Sun HPC ClusterTools 3.0 Administrator's Guide: With CRE

901 San Antonio Road
Palo Alto, , CA 94303-4900
USA 650 960-1300 Fax 650 969-9131

Part No: 806-0295-10
June 1999, Revision A

Contents

Preface

The *Sun HPC ClusterTools 3.0 Administrator's Guide: With CRE* explains how to configure and manage a Sun HPC cluster that is running the Sun HPC Cluster Runtime Environment (CRE) job management software. These instructions are designed for an experienced system administrator with networking knowledge.

Using UNIX Commands

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

See one or more of the following for this information:

- AnswerBook™ online documentation for the Solaris™ 2.x and Solaris™ 7 software environment
- Other software documentation that you received with your system

Typographic Conventions

TABLE P–1 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. Command-line variable; replace with a real name or value.	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be `root` to do this. To delete a file, type `rm` *filename.*

Shell Prompts

TABLE P–2 Shell Prompts

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

Related Documentation

TABLE P–3 Related Documentation

Application	Title	Part Number
All	*Sun HPC ClusterTools 3.0 Read Me First*	805-6281-10
All	*Sun HPC ClusterTools 3.0 Product Notes*	805-6282-10
SCI	Sun HPC SCI Guide	805-6263-10
Installation	Sun HPC *ClusterTools 3.0 Installation Guide*	805-6264-10
Sun MPI Programming	*Sun MPI 4.0 User's Guide: With CRE*	806-0296-10
Sun MPI Programming	*Sun MPI 4.0 Programming and Reference Guide*	805-6269-10
Prism	*Prism 6.0 User's Guide*	805-6277-10
Prism	*Prism 6.0 Reference Manual*	805-6278-10
Sun S3L	*Sun S3L 3.0 Programming and Reference Guide*	805-6275-05

Sun Documentation on the Web

The `docs.sun.com`™ web site enables you to access 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`

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 the CRE. The system administration topics covered in this manual 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 start and stop PFS daemons. 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 will 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 procedure for installing Sun HPC ClusterTools 3.0 from the command line rather that using the graphical user interface supplied with the ClusterTools software.

  Appendix B describes the Cluster Console Manager (CCM), a set of cluster administration tools.

The balance of this chapter provides an overview of the Sun HPC ClusterTools 3.0 software and the Sun HPC cluster hardware on which it runs.

Sun HPC System Hardware

A Sun HPC cluster configuration can range from a single Sun SMP (symmetric multiprocessor) server to a cluster of SMPs connected via any Sun-supported, TCP/IP-capable interconnect.

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

The recommended interconnect technology for clustering Sun HPC servers is the Scalable Coherent Interface (SCI). SCI's bandwidth and latency characteristics make it the preferred choice for the cluster's primary network. An SCI network can be used to create Sun HPC clusters with up to four nodes.

Larger Sun HPC clusters can be built using a Sun-supported, TCP/IP interconnect, such as 100BaseT Ethernet or ATM. The CRE supports parallel jobs running on clusters of up to 64 nodes containing up to 256 CPUs.

Any Sun HPC node that is connected to a disk storage system can be configured as a Parallel File System (PFS) I/O server. PFS file systems are configured by editing the appropriate sections of the system configuration file, `hpc.conf`. See Chapter 7 for details.

The Cluster Runtime Environment

The CRE comprises two sets of daemons—the master daemons and the nodal daemons.

The master daemons consist of the `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.

These two sets of daemons work cooperatively to maintain the state of the cluster and manage program execution. See "Overview of the CRE Daemons" on page 12 for individual descriptions of the CRE daemons.

Sun HPC ClusterTools 3.0 Software

Sun HPC ClusterTools 3.0 Software is an integrated ensemble 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 LSF Suite 3.2.2, Platform Computing Corporation's resource management software, extended with parallel support.

Sun HPC ClusterTools components run under Solaris 2.6 or Solaris 7 (32- or 64-bit).

