three tools reside in `/opt/SUNWcluster/bin`. For example,

   ```
   # /opt/SUNWcluster/bin/ctelnet
   ```

   All three CCM tools require the name of the cluster as an argument.

   The CCM tool then creates a Common Window and separate Term Windows for all the nodes in the cluster.

5. **Position the cursor in the Common Window and press Return.**

   This activates a prompt in each Term Window. Note that the Common Window does not echo keyboard entries. These appear only in the Term Windows.

   You can now use CCM to install the software packages or to remove HPC ClusterTools software, either version 3.1 or 4, as described in the following sections.

Installing HPC ClusterTools 4 Software

This section describes the procedure for installing the Sun HPC ClusterTools 4 software.

The installation process requires the following directories:

- CD-ROM mount point – The CD-ROM must be mounted on all nodes, either locally or via NFS, on which the HPC ClusterTools software will be installed.
- Configuration file directory – The installation process creates a configuration file with the default name `hpc_config`. You will be asked to specify where you want this file to be saved. You can also save it with a file name other than `hpc_config`. All nodes involved in the HPC ClusterTools installation must have read and write access (as `root`) to the saved configuration file and to the directory in which it resides. This directory must be accessible using the same path from all nodes.

  This directory also serves as a synchronization area during the install.
- Installation target directory – You will be asked to specify where you want the HPC ClusterTools packages to be installed. For local installations, the default installation directory is `/opt`.

Note – If the software will be installed on an NFS server, the NFS installation directory must be mounted on each NFS client in the cluster. Specify the installation directory as the NFS mount point on the NFS client.

Note that if you are installing HPC ClusterTools 4 software on an NFS server, you must set up root-level access to the server's installation directory on every node in the cluster. To set up root-level access, use `share_nfs`. For example,

```
# share -F nfs -o root=nfs.client1:nfs.client2:...
```

For further information about `share_nfs`, see the `share_nfs(1M)` man page.

Installation Procedure

The exact procedure for each step depends on which installation mode you are in, `cluster-tool` or `rsh`.

- In `cluster-tool` mode, perform each step in the Common Window. Each entry will be echoed in every Term Window.
- In `rsh` mode, perform each step at the shell prompt of one of the nodes.

See Appendix A for more information about the CCM tools that are available to you in `cluster-tool` mode.

Note – The `hpc_install` command writes various `SYNC` files in the directory containing `hpc_config` as part of the package installation. If the installation process stops prematurely (for example, if you press `Ctrl-C`), some `SYNC` files may be left. You must remove these files before executing `hpc_install` again so they don't interfere with the next software installation session.

---

1. **Log in to each node as root.**

   If you selected the `cluster-tool` installation method, you need to know the superuser password for each node.

2. **Go to the `Install_Utilities` directory on the Sun HPC ClusterTools 4 distribution CD-ROM.**

   Note that this directory must be mounted on all nodes in the cluster.

   ```
   # cd /cdrom/hpc_4_0_ct/Product/Install_Utilities/bin
   ```

3. **Run** `hpc_install`.

```
# ./hpc_install -c config_dir_install
```

where *config_dir_install* represents the directory containing the `hpc_config` file.

The `-c` tag causes `hpc_install` to look for a file named `hpc_config` in the specified directory. If you want to install the software using a configuration file with a different name, you must specify a full path including the new file name after the `-c` tag.

4. **Activate the software.**

`hpc_install` prompts you to activate the software. Answer *yes* and the software will be activated and ready for use.

If you choose *not* to activate the software at this time, the software will not be ready for use. To activate the software later, run `hpc_reconfigure`, using the `-c` option to specify the directory containing the `hpc_config` file and the `-a` option to specify the HPC ClusterTools version to be activated.

```
# ./hpc_reconfigure -c config_dir_install -a 4.0
```

Removing HPC ClusterTools 3.1 or 4 Software

You can remove either version of the HPC ClusterTools software, 3.1 or 4, by running `hpc_remove` from a shell.

The installation process will have created a configuration file with the default name `hpc_config`. It may have been saved with a file name other than `hpc_config`. All nodes involved in the HPC ClusterTools installation must have read and write access (as `root`) to the saved configuration file and to the directory in which it resides. This directory must be accessible using the same path from all nodes.

The removal procedure is described below.

Using the Existing hpc_config File

1. **Locate the cluster configuration file, hpc_config.**

   Find the directory in which the hpc_config file resides. You will need to supply this path on the hpc_remove command line in Step 4. If you cannot locate the original hpc_config file, you will have to create one, as described in "Accessing and Editing hpc_config" on page 151.

2. **Go to the Install_Utilities directory**

   This directory must be mounted with read/execute permissions on all the nodes in the cluster:

   ```
   # cd /opt/SUNWhpc/HPC4.0/bin/Install_Utilities/bin
   ```

3. **Run hpc_remove; use the -c option to specify the directory containing the hpc_config file and the -r option to specify the version of the software to be removed.**

   ```
   # ./hpc_remove -c config_dir_install -r version_number
   ```

   The -c option causes hpc_remove to look for a file named hpc_config in the specified directory. If you want to remove the software using a configuration file with a different name, you must specify a full path including the new file name after the -c option.

   Enter either 3.1 or 4 as the *version_number* argument to -r.

Creating a Replacement hpc_config File

If the hpc_config file has been deleted, or cannot be found, you can remove either version of the HPC ClusterTools software, 3.1 or 4, by re-creating the hpc_config file, then running hpc_remove. The procedure for doing this is described below.

Copy the Template of the hpc_config File

A template for hpc_config is provided on the Sun HPC ClusterTools distribution CD-ROM to simplify creation of this file. The hpc_config template is located in

```
/opt/SUNWhpc/HPC4.0/bin/Install_Utilities/config_dir/hpc_config
```

Edit the hpc_config File

You must complete sections I through IV to enable the removal scripts to operate effectively.

Section I: Specify the Resource Manager

Part A: To identify the installed resource manager, use the hpc_rte command. For example,

```
# /opt/SUNWhpc/bin/hpc_rte
lsf
```

If your installation uses the CRE for resource management, hpc_rte returns cre. Enter no in the LSF_SUPPORT field and proceed directly to Part B. If hpc_rte returns lsf, enter yes in the LSF_SUPPORT field and fill in the rest of Part A.

```
LSF_SUPPORT="yes"
```

If your cluster runs LSF resource management software, specify whether certain LSF parameters have been modified to optimize HPC job launching. To verify whether these LSF parameters have been modified, look in /etc/lsf.conf and identify the value of LSB_CONFDIR. Then, examine

```
$LSB_CONFDIR/cluster_name/configdir/lsb.params
```

If lsb.params contains references to SunHPC, then set MODIFY_LSF_PARAM in your hpc_config file to yes. For example:

```
MODIFY_LSF_PARAM="yes"
```

Next, enter the name of the LSF cluster. To identify the LSF cluster's name, use the lsid command. For example,

```
# /opt/SUNWlsf/bin/lsid
LSF 4.0.1, Jun 2 2000
Copyright 1992-2000 Platform Computing Corporation
My cluster name is ct-srv
My master name is ct-srv-0
```

Enter the cluster's name in the LSF_CLUSTER_NAME field. In this example, the name is ct-srv.

```
LSF_CLUSTER_NAME="ct-srv"
```

Part B: If your cluster uses CRE for resource management, you must specify the host name of the node that serves as master node. The main feature of this node is that it is the platform on which a set of master daemons run. To identify the master node, view the contents of /etc/hpc-system. For example,

```
# cat /etc/hpc-system
ct-srv-0
```

Set MASTER_NODE in your hpc_config file to the name specified in hpc-system. For example,

```
MASTER_NODE="ct-srv-0"
```

Next, specify the installed authentication method. The options are: DES (des), Kerberos 5 (krb5) or standard UNIX authentication (none).

To identify the installed authentication method

1. Type

```
# pkginfo | grep SUNWcre
```

   then

2. Type

```
# pkgparam SUNWcre.instance CRE_AUTH
```

   where *instance* is the highest instance number (if any) reported by pkginfo in step 1. This command returns the authentication method.

For example,

```
# pkginfo | grep SUNWcre
system     SUNWcre     HPC Cluster Runtime Environment
system     SUNWcre.2     HPC Cluster Runtime Environment
# pkgparam SUNWcre.2 CRE_AUTH
none
```

Specify that authentication method in your `hpc_config` file. For example,

```
AUTHENTICATION="none"
```

If a file named `/etc/sunhpc-rhosts` exists on the CRE master node, set `CREATE_REMOTE_AUTH_FILE` in your `hpc_config` file to `yes`. Otherwise, set this to `no`.

```
CREATE_REMOTE_AUTH_FILE="yes"
```

Section II: Specify Information About Installation Types and Locations

Three types of installation are possible for Sun HPC ClusterTools:

- `nfs` – HPC ClusterTools software has been installed on an NFS server for remote mounting on client nodes.
- `smp-local` – For single-node clusters only. HPC ClusterTools software has been installed locally on this node for use only on this node.
- `cluster-local` – For multinode clusters only. HPC ClusterTools software has been installed locally on every node in the cluster.

Specify one of the installation types: `nfs`, `smp-local`, or `cluster-local`. There is no default type of installation.

To determine which type of installation you have, issue one or more of these commands:

- `mpinfo -N`
- `lshosts`
- `bhosts`

If these commands report a single host name, your installation type is `smp-local`. For example,

```
# mpinfo -N
NAME       UP  PARTITION  OS     OSREL  NCPU  FMEM  FSWP  LOAD1  LOAD5  LOAD1
ct-srv-0   y   all        SunOS  5.8    28    2364  3166  6.58   6.43   36

# lshosts
HOST_NAME  type     model     cpuf   ncpus   maxmem   maxswp   server  RESOURCES
ct-srv-0   DEFAULT  DEFAULT   1.0    28      3584M    4293M    Yes     ()

# bhosts
HOST_NAME    STATUS   JL/U   MAX   NJOBS   RUN   SSUSP   USUSP   RSV
ct-srv-0     ok       -      -     2       2     0       0       0
```

If these commands report multiple hosts, you have an `nfs` or `cluster-local` installation. On each node in your cluster, type

```
# mount | grep SUNWhpc
```

If the nodes report a mount, then your installation is `nfs`. Otherwise your installation is `cluster-local`.

Set `INSTALL_CONFIG` to your installation type. For example,

```
INSTALL_CONFIG="smp-local"
```

Specify the installation location. If you type

```
# ls -l /opt
...
drwxr-xr-x    3 root     bin          512 Dec   6 22:09 SUNWhpc
...
```

then look at `SUNWhpc`. If `SUNWhpc` is a directory, then set `INSTALL_LOC` to `/opt`. If `SUNWhpc` is a symbolic link, set `INSTALL_LOC` to the directory pointed to by `SUNWhpc`.

You must enter a full path name. For example,

```
INSTALL_LOC="/opt"
```

Specify a mount point for the CD-ROM. This mount point must be mounted on (that is, NFS-accessible to) all the nodes in the cluster. The default mount point for HPC ClusterTools 4 is `/cdrom/hpc_4_0_ct`. For example:

```
CD_MOUNT_PT="/cdrom/hpc_4_0_ct"
```

Section III: Specify Information About Installation Methods

If your installation uses either an NFS server for remote mounting or exists locally on each node of a multinode cluster, you need to complete this section.

Specify either `rsh` or `cluster-tool` as the method for propagating the installation to all the nodes in the cluster. If a directory `/opt/SUNWcluster` exists, then set the `INSTALL_METHOD` to cluster–tool. Otherwise, set `INSTALL_METHOD` to `rsh`.

If your installation was done using the `rsh` method, there will be a `.rhosts` file in the root directory.

For example,

```
INSTALL_METHOD="cluster-tool"
```

Specify the list of nodes in the cluster.

If your resource manager is CRE, you can get a list of nodes using the `mpinfo -N` command. If your resource manager is LSF, you can get a list of nodes using either the `bhosts` or `lsload` commands.

There are two ways to enter information in the `NODES` section:

- If the cluster nodes are connected to a terminal concentrator, list each node in the following triplet format.

  `NODES="`*hostname1/termcon_name/port_id hostname2/termcon_name/port_id ...*`"`

  In each triplet, specify the host name of a node, followed by the host name of the terminal concentrator and the port ID on the terminal concentrator to which that node is connected. Separate the triplet fields with virgules (/). Use spaces between node triplets.

- If the cluster nodes are not connected to a terminal concentrator, simply list the node host names, separated by spaces, as follows.

  `NODES="`*hostname1 hostname2 hostname3 ...*`"`

Section IV: Specify NFS Information

Complete this section only if your software is installed on an NFS server.

The format for setting the NFS server host name is the same as for setting the host names for the nodes in the cluster. There are two ways to define the host name of the NFS server:

- If you have a terminal concentrator, describe the NFS server in the following triplet format:

  `NFS_SERVER="`*hostname/termcon_name/port_id*`"`

- If you do not have a terminal concentrator, simply specify the host name of the NFS server.:

```
NFS_SERVER="hostname"
```

The NFS server can be one of the cluster nodes or it can be external (but connected) to the cluster. If the server is part of the cluster—that is, it is also an execution host for the Sun HPC ClusterTools software—it must be included in the `NODES` field. If the NFS server is *not* part of the cluster, it must be available from all the hosts listed in `NODES`, but it should not be included in the `NODES` field.

The directory specified in `INSTALL_LOC_SERVER` is the location where the software was installed on the NFS server.

```
INSTALL_LOC_SERVER="directory"
```

Recall that the directory specified in `INSTALL_LOC` defines the mount point for `INSTALL_LOC_SERVER` on each NFS client.

Remove the HPC ClusterTools Software

Follow the procedure described in "Using the Existing `hpc_config` File" on page 165.

Removing and Reinstalling Individual Packages

To remove a single package and install (or reinstall) another package in its place, perform the following steps:

```
# cd /opt/SUNWhpc/HPC4.0/bin/Install_Utilities/bin
# ./hpc_remove –c config_dir_install -d package_name
# ./hpc_install –c config_dir_install -d package_name
```

For example:

```
# cd /opt/SUNWhpc/HPC4.0/bin/Install_Utilities/bin
# ./hpc_remove -c /home/hpc_admin -d SUNWs3l
# ./hpc_install -c /home/hpc_admin -d /cdrom/hpc_4_0_ct/Product/SUNWs3l
```

Selecting the Active HPC ClusterTools Version

When both HPC ClusterTools versions 3.1 and 4 are installed on a cluster, you can use `hpc_reconfigure` to specify which version is to be active.

1. **Locate the cluster configuration file, `hpc_config`.**

   Find the directory in which the `hpc_config` file resides. You need to supply this path on the `hpc_reconfigure` command line in Step 4. If you cannot locate the original `hpc_config` file, you will have to create one, as described in "Accessing and Editing `hpc_config`" on page 151.

2. **Go to the `Install_Utilities/bin` directory on the CD-ROM.**

   This directory must be mounted with read/execute permissions on all the nodes in the cluster:

   ```
   # cd /opt/SUNWhpc/HPC4.0/bin/Install_Utilities/bin
   ```

3. **Select the version that you want to be active. To do this, run `hpc_reconfigure`, using the `-c` option to specify the directory containing the `hpc_config` file and the `-r` option to specify the HPC ClusterTools version to be activated.**

   ```
   # ./hpc_reconfigure -c config_dir_install -r 4.0
   ```

   The `hpc_reconfigure` script tries to retain any customizations you may have in HPC ClusterTools 3.1 when it reconfigures your system to HPC ClusterTools 4. If that effort fails, the script issues a warning message that it has used default values to create `hpc.conf`. In that case, you must edit `hpc.conf` on each node in your cluster to keep customizations from HPC ClusterTools 3.1. See Chapter 7 "`hpc.conf` Configuration File" for more information about editing `hpc.conf`.