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 of the MPI 1.2 standard as well as a significant subset of the MPI 2.0 feature list. For example, Sun MPI provides the following features:

- Support for multithreaded programming.
- Seamless use of different network protocols; for example, code compiled on a Sun HPC cluster that has a Scalable Coherent Interface (SCI) network, can be run without change on a cluster that has an ATM network.
- Multiprotocol support such that MPI picks the fastest available medium for each type of connection (such as shared memory, SCI, or ATM).
- Communication via shared memory for fast performance on clusters of SMPs.
- Finely tunable shared memory communication.
- Optimized collectives for symmetric multiprocessors (SMPs).
- Prism support – Users can develop, run, and debug programs in the Prism programming environment.
- MPI I/O support for parallel file I/O.
- Sun MPI is a dynamic library.

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

Parallel File System

Sun HPC ClusterTools'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. This means the time required to read or write a PFS file can be reduced by an amount roughly proportional to the number of file server nodes in the PFS file system.

PFS is optimized for the large files and complex data access patterns that are characteristic of parallel scientific applications.

Prism

Prism is the Sun HPC graphical programming environment. It allows you to develop, execute, debug, and visualize data in message-passing programs. With Prism you can

- Control program execution, such as:
  - Start and stop execution.
  - Set breakpoints and traces.
  - Print values of variables and expressions.
  - Display the call stack.
- Visualize data in various formats.
- Analyze performance of MPI programs.
- Control entire multiprocess parallel jobs, aggregating processes into meaningful groups, called process sets or *psets*.

Prism 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 and provides the following functionality for Sun MPI programmers:

- Vector and dense matrix operations (level 1, 2, 3 Parallel BLAS)
- Iterative solvers for sparse systems
- Matrix-vector multiply for sparse systems
- FFT
- LU factor and solve
- Autocorrelation
- Convolution / deconvolution
- Tridiagonal solvers
- Banded solvers
- Eigensolvers
- Singular value decomposition
- Least squares
- One-dimensional and multidimensional sorts

- Selected ScaLAPACK and BLACS application program interface
- Conversion between ScaLAPACK and S3L
- Matrix transpose
- Random number generators (linear congruential and lagged Fibonacci)
- Random number generator and I/O for sparse systems
- Matrix inverse
- Array copy
- Safety mechanism
- An array syntax interface callable from message-passing programs
- Toolkit functions for operations on distributed data
- Support for the multiple instance paradigm (allowing an operation to be applied concurrently to multiple, disjoint data sets in a single call)
- Thread safety
- Detailed programming examples and support documentation provided online

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

Sun Compilers

The Sun HPC ClusterTools 3.0 release supports the following Sun compilers:

- Sun WorkShop Compilers C/C++ 4.2 (also included in Sun Visual WorkShop C++ 3.0)
- Sun WorkShop Compilers Fortran 4.2 (also included in Sun Performance WorkShop Fortran 3.0)
- Sun Visual WorkShop C++ 5.0
- Sun Performance WorkShop Fortran 5.0

Cluster Console Manager

The Cluster Console Manager is a suite of applications (`cconsole`, `ctelnet`, and `crlogin`) that simplify cluster administration by enabling the administrator to initiate commands on all nodes in the cluster simultaneously. When invoked, the selected Cluster Console Manager application opens a master window and a set of terminal windows, one for each node in the cluster. Any command entered in the master window is broadcast to all the nodes in the cluster. The commands are echoed in the terminal windows, as are messages received from the respective nodes.

Switch Management Agent

The Switch Management Agent (SMA) supports management of the Scalable Coherent Interface (SCI), including SCI session management and various link and switch states.