Adding/Removing Nodes in an Existing Cluster

This section describes the procedures for adding nodes to and removing nodes from an existing cluster. It also explains how to *remaster* a cluster when a master node is added or removed.

Note – These procedures do not support adding an NFS server to or removing it from an existing cluster.

Adding Nodes to an Existing Cluster

Perform the steps described below to add one or more nodes to an existing cluster.

Note – This procedure assumes that you use the same installation method, `cluster-tools` or `rsh`, as was used to install the HPC ClusterTools 4 software originally.

---

1. **If you will be using the `cluster-tool` installation method, you need to take the following steps first. If you will be using the `rsh` installation method, skip this step and go directly to Step 2.**

   a. **Add the host name(s) of the new node(s) to the relevant cluster description in the `/etc/clusters` file.**

   b. **Add the node host name(s) and the applicable terminal concentrator host name and port information to the `/etc/serialports` file.**

2. **Update the `hpc_config` file with the nodes being added.**

   Add the host names of the new nodes to both the `NODES` and `ADD_NODES` fields of `hpc_config`. See "Hardware Information" on page 158 for a description of host names syntax in the `NODES` field. Use the same syntax in the `ADD_NODES` field.

3. **Update the cluster's security facility. If your cluster uses either DES or Kerberos 5, update the authentication system to accommodate the new nodes. Otherwise, update the appropriate security files on each of the cluster nodes to reflect the new nodes, such as: `/hosts.equiv`, `/etc/sunhpc_rhosts`, and `/.rhosts`.**

4. **Execute `/opt/SUNWhpc/bin/Install_Utilities/bin/hpc_install` on all of nodes in the cluster.**

   Use the `-c` switch to specify the location of the configuration file. This could be either the directory name containing the default configuration file `hpc_config` or the full path name of the configuration file with the customized file name.

```
# cd /opt/SUNWhpc/bin/Install_Utilities/bin
# hpc_install -c (default_config_dir | full_custom_config_path)
```

   For example, if you use the default configuration file `/usr/admin/hpc_config`, the `hpc_install` command line should look like this:

```
# cd /opt/SUNWhpc/bin/Install_Utilities/bin
# hpc_install -c /usr/admin
```

5. **Delete the host names of the newly added nodes from the `ADD_NODES` field of the configuration file.**
6. **If one of the new nodes is to be the cluster's new master node, perform the steps described in "Remastering the Cluster" on page 177.**
7. **If any of the new nodes will be PFS servers, do the following:**
   a. **Update the `/opt/SUNWhpc/etc/hpc.conf` file on all PFS server and client nodes to include the new PFS server(s).**
   b. **Unmount the PFS file systems.**
   c. **Stop and restart the PFS daemons. See Chapter 4 of this manual for instructions.**
8. **If you want the new nodes to be included in partitions, use `mpadmin` to add them to the partitions of interest. See Chapter 6 in this manual for instructions.**

Removing Nodes From an Existing Cluster

Perform the steps described below to remove one or more nodes from an existing cluster.

1. **If one of the nodes to be removed is the cluster's master node, perform the steps described in "Remastering the Cluster" on page 177.**

2. **Update the `REMOVE_NODES` field of the `hpc_config` file with the nodes being removed.**

   List the host names of the nodes to be removed in the `REMOVE_NODES` field. See "Section V – Removing Nodes" on page 159 for a description of the `REMOVE_NODES` field.

3. **Stop all CRE jobs running on the nodes that are to be removed.**

4. **If any of the new nodes to be removed are PFS servers, do the following:**

   a. **Update the `/opt/SUNWhpc/etc/hpc.conf` file on all PFS server and client nodes, deleting references to the PFS server(s) that will be removed.**

   b. **Unmount the PFS file systems.**

   c. **Stop and restart the PFS daemons. See Chapter 4 of this manual for instructions.**

5. **Execute `/opt/SUNWhpc/bin/Install_Utilities/bin/hpc_remove` on all of the nodes in the cluster.**

   Use the `-c` switch to specify the location of the configuration file. This could be either the directory name containing the default configuration file `hpc_config` or the full path name of the configuration file with the customized file name.

   Use the `-r` switch to specify which HPC ClusterTools version is to be removed, 3.1 or 4. For example:

```
# cd /opt/SUNWhpc/bin/Install_Utilities/bin
# hpc_remove -c /usr/admin -r 4.0
```

6. **If you are going to use the `cluster-tool` method of removal, you need to take the following steps first. If you will be using the `rsh` method, skip this step and go directly to Step 7.**

   a. **Delete the host name(s) of the node(s) that will be removed from the relevant cluster description in the `/etc/clusters` file.**

   b. **Delete the node host name(s) and the applicable terminal concentrator host name and port information from the `/etc/serialports` file.**

7. **Make the following changes to the configuration file.**

   a. **Delete the host names from the `REMOVE_NODES` field of the configuration file.**

   b. **Delete the host names of the removed nodes from the `NODES` field of the configuration file.**

8. **Update the cluster's security facility to reflect the removal of the node(s). If either DES or Kerberos is used, update it as applicable. Otherwise, update the security files on each of the cluster nodes to reflect the removal of the node(s). These might include:** `/hosts.equiv`, `/etc/sunhpc_rhosts`, **and** `/.rhosts`.

9. **Activate the cluster. Do this by executing** `hpc_reconfigure`.

   Use the `-c` switch to specify the location of the updated configuration file. Use the `-a` switch to specify which version of HPC ClusterTools is to be activated, either 3.1 or 4. For example:

```
# cd /opt/SUNWhpc/bin/Install_Utilities/bin
# hpc_reconfigure -c /usr/admin -a 4.0
```

10. **Use** `mpadmin` **to update partition descriptions to reflect node removal. See Chapter 6 of this manual for instructions.**

Remastering the Cluster

This section explains how to create a new master node for a cluster. This can happen when a new node is made master node or when the current master node is removed from a cluster.

When adding a new node that will be made master node, do the following after the node is added to the cluster. When removing a current master node from a cluster, do the following before the master node is removed.

1. **Rename the current configuration file.**

2. **Create a new configuration file in which the** `MASTER_NODE` **field specifies the host name of the new master node.**

3. **Stop all CRE jobs.**

4. **If your old configuration includes customized attributes that you would like to incorporate into the new configuration, use the** `mpadmin dump` **command to save the cluster configuration information to a file.**

```
# mpadmin -c dump > file
```

5. **Deactivate the cluster with the old master node. Do this by executing `/opt/SUNWhpc/bin/Install_Utilities/bin/hpc_reconfigure` on each node in the cluster. Specify the same configuration that was used when the cluster was originally configured.**

   Use the `-c` switch to specify the location of the old configuration file. This could be either the name of the directory containing the default configuration file `hpc_config` or the full path name of the configuration file with the customized file name. In either case, the specification must be for the old configuration file.

   Use the `-d` switch to specify the version of HPC ClusterTools that is to be deactivated, either 3.1 or 4. For example,

```
# cd /opt/SUNWhpc/bin/Install_Utilities/bin
# hpc_reconfigure -c /usr/admin -d 4.0
```

6. **Flush the rdb on the old master node.**

```
# rm /var/hpc/rdb-log /var/hpc/rdb-save
```

7. **Activate the cluster with the new master node. Do this by executing `hpc_reconfigure`.**

   Use the `-c` switch to specify the location of the new configuration file. Use the **`-a`** switch to specify which version of HPC ClusterTools is to be activated, either 3.1 or 4. For example:

```
# cd /opt/SUNWhpc/bin/Install_Utilities/bin
# hpc_reconfigure -c /usr/admin -a 4.0
```

8. **If you saved the old configuration attributes to a file (as described in Step 4), you can restore them now with the `mpadmin -f` option.**

```
# mpadmin -f file
```

APPENDIX **C**

Administering the LSF Plugins

A Sun HPC cluster may be configured to use the Load-Sharing Facility (LSF) suite from Platform Computing as its resource manager, as an alternative to Sun CRE. For this case, Sun HPC ClusterTools includes a set of plugins that allow other ClusterTools components to operate with the LSF suite.

This appendix discusses various Sun HPC-specific administration issues for LSF-based clusters. In particular, it describes `hpc.conf`, a Sun HPC configuration file that extends the definition of Sun HPC system attributes beyond the parameters defined in the LSF configuration files.

Before reading this guide, Sun HPC cluster administrators should read the *LSF Batch Administrators Guide*, which is supplied with the LSF software.

The LSF Suite

The LSF suite is a collection of resource-management products that provide distributed batch scheduling, load balancing, job execution, and job termination services across a network of computers. The LSF products used by Sun HPC ClusterTools software are: LSF Base, LSF Batch, and LSF Parallel.

- LSF Base – Provides the fundamental services upon which LSF Batch and LSF Parallel depend. It supplies cluster configuration information as well as the up-to-date resource and load information needed for efficient job allocation. It also supports interactive job execution.
- LSF Batch – Performs batch job processing, load balancing, and policy-based resource allocation.
- LSF Parallel – Extends the LSF Base and Batch services with support for parallel jobs.

Refer to the *LSF Administrator's Guide* for a fuller description of LSF Base and LSF Batch and to the *LSF Parallel User's Guide* for more information about LSF Parallel.

LSF supports the concept of *interactive batch* execution of Sun HPC jobs as well the conventional batch method. Interactive batch mode allows users to submit jobs through the LSF Batch system and remain attached to the job throughout execution.

Sun HPC Configuration File

The Sun HPC configuration file hpc.conf, defines attributes of Sun HPC clusters that are not defined in any LSF configuration files. A single hpc.conf file is shared by all the nodes in a cluster. It resides in the LSF shared directory `LSF_SHAREDIR`.

Sun HPC ClusterTools software is distributed with an hpc.conf template, which you can edit to suit your particular configuration requirements. The `hpc.conf` file follows the formatting conventions of LSF configuration files. That is, each configuration section is bracketed by a `Begin/End` keyword pair and, when a parameter definition involves multiple fields, the fields are separated by spaces.

The `hpc.conf` file is organized into several sections, as described in Chapter 7 of this manual. Only the section `HPCNodes` pertains exclusively to clusters configured with the LSF suite. The `HPCNodes` section can be used to define an HPC cluster that consists of a subset of the nodes contained in the LSF cluster.

Editing `hpc.conf`

When any changes are made to `hpc.conf`, the system should be in a quiescent state. To ensure that it is safe to edit `hpc.conf`, shut down the LSF Batch daemons. See the *LSF Batch Administrator's Guide* for instructions on stopping and starting LSF Batch daemons.

Edit the various sections of `hpc.conf` to suit the need of your cluster. These sections are described in this manual in Chapter 3 (briefly) and Chapter 7 (in detail).

The section `HPCNodes` allows you to define a Sun HPC cluster that consists of a subset of the nodes contained in the LSF cluster. To use this configuration option, enter the host names of the nodes that you want to include in the HPC cluster, one host name per line. An `HPCNodes` section with two nodes listed might look like this example:

```
Begin HPCNodes
node1
node2
End HPCNodes
```

Propagating `hpc.conf` Information

Whenever `hpc.conf` is changed, the LSF daemons must be updated with the new information. After all desired changes to `hpc.conf` have been made, restart the LSF Base daemons `LIM` and `RES`. Use the `lsadmin` subcommands `reconfig` and `resrestart` as follows:

```
# lsadmin reconfig
# lsadmin resrestart all
```

Then, use the `badmin` subcommands `reconfig` to restart `mbatchd` and `hrestart` to restart the slave batch daemons:

```
# badmin reconfig
# badmin hrestart all
```

This only needs to be done from one node. See the *LSF Batch Administrator's Guide* for additional information about restarting LSF daemons.

Creating Sun HPC–Specific Queues

Because HPC jobs distribute multiple processes across multiple nodes, their batch queue requirements are different from those of serial jobs. For this reason, you should specifically configure one or more batch queues for running Sun HPC jobs. This involves editing the queue configuration file, `lsb.queues`.

Specify PAM as Job Starter

The `JOB_STARTER` parameter allows an LSF batch queue to pass job launching control over to a special job-starting procedure rather than launching the job itself. For Sun HPC applications, this job launching role is given to the Parallel Application Manager (PAM), which is a utility for starting and managing MPI jobs.

PAM should be specified as the job starter on all queues that will be used by Sun HPC jobs. To do this, simply edit the `JOB_STARTER` line in `lsb.queues` to read as follows:

```
JOB_STARTER=pam
```

When a Sun HPC job is submitted to a PAM-configured queue, the queue will start PAM running. PAM, in turn, will launch the Sun HPC job on the cluster.

Enable Interactive Batch Mode

LSF supports the concept of *interactive batch* job execution. When a job is submitted in interactive batch mode, it receives the same batch scheduling and host selection services as noninteractive batch jobs, but the terminal from which the job was submitted remains attached to the job as if it were launched interactively.

By default, both batch mode and interactive batch mode are available. To select interactive batch mode, include the `-I` option on the `bsub` command line. Without this option, `bsub` invokes conventional batch mode.

The `INTERACTIVE` parameter in the `lsb.queues` file allows you restrict a queue to *accept only* interactive batch jobs or *exclude all* interactive batch jobs. Use it to restrict Sun HPC–dedicated queues to interactive batch jobs. Otherwise, noninteractive jobs could be added to the queue, which could make the queue less efficient for handling the interactive batch jobs. To impose this restriction, add the following line to the appropriate queue descriptor in the `lsb.queues` file:

```
INTERACTIVE=ONLY
```

All jobs submitted to a queue configured in this way must include the `-I` option on the `bsub` command line.

Note – Separate queues can be configured for batch-mode–only jobs as well.

Because interactive batch jobs need fast response times, there are other steps you should take to minimize job launch latencies normally associated with batch queue behavior. These are described in the next section.

Configuring for Fast Interactive Batch Response Time

This section discusses several steps you can take to optimize the response time of an interactive batch queue.

Set PRIORITY in lsb.queues

The PRIORITY parameter defines a batch queue's priority relative to other batch queues. To ensure faster dispatching, assign a higher PRIORITY value to interactive batch queues than you give to noninteractive queues. A higher number equals a higher priority. For example, the following setting

```
PRIORITY=12
```

means that jobs on that queue will usually be serviced sooner than jobs on queues with a setting of PRIORITY=11 or lower.

Set NICE in lsb.queues

Set the queue's NICE parameter to 10. This ensures that it receives the same CPU priority as other interactive queues.

Set NEW_JOB_SCHED_DELAY in lsb.queues

Set the NEW_JOB_SCHED_DELAY parameter to 0. This allows a new job scheduling session to be started as soon as a job is submitted to this queue.

Add Optimization Parameters to lsb.params

During installation of the Sun HPC ClusterTools packages, you are asked if you want to modify the lsb.params file to optimize interactive batch response time. If you answered yes, the SUNWrte package makes the following changes to the lsb.params file:

```
MBD_SLEEP_TIME=1
```

```
MAX_SBD_FAIL=30
JOB_ACCEPT_INTERVAL=0
```

The first parameter, `MBD_SLEEP_TIME`, specifies the number of seconds LSF Batch will wait between attempts to dispatch jobs. The default is 60 seconds. `SUNWrte` changes the interval to 1 second.