CHAPTER 2

Getting Started

This chapter describes the basic steps involved in getting a Sun HPC cluster ready for use. It also describes the procedure for shutting down the CRE daemons. The steps covered in this chapter include

- Starting the CRE daemons – “Start the CRE Daemons” on page 7
- Verifying system readiness – “Verify Basic Functionality” on page 8
- Testing MPI communications – “Running Basic MPI Tests” on page 9
- Shutting down the CRE – “Stopping and Restarting the CRE” on page 10

The chapter ends with a brief description of the CRE daemons.

Start the CRE Daemons

If they are not already running, start the CRE daemons on the master node and then on all the other nodes in the cluster.

If you do not know which node in your cluster is the master node, look at the line `MASTER_NODE="`*hostname*`"` in the `hpc_config` file. This file was used in the ClusterTools installation process.

1. On the master node, start the CRE master daemons as root.

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

1. **On all the nodes in the cluster, start the CRE nodal daemons.**

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

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

Verify Basic Functionality

Use the following procedure to test the cluster's ability to perform basic operations.

Run `mpinfo`

Run `mpinfo -N` to display information about the cluster nodes. This step requires `/opt/SUNWhpc/bin` to be in your path.

CODE EXAMPLE 2–1 Sample mpinfo -N Output for a Two-Node System

```
# mpinfo -N
NAME   UP  PARTITION  OS     OSREL  NCPU  FMEM   FSWP   LOAD1   LOAD5   LOAD15
node1 y   -          SunOS  5.6    1     7.17   74.76  0.03    0.04    0.05
node2 y   -          SunOS  5.6    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` entry in the `UP` column:

- Verify that the license daemons are running.
- Restart their nodal daemons as described on "Start the CRE Daemons" on page 7.

Create a Default Partition

You can create a cluster-wide default partition by running an initialization script named `part_initialize` on any node in the cluster. This will create a single partition named `all`, which will include all the nodes in the cluster as members.

Then, run `mpinfo -N` again to verify the successful creation of `all`. See Code Example 2–2 for an example of `mpinfo -N` output when the `all` partition is present.

CODE EXAMPLE 2–2 mpinfo -N Output for the Sample Partition all

```
# /opt/SUNWhpc/bin/part_initialization
# mpinfo -N
NAME     UP  PARTITION  OS     OSREL  NCPU  FMEM   FSWP   LOAD1  LOAD5  LOAD15
node1    y   all        SunOS  5.6    1     8.26   74.68  0.00   0.01   0.03
node2    y   all        SunOS  5.6    1     34.69  38.08  0.00   0.00   0.01
```

Verify That CRE Executes Jobs

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