The `MAX_SBD_FAIL` parameter specifies how many times LSF Batch will try to reach an unresponsive slave batch daemon before giving up. `MBD_SLEEP_TIME` controls the frequency of these attempts. If `MAX_SBD_FAIL` is not specified, its default value is three times the `MBD_SLEEP_TIME` value. `SUNWrte` sets `MAX_SBD_FAIL` to 30.

The `JOB_ACCEPT_INTERVAL` parameter specifies how many `MBD_SLEEP_TIME` periods LSF Batch will wait after successfully dispatching a job to a host before it dispatches another job to the same host. `SUNWrte` sets this parameter to `0`, allowing the host to accept multiple jobs in each job dispatching period (`MBD_SLEEP_TIME`).

If you answered `no` during the installation, but now wish to enable these optimizations, simply edit these parameters in the `lsb.params` file as shown above.

Control of Queue Access to Network Interfaces for RSM Communication

In the `hpc.conf` file, the administrator may enable or disable the availability of network interfaces to communications that use the Remote Shared Memory (RSM) protocol. This feature is useful for isolating the traffic that originates from different types of jobs, say, development versus production or benchmark jobs. It is also useful for temporarily configuring out one or more controllers while maintenance is being performed on the network hardware.

Note – Controlling the availability of an interface for RSM communications does not restrict the interface's availability for other purposes, if the interface is enabled for another protocol.

The administrator lists RSM-enabled interfaces in the `PM=rsm` section of the `hpc.conf` file, each with a number appended. The `AVAIL` column may specify 1 or 0, indicating that the interface is or is not available to queues for RSM communication.

Consider this example:

```
#NAME    RANK    AVAIL
Begin PM=rsm
wrsm0   15      0
wrsm1   15      1
wrsm2   15      1
wrsm3   15      0
End PM
```

The interfaces `wrsm1` and `wrsm2` are available for use, whereas interfaces `wrsm0` and `wrsm3` are not available for use.

In addition to supporting numbered instances of an interface, `hpc.conf` also supports the wildcard format (e.g., `wrsm`). The description of a wildcard interface applies to all instances of that interface type except those that are listed explicitly. For example, consider a situation where four network interface cards are present and the `hpc.conf` entry looks like this:

```
#NAME    RANK    AVAIL
Begin PM=rsm
wrsm     15      1
wrsm2    15      0
wrsm3    15      0
End PM
```

The wildcard entry applies to instances 0 and 1, and makes them available for use by RSM. Instances 2 and 3 will be unavailable.

APPENDIX **D**

Creating a Loose Integration With Sun Grid Engine Software

Sun Microsystems makes available a package of scripts and Sun Grid Engine object templates you can use to create a loose integration between the Sun HPC ClusterTools 4 software and the Sun Grid Engine 5.2 software. This package is currently available for downloading :

`http://supportforum.sun.com/gridengine/appnote_hpc.html`

Note – This loose integration package is *not* a supported Sun Microsystems product. It is provided as an unsupported convenience for use on an *as-is* basis.

This appendix describes the procedure for creating the loose integration. It is provided as a supplement to the loose integration package that is currently available at the `supportforum` web site.

The information in this appendix is based on the version of the loose integration package current at the time this manual is published. Because the contents of the package may change with time, and the URL for the download site may change or cease to exist, the loose integration procedure described in this appendix is self-contained. In other words, you can create the same loose integration by following the procedure described here as you would have if you used the scripts available from the download site.

The loose integration procedure consists of the following general tasks:

- Provide a way for Sun Grid Engine to acquire Sun MPI job IDs.
- Create a script for starting Sun MPI jobs. The chief purpose of this script is to create a rank map file that defines the execution resources to be used by Sun MPI jobs.
- Create a script for stopping Sun MPI jobs.
- Create an exclusive parallel environment in which Sun MPI jobs can be submitted to batch queues in isolation from non-MPI batch jobs.

- Create one or more batch queues for the exclusive use of Sun MPI jobs (referred to here as CRE-exclusive queues).
- Create a CRE-exclusive complex for attachment to the CRE-exclusive queues.
- Create suitable suspend, resume, and terminate methods for controlling Sun MPI jobs running on exclusive queues.
- Create one or more batch files for submitting Sun MPI jobs to the CRE-exclusive queues.

These tasks are described more fully in the following sections.

Initial Conditions

The loose integration procedure described in this appendix assumes that the following conditions are met:

- Both Sun HPC ClusterTools 4 and Sun Grid Engine 5.2 software are already installed and their daemons are running.
- The person performing the procedure is experienced with Sun Grid Engine commands and procedures.

Acquiring Sun MPI Job IDs

This section describes the sample script `MPRUN`, which is a wrapper for the CRE command `mprun`. The chief purpose of `MPRUN` is to capture the process ID of `mprun` whenever it starts a Sun MPI job. The following code example illustrates the contents of the `MPRUN` script.

```
#!/bin/sh -f
#
# (c) 2000 Sun Microsystems, Inc.
#
pid_file=$TMPDIR/mprun_pid
#
# Save mprun pid into a temporary file for later use
#
echo "$$" > $pid_file
MPRUN_FLAGS="$MPRUN_FLAGS –Mf $TMPDIR/machines"; export MPRUN_FLAGS

exec /opt/SUNWhpc/bin/mprun $*

/bin/rm $pid_file
```

When `MPRUN` is used to start a Sun MPI job, it saves the `mprun` process ID into the temporary file `$TMPDIR/mprun_pid`. The Sun Grid Engine interface can then use this process ID to acquire the associated MPI job ID, which it needs for controlling the MPI job and its processes.

Note – If `MPRUN` (or an equivalent `mprun` wrapper) is not used to start Sun MPI jobs, Sun Grid Engine commands will be unable to control MPI jobs or their processes.

Two of the script templates provided in this loose integration, `suspend_sunmpi.sh` and `terminate_sunmpi.sh`, illustrate how an MPI job ID can be acquired based on the saved process ID of the `mprun` shell that started the job. These scripts are described in the sections, "Suspending a Sun MPI Job" on page 196 and "Terminating a Sun MPI Job" on page 199.

Creating a Rank Map for Sun HPC ClusterTools

The script template described in this section, `startsunmpi.sh`, creates a rank map file named `machines` and places it in the directory `$TMPDIR`. This script is activated by the parallel environment whenever a batch job is allocated to a CRE-exclusive queue.

The `MPRUN` script should reference `machines` with the `-Mf` *rankmap* option so that any MPI jobs it starts will execute on resources allocated by Sun Grid Engine. This usage is illustrated in the sample batch script, batch.sunmpi.template, which is discussed in "Sample Parallel Environment Batch File" on page 206.

The `startsunmpi.sh` script requires two arguments:

- `$pe_hostfile` – This specifies a file containing a list of the host names and the number of processes to be used for a particular MPI job. This file is automatically generated by Sun Grid Engine, based on the contents of the parallel environment. The Sun Grid Engine software dynamically determines the location of `$pe_hostfile`, depending on where job execution starts.
- *mprun_path* – This is the absolute path of the Sun HPC ClusterTools binaries.

The startsunmpi.sh script parses `$pe_hostfile` and generates the rank map file `machines` in `$TMPDIR`.

At runtime, the Sun Grid Engine software creates `$TMPDIR` for each batch job on the master execution host, which is an execution host where Sun Grid Engine's shepherd process starts. The Sun Grid Engine software determines the absolute path for `$TMPDIR` based on the batch job ID.

Note – The batch job ID is not the same as the CRE job ID.

Once the loose integration is complete, the `startsunmpi.sh` script will be located in `$CODINE_ROOT/mpi`, where `$CODINE_ROOT` is the absolute path for the Sun Grid Engine root directory.

The contents of the `startsunmpi.sh` script are as follows:

```
- startsunmpi.sh: a script to prepare a rank map file for HPC ClusterTools
#!/bin/sh
#
# (c) 2000 Sun Microsystems, Inc.
#
# preparation of the mpi machine file
#
# usage: startsunmpi.sh [options] <pe_hostfile> <mprun_path>
#
#    options are:
#                -catch_hostname
#                 force use of hostname wrapper in $TMPDIR when starting mprun
#                -catch_rsh
#                 force use of rsh wrapper in $TMPDIR when starting mprun
#                -unique
#                 generate a machinefile where each hostname appears only once
#                 This is needed to set up a multithreaded mpi application
#
PeHostfile2MachineFile()
{
if [ "$allocation_rule" = "\$round_robin" ]
then
            cat $1 | nawk ’
                { line[NR] = $0
                }
                END { ncpus=0
                      for (i = 1; i < NR+1; i++) {
                          n = split(line[i], temp, " ")
                          ncpus += temp[2]
                      }
                      for (i = 0; i < ncpus; i++) {
                            irem = i
                            irem %= NR
                            ipos = irem + 1
                             n = split(line[ipos], temp, " ")
                             print temp[1] " 1"
                      }
               }’
```

```
else
   cat $1 | while read line; do
      # echo $line
      host=‘echo $line|cut -f1 -d" "|cut -f1 -d"."‘
      nslots=‘echo $line|cut -f2 -d" "‘
      i=1
      while [ $i -le $nslots ]; do
         # add here code to map regular hostnames into ATM hostnames
         # Add a unit number after hostname for mprun -Mf format
         echo $host 1
         i=‘expr $i + 1‘
      done
   done
fi
}
#
# startup of MPI conforming with the CODINE/GRD
# Parallel Environment interface
#
# on success the job will find a machine-file in $TMPDIR/machines
#
# parse options
catch_rsh=0
catch_hostname=0
unique=0
while [ "$1" != "" ]; do
   case "$1" in
      -catch_rsh)
         catch_rsh=1
         ;;
      -catch_hostname)
         catch_hostname=1
         ;;
      -unique)
         unique=1
         ;;
      *)
         break;
         ;;
   esac
   shift
done
me=‘basename $0‘
```

```
# test number of args
if [ $# -ne 2 ]; then
   echo "$me: got wrong number of arguments" >&2
   exit 1
fi
# get arguments
pe_hostfile=$1
MPIR_HOME=$2
export MPIR_HOME
# ensure job will be able to exec mprun
if [ ! -x $MPIR_HOME/mprun ]; then
   echo "$me: can't execute $MPIR_HOME/mprun" >&2
   exit 1
fi
# ensure pe_hostfile is readable
if [ ! -r $pe_hostfile ]; then
   echo "$me: can't read $pe_hostfile" >&2
   exit 1
fi
# create machine-file
# remove column with number of slots per queue
# mpi does not support them in this form
machines="$TMPDIR/machines"
mype=$TMPDIR/mype
if [ "'isalist | grep sparcv9'" != "" ]; then
         ARCH=solaris64
else
         ARCH=solaris
fi
$CODINE_ROOT/bin/$ARCH/qconf -sp $PE > $mype
allocation_rule='grep allocation $mype | nawk '{print $2}''
if [ $unique = 1 ]; then
   PeHostfile2MachineFile $pe_hostfile | uniq >> $machines
else
   PeHostfile2MachineFile $pe_hostfile >> $machines
fi
#
# Make script wrapper for 'rsh' available in jobs tmp dir
#
if [ $catch_rsh = 1 ]; then
   rsh_wrapper=$CODINE_ROOT/mpi/rsh
   if [ ! -x $rsh_wrapper ]; then
      echo "  $me: can't execute $rsh_wrapper" >&2
     echo "  maybe it resides at a file system not available at this machine" >&2
      exit 1
   fi
```

```
   rshcmd=rsh
   case "$ARC" in
      hp|hp10|hp11) rshcmd=remsh ;;
      *) ;;
   esac
   # note: This could also be done using rcp, ftp or s.th.
   #       else. We use a symbolic link since it is the
   #       cheapest in case of a shared filesystem
   #
   ln -s $rsh_wrapper $TMPDIR/$rshcmd
fi
#
# Make script wrapper for 'hostname' available in jobs tmp dir
#
if [ $catch_hostname = 1 ]; then
   hostname_wrapper=$CODINE_ROOT/mpi/hostname
   if [ ! -x $hostname_wrapper ]; then
      echo " $me: can't execute $hostname_wrapper" >&2
     echo " maybe it resides at a file system not available at this machine" >&2
      exit 1
   fi
   # note: This could also be done using rcp, ftp or s.th.
   #       else. We use a symbolic link since it is the
   #       cheapest in case of a shared filesystem
   #
   ln -s $hostname_wrapper $TMPDIR/hostname
fi
# signal success to caller
exit 0
```

Stopping Sun MPI Jobs

This section describes the script stopsunmpi.sh, which the parallel environment runs after a Sun MPI job completes. This script reads a file called `$TMPDIR/ptree.list` and kills all processes listed in that file. It also removes all temporary files associated with the job.

Note – `ptree.list` is created by `terminate_sunmpi.sh`, a job termination script, which is described in the section, "Terminating a Sun MPI Job" on page 199.

Like the start script, `stopsunmpi.sh` is located in `$CODINE_ROOT/mpi`.

The contents of `stopsunmpi.sh` are as follows.

```
- stopsunmpi.sh: a script to be executed at the end of MPI jobs
#!/bin/sh
#
# (c) 2000 Sun Microsystems, Inc.
#
# shutdown of MPI conforming with the CODINE/GRD
# Parallel Environment interface
#
#-----------------------------------
# KillTree
KillTree()
{
# Kill all the remaining processes
# related to a given CRE Job ID
# using the output generated by "terminate_method"
#
cat $1 | while read cmd; do
    host=‘echo $cmd | nawk ’{print $3}’‘
    plist="‘$cmd | grep -v spmd | nawk ’{print $1}’‘"
    exec rsh -n $host kill -9 $plist
done
}
#
# Just remove machine-file that was written by startmpi.sh
#
rm $TMPDIR/machines
#
# Clean-up any remaining, if any, MPI processes
#
if [ -s $TMPDIR/ptree.list ]; then
   myptree=$TMPDIR/ptree.list
   KillTree $myptree
#
```

```
#  Delete all temporary files
#
   /bin/rm  $myptree
fi
rshcmd=rsh
case "$ARC" in
   hp|hp10|hp11) rshcmd=remsh ;;
   *) ;;
esac
if [ -s $TMPDIR/$rshcmd ]; then
   rm $TMPDIR/$rshcmd
fi
exit 0
```

Suspending a Sun MPI Job

This section describes the script `suspend_sunmpi.sh`, which suspends a Sun MPI job when the parent batch job is suspended.

`suspend_sunmpi.sh` acquires the CRE job ID of the MPI job by accessing the process ID of the `mprun` process in which the job was started. It does this by saving the core image of the `mprun` process to the temporary file `$TMPDIR/mpcr`. Then, it uses the debug utility `adb` to search for the CRE job ID of the corresponding Sun MPI job, which it saves as `cre_jid`.

`suspend_sunmpi.sh` uses the `mpkill` command with `-STOP` to suspend processes belonging to the Sun MPI job specified by `cre_jid`.

The `suspend_sunmpi.sh` script is also located in the `$CODINE_ROOT/mpi` directory.