```
# mprun --Ns --np 0 hostnamenode1
node2
```

`mprun` is the CRE command that launches message-passing jobs. The combination of `--Ns` and `--np 0` ensures that the 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 - Note that the CRE does not sort or rank the output of `mprun` by default, so host name ordering may vary from one run to another.

Running Basic MPI Tests

Verify MPI Communications

You can verify MPI communications by running a simple MPI program. To do so, you must have one of the supported compilers installed on your system. See for more information about supported compilers.

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 `p` 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 Chapter 3 of the *Sun MPI 4.0 User's Guide.*

Stopping and Restarting the 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 the CRE without shutting down the entire cluster, the CRE supports a pair of scripts that simplify this process:

- `sunhpc.cre_master` – Use this command to stop or start the CRE master daemons as described on "Start the CRE Daemons" on page 7 and "Stopping and Restarting the CRE" on page 10.
- `sunhpc.cre_node` – Use this command to stop or start the CRE nodal daemons as described on "Start the CRE Daemons" on page 7 and "Stopping and Restarting the CRE" on page 10.

To Shut Down the CRE Without Shutting Down Solaris

To shut down only the CRE daemons, execute the following commands as root:

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

   Note that 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.

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

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

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

Be sure to stop the nodal daemons before stopping the master daemons. Otherwise, if you shut the master node down before shutting down the rest of the nodes, the CCM tools will not be available and you will have to shut down each node individually.

`sunhpc.cre_node` and `sunhpc.cre_master` behave differently when executing a `stop` command:

- `sunhpc.cre_master` stops all processes that have been spawned by the CRE before killing the daemons.
- `sunhpc.cre_node` does not kill user processes. Consequently, client daemons can be restarted without affecting running jobs.

To Restart the CRE Without Rebooting Solaris

Execute the following commands on the master node:

1. **Start the master daemons.**

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

1. **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 the 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 restarting the CRE daemons.

Overview of the CRE Daemons

The sections on "The Role of `tm.rdb`" on page 12 through "The Role of `tm.watchd`" on page 12 provide brief descriptions of the CRE master daemons: `tm.rdb`, `tm.mpmd`, and `tm.watchd`. The sections on "The Role of `tm.omd`" on page 13 describe the nodal daemons, `tm.omd`, and `tm.spmd`.

The Role of `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 the CRE. This database represents the state of the cluster and the jobs running on it.

When changes are made to the cluster configuration, the `tm.rdb` daemon must be restarted to update the database to reflect the new conditions. For example, if a node is added to a partition, `tm.rdb` must be restarted to implement these changes.

The Role of `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.

The Role of `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 of 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.

The Role of `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 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 the CRE starts up.

The Role of `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`.

CHAPTER 3

Cluster Administration: A Primer

The Sun HPC cluster's default configuration will support execution of MPI applications. In other words, if you have started the CRE daemons on your cluster) and created the default partition `all` (as described in the previous chapter), you can begin executing MPI jobs on the cluster.

You may, however, want to customize your cluster's configuration to make it better suited to the specific administration and use requirements of your site. For example, you may want to create other partitions of different sizes containing different sets of nodes as members to match a variety of job execution needs. You may also want to create one or more PFS file systems to handle the more demanding disk storage requirements that MPI applications often have.

This chapter provides an overview of the principal ways in which you can control your cluster's configuration and behavior. There are three distinct types of control you can use to manage your cluster:

- Environment variables – See "CRE Environment Variables" on page 16.
- Administration interface (`mpadmin`) – See "`mpadmin`: Administration Interface" on page 17 for an introduction to the `mpadmin` command interface and instructions for using it to perform basic administration tasks. See Chapter 6 for a more comprehensive discussion of mpadmin.
- Cluster configuration file (`hpc.conf`) – See "`hpc.conf`: Cluster Configuration File" on page 26 for a brief description of the `hpc.conf` file and Chapter 7 for instructions on how to modify it.

CRE Environment Variables

You can use the following environment variables to specify default values for various features of your Sun HPC cluster.

- SUNHPC_CLUSTER
- SUNHPC_CONFIG_DIR
- SUNHPC_PART

These are described in “SUNHPC_CLUSTER ” on page 16, “SUNHPC_CONFIG_DIR ” on page 16, and “SUNHPC_PART ” on page 16, respectively.

SUNHPC_CLUSTER

SUNHPC_CLUSTER specifies the name of the default Sun HPC cluster. This is the cluster to which users will automatically be connected—unless a different cluster is chosen using the `mpadmin --s` option, as described in “`mpadmin` Syntax” on page 17.

You can find out the name of your cluster by running `mpinfo --C`, which displays information about the cluster, including its name.

Note - The name of a cluster is always the host name of the cluster’s master node—that is, the node on which the master daemons are running.

SUNHPC_CONFIG_DIR

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

SUNHPC_PART

SUNHPC_PART specifies the name of the default partition. This is the partition on which user’s jobs will be executed unless a different partition is selected via the `mprun --p` option. This option is discussed in the *Sun HPC Cluster Runtime Environment 1.0 User’s Guide*.

mpadmin: Administration Interface

The CRE provides an interactive command interface, mpadmin, which you can use to administer your Sun HPC cluster. It must be invoked by root.

This section explains how to use mpadmin to perform the following administrative tasks:

- List the names of all nodes in the cluster – see "List Names of Nodes" on page 21
- Enable nodes – see "Enabling Nodes" on page 22
- Create and enable partitions – see "Creating and Enabling Partitions" on page 23
- Customize some aspects of cluster administration – see "Customizing Cluster Administration" on page 25

These descriptions are preceded by an introduction of the mpadmin command interface, which is presented in "Introduction to mpadmin" on page 17.

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

Introduction to mpadmin

mpadmin Syntax

The mpadmin command has the following syntax.

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

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

When you invoke mpadmin with any of the other options (--f, --q, or --s), it performs the specified operation and then displays an mpadmin prompt, indicating that it is in the interactive mode. In this mode, you can execute any number of mpadmin commands until you quit the interactive session.

When you invoke mpadmin without any options, it goes immediately into the interactive mode, displaying an mpadmin prompt.

The mpadmin command-line options are summarized in Table 3–1 and more fully following the table.

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.
`--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.

`--c` *command – Single Command Option*

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"
#
```