The contents of the `suspend_sunmpi.sh` script are as follows.

```
- suspend_sunmpi.sh: a script to suspend MPI jobs
#!/bin/sh
#
# (c) 2000 Sun Microsystems, Inc.
#
# Suspend an MPI job when its parent batch job is suspended
#
# In order to get CRE JID, adb is used to look at mprun image.
# A better way to catch an CRE job id should be provided
# to make this suspend method work more robust.
#
# GetCreJid
#
GetCreJid()
{
  mprun_pid=‘cat $TMPDIR/mprun_pid‘
  adb=/usr/bin/adb
  if [ "/bin/ls -l /opt/SUNWhpc/bin | grep HPC4.0" != "" ]; then
#    HPC 4.0: A single mprun (32-bit binary) is provided
     mprun_prog_name=/opt/SUNWhpc/bin/mprun
  else
#    HPC 3.1 release
     if [ "‘isalist | grep sparcv9‘" != "" ]; then
        mprun_prog_name=/opt/SUNWhpc/bin/sparcv9/mprun
     else
        mprun_prog_name=/opt/SUNWhpc/bin/sparcv7/mprun
     fi
  fi
#
# we need to use adb and have a commands file
# we could also keep the file in same dir as this exec
# but not much of a perf loss to keep creating it in $TMPDIR
```

```
#
  adb_comms=$TMPDIR/adb-comms.$$
  if [ "/bin/ls -l /opt/SUNWhpc/bin | grep HPC4.0" != "" ]; then
#    HPC 4.0: A single mprun (32-bit binary) is provided
     /usr/bin/echo "*task+4/D" > $adb_comms
  else
#    HPC 3.1: Two mprun binaries are provided
     if [ "‘isalist | grep sparcv9‘" != "" ]; then
        /usr/bin/echo "*task+8/D" > $adb_comms
     else
        /usr/bin/echo "*task+4/D" > $adb_comms
     fi
  fi
#
# get a core of the process and use adb to get job/task id
# would be nice to reference the variable directly
# through /proc interface
#
  corefile=$TMPDIR/mpcr
  ret=‘/usr/bin/gcore -o $corefile $mprun_pid 2>&1‘
  corefile=$corefile.$mprun_pid
  outputfile=$TMPDIR/adbres.$$
  $adb $mprun_prog_name $corefile < $adb_comms > $outputfile 2>&1
  cre_jid=‘/usr/bin/grep tty_s_orig $outputfile | nawk ’{print $2}’‘
#
# Clean up temporary files
#
  /bin/rm $adb_comms $corefile $outputfile
}

GetCreJid
/opt/SUNWhpc/bin/mpkill -STOP $cre_jid
#
# Save CRE Job ID for resume script
#
/usr/bin/echo $cre_jid > $TMPDIR/resume_cre_jid
#
# signal success to caller
#
exit 0
```

Resuming a Suspended Sun MPI Job

This section describes the script `resume_sunmpi.sh`, which resumes a suspended Sun MPI job when the parent batch job is resumed.

`resume_sunmpi.sh` uses the `cre_jid` that was extracted by the previous execution of `suspend_sunmpi.sh`.

The following shows the contents of the `resume_sunmpi.sh` template.

```
- resume_sunmpi.sh: a script to resume a suspended MPI job. It uses the CRE
  Job Id saved by suspend_sunmpi.sh script.
#!/bin/sh
#
# (c) 2000 Sun Microsystems, Inc.
#
# Resume an MPI job when its parent batch job is resumed
#
# It assumes that CRE JID is saved by suspend method.
#
#----------------
cre_jid=‘cat $TMPDIR/resume_cre_jid‘
#
# Resume the CRE job
#
/opt/SUNWhpc/bin/mpkill -CONT $cre_jid
#
# Delete the CRE job id file
#
/bin/rm $TMPDIR/resume_cre_jid
# signal success to caller
exit 0
```

Terminating a Sun MPI Job

This section describes the script `terminate_sunmpi.sh`, which terminates a Sun MPI job when the parent batch job is terminated.

Like the suspend script, `terminate_sunmpi.sh` extracts the CRE job ID from the core image of the `mprun` process, based on the `mprun` process ID. It then uses `mpkill` to kill the associated processes.

`terminate_sunmpi.sh` also creates the temporary file `$TMPDIR/ptree.list`, which lists all the processes belonging to the MPI job being terminated.

```
- terminate_sunmpi.sh: a script to terminate MPI jobs. It will save all the
  process Ids spawned by an MPI jobs.
#!/bin/sh
#
# (c) 2000 Sun Microsystems, Inc.
#
# Terminate an MPI job when its parent batch job is terminated
#
# Save CRE job info, which will be used by stopsunmpi.sh
#
#-----------------------------------
# GetPList
GetPList()
{
#
# Process "mpps -p" output and identify all the processes
# under a given CRE Job ID (stored at cre_jid)
# Put the output in "rsh" command string.
#
iline=1
cat $1 | while read line; do
    if [ $iline -eq 1 ]
    then
#        skip the first label for JID, the second label for Rank,
#        or the last blank line if applicable
      iline=‘expr $iline + 1‘
    elif [ $iline -eq 3 ]
    then
#        skip the first label for JID, the second label for Rank,
#        or the last blank line if applicable
       iline=‘expr $iline + 1‘
    elif [ $iline -eq 2 ]
    then
#        Pick up JID, NPROC
#         echo pick up $iline -th line for jid and nproc
       jid=‘echo $line|cut -f1 -d" "‘
       nproc=‘echo $line|cut -f2 -d" "‘
       mylines=‘expr $nproc + 3‘
       iline=‘expr $iline + 1‘
    else
```

```
#        Pick up RANK, PID, NODE if JID matches with mine.
       if [ $jid = $cre_jid ]
       then
          if [ $iline -le ‘expr $nproc + 3‘ ]
          then
#               echo pick up $iline -th line for rank and others
             rank=‘echo $line|cut -f1 -d" "‘
             pid=‘echo $line|cut -f2 -d" "‘
             node=‘echo $line|nawk ’{print $4}’‘
#              output pid into file
#              we need to make it more robust
             echo rsh -n $node ptree $pid
#             echo ‘rsh -n $node ptree $pid | grep -v spmd | nawk ’{print $1}’‘
          else
#               echo Done picking $cre_jid, reset $cre_jid
             cre_jid=-1
          fi
       fi
       iline=‘expr $iline + 1‘
       if [ $iline -eq ‘expr $nproc + 5‘ ]
       then
#            set to the beginning of the next set
#            echo skip $iline -th line
          iline=1
       fi
   fi
done
}
#----------------
# GetCreJid
#
GetCreJid()
{
  mprun_pid=‘cat $TMPDIR/mprun_pid‘
  adb=/usr/bin/adb
  if [ "/bin/ls -l /opt/SUNWhpc/bin | grep HPC4.0" != "" ]; then
#    HPC 4.0: A single mprun (32-bit binary) is provided
     mprun_prog_name=/opt/SUNWhpc/bin/mprun
  else
#    HPC 3.1 release
     if [ "‘isalist | grep sparcv9‘" != "" ]; then
        mprun_prog_name=/opt/SUNWhpc/bin/sparcv9/mprun
     else
        mprun_prog_name=/opt/SUNWhpc/bin/sparcv7/mprun
     fi
  fi
```

```
#
# we need to use adb and have a commands file
# we could also keep the file in same dir as this exec
# but not much of a perf loss to keep creating it in $TMPDIR
#
  adb_comms=$TMPDIR/adb-comms.$$
  if [ "/bin/ls -l /opt/SUNWhpc/bin | grep HPC4.0" != "" ]; then
#    HPC 4.0: A single mprun (32-bit binary) is provided
     /usr/bin/echo "*task+4/D" > $adb_comms
  else
#    HPC 3.1: Two mprun binaries are provided
     if [ "‘isalist | grep sparcv9‘" != "" ]; then
        /usr/bin/echo "*task+8/D" > $adb_comms
     else
        /usr/bin/echo "*task+4/D" > $adb_comms
     fi
  fi
#
# get a core of the process and use adb to get job/task id
# would be nice to reference the variable directly
# through /proc interface
#
  corefile=$TMPDIR/mpcr
  ret=‘/usr/bin/gcore -o $corefile $mprun_pid 2>&1‘
  corefile=$corefile.$mprun_pid
  outputfile=$TMPDIR/adbres.$$
  $adb $mprun_prog_name $corefile < $adb_comms > $outputfile 2>&1

  cre_jid=‘/usr/bin/grep tty_s_orig $outputfile | nawk ’{print $2}’‘
#
# Clean up temporary files
#
  /bin/rm $adb_comms $corefile $outputfile
}
#
# Get CRE Job ID
#
GetCreJid
#
# List all my MPI processes
#
mympps=$TMPDIR/mpps.p.out
/opt/SUNWhpc/bin/mpps -p > $mympps
#
```

```
# Create rsh command to get all the process trees
# related to a given CRE Job ID
# This file will be used by stopsunmpi.sh script
# in case there’re any zombie processes
#
myptree=$TMPDIR/ptree.list
GetPList $mympps > $myptree
/bin/rm  $mympps
#
# Use mpkill to kill the MPI job
#
/opt/SUNWhpc/bin/mpkill -KILL $cre_jid
#
# signal success to caller
exit 0
```

Creating a CRE-Exclusive Complex

You can create a CRE-exclusive complex object in the same way you would create any other Sun Grid Engine complex object. That is, you can use `qconf -ac`, which brings up an editor with a default complex template ready to be modified. When you use this command, specify the name for the exclusive complex as an argument to the command. Later, when you quit out of the editor, the template will be saved under that name and will become the CRE-exclusive complex object.

Note – If you want to edit an existing complex, specify the file name of the complex to be edited as a second argument to the `qconf -ac` command.

Edit the template so it contains appropriate attributes for the CRE-exclusive queues. When you quit out of the editor, the template becomes the exclusive complex object, with the name you specified.

Alternatively, you can use the `qmon` graphical user interface to create the complex interactively.

In either case—with `qconf` or `qmon`—use the following example as your model for the CRE-exclusive complex you create.:

```
#name shortcut type  value relop requestable consumable  default

cre   cre      BOOL  true  ==    FORCED      NO          false
```

If you use `qmon` to create the exclusive complex, click on the following buttons in the indicated order: complex_configuration -> Add -> (enter attributes) -> Save -> Ok.

Whenever a batch job is submitted to a CRE-exclusive queue, the name you give the CRE-exclusive complex will be specified on the `qsub` command line, as in the following example:

`%` **`qsub -l cre`** *other-arguments*

This ensures that the job will be assigned to a queue with the necessary attributes.

Creating CRE-Exclusive Queues

To create CRE-exclusive queues, use either `qconf -aq` or `qmon`.

If you choose the command-line interface (`qconf -aq`), specify the name you want the exclusive queue to be known by as an argument to the command. The recommended naming convention is *hostname*`.cre`, where *hostname* is the name of the execution host on which the queue resides.

The `qconf -aq` command brings up an editor with a default queue template ready to be modified. Edit the template so it defines a CRE-exclusive queue. The following example shows the lines of the queue definition template that must be modified to adapt the queue for CRE-exclusive use. In this example, the name of the execution host is `example-1`.

```
qname               example-1.cre
qtype               BATCH INTERACTIVE PARALLEL
suspend_method      $CODINE_ROOT/mpi/suspend_sunmpi.sh
resume_method       $CODINE_ROOT/mpi/resume_sunmpi.sh
terminate_method    $CODINE_ROOT/mpi/terminate_sunmpi.sh
complex_list        cre_exclusive
complex_values      cre-true
```

Note – In this example, `$CODINE_ROOT` is the absolute path for the Sun Grid Engine root directory.

Include this queue name and all other CRE-exclusive queue names on the `queue_list` line of the parallel environment, as described in the next section.

Use the command `qconf -sq` to display a complete queue definition template.

Creating a Parallel Environment

You can set up a parallel environment interactively with the `qconf -ap` command. This command brings up an editor with a default `pe_environment` configuration template ready to be modified.

Edit the default configuration, using the following template as a guide in creating the parallel environment definition file.

```
pe-name                 name
queue_list              list_of_exclusive_cre_queues
slots                   number_of_available_cpus
user_lists              user_access_list
xuser_lists             NONE
start_proc_args         $CODINE_ROOT/mpi/startsunmpi.sh $pe_hostfile mprun_path
stop_proc_args          $CODINE_ROOT/mpi/stopsunmpi.sh
allocation_rule         $round_robin
control_slaves          FALSE
job_is_first_task       FALSE
```

Note – In this example, `$CODINE_ROOT` is the absolute path for the Sun Grid Engine root directory.

For the `pe_name` entry, specify the name you want the parallel environment to be known by. When submitting a Sun Grid Engine job to a CRE-exclusive queue, specify this `pe_name` value for the argument to the `qsub -pe` option.

On the `queue_list line`, enter the names of all the CRE-exclusive queues in the parallel environment. Separate the names with spaces.

On the `slots` line, enter the total number of CPUs in the parallel environment.

On the `user_lists` line, enter the user names of all users who can submit batch jobs to queues in the parallel environment. These names must be in an ACL. This ACL can be created for exclusive use with the parallel environment's CRE queues or it can be an existing, general-purpose ACL.

For the *mprun_path* value, use the absolute path of the Sun HPC ClusterTools binaries.

In the parallel environment example shown above, the `allocation_rule` entry specifies that MPI processes will be distributed to the CPUs in round-robin fashion.

Sample Parallel Environment Batch File

This section describes a template for a Sun Grid Engine batch script that can be used, with editing, to manage job submission to CRE-exclusive queues. The batch script would be located in the `$CODINE_ROOT/mpi` directory along with the start and stop scripts.

The following code example illustrates the contents of this batch template:

```
#!/bin/sh
#
# (c) 2000 Sun Microsystems, Inc.
#
# ----------------------------
# User needs to customize the following items
# enclosed by <>
#
#$ -N <Job_Name>
#$ -S /bin/sh
#$ -o <STD_Output_File>
#$ -e <STD_Error_File>
#$ -M <User@Domain>
#$ -m es
# ----------------------------
#
# Execute the job from  the  current  working  directory
#$ -cwd
#
# Parallel environment request
# ----------------------------
# User needs to customize the following items
# enclosed by <>
#
#$ -l <shortcut_for_cre_exclusive_complex>
#
#      PE_name  CPU_Numbers_requested
#$ -pe <pe_name>     <Num_of_CPUs>
```

```
# --------------------------
#
# All resources are defined here
#
# A sample mprun command
# User must use MPIRUN wrapper to make a loose integration
# Between Sun Grid Engine and HPC ClusterTools working
# In the case of suspending/resuming/terminating batch jobs
#
# --------------------------
# enclosed by <>
#
$CODINE_ROOT/mpi/MPRUN -np $NSLOTS <sleep 120>
#-------------------------
```

The first two sections are not specific to the loose integration. They simply initialize the batch script with conventional job management parameters.

The third section specifies values for two qsub arguments, as follows:

- `-l` *cre_exclusive_shortcut* – Specify the name of the CRE-exclusive complex. When a batch job is submitted for execution on CRE-exclusive queues, `-l` must be used and the name of the exclusive complex must be specified.
- `-pe` *pe_name num_cpus* – For *pe_name*, specify the name of the parallel environment. For *num_cpus*, specify the number of CPUs you want the MPI job to run on.

In the following `qsub` example, the name of the exclusive complex is `cre`, the name of the parallel environment is `hpc`, and the number of CPUs requested is 4.

```
% qsub -l cre -pe hpc 4 mprun_command_line_input
```

The last section shows the `MPRUN` wrapper script being used.

Index

I

J

K

L

M

N

O

P

R

S

T

U

V