Note - 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.

`--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

`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
- The net work interfaces used by the nodes

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. For example, a node's `enabled` attribute can be

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

Note - The CRE sets many attributes in a cluster to default values each time it boots up. Except for attribute modifications described here and in Chapter 6, *do not* change attribute values.

`mpadmin` *Contexts*

`mpadmin` commands are organized into four *contexts*, which correspond to the four 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.
- Network – These commands affect network interface attributes.
- Partition – These commands affect partition attributes.

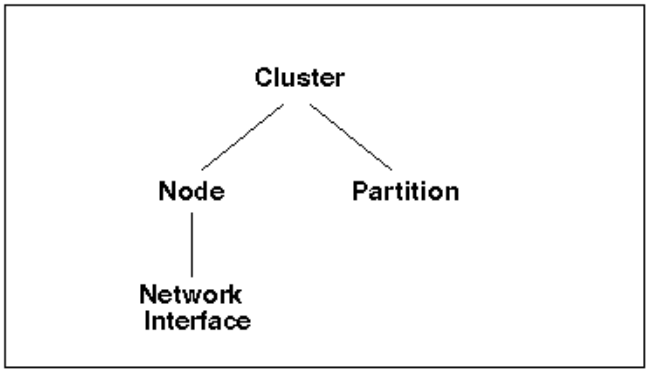

Figure 3–1 `mpadmin` Contexts

`mpadmin` Prompts

In the 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.
[*cluster-name*]N(*node-name*) Network::	Current context = Network, but not a specific network interface.
[*cluster-name*]N(*node-name*) I(*net-if-name*)::	Current context = a specific network interface.

Note - When the prompt indicates a specific network interface, it uses `I` as the abbreviation for `Network` to avoid being confused with the `Node` abbreviation `N`.

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 the 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 the previous chapter), the cluster will contain no partitions at this stage. 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 be run on it. To enable a node, 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[n
```

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[nod
```

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. The next time the CRE starts its master daemons, it will add the names of any newly created partitions to its resource database.
- 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.

Once a partition is created and enabled, you can run serial or parallel jobs on it. A serial program will run on a single node of the partition. Parallel programs will be distributed to as many nodes of the partition as the CRE determines to be appropriate for the job. Job placement on a partition's nodes is discussed in the *Sun MPI 4.0 User's Guide: With CRE*.

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[pa
    node0
    node1
[node0] P[part0]::
```

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: 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.

If both partitions were enabled at the same time and you tried to run a job on either, the CRE would fail and return an error message. When you want to use `part1`, you will need to disable `part0` first.

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.

Shared vs. Dedicated Partitions

The CRE can configure a partition to allow multiple MPI jobs to be running on it concurrently. Such partitions are referred to as *shared* partitions. The 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 ma
```

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 job can be running at a time. It is set to 4 on part1 to allow up to four jobs to be started on that partition.

Note again, that the context-changing shortcut (introduced in “Enabling Nodes” on page 22) is used in the second and fourth lines of this example.

Customizing Cluster Administration

There are two cluster attributes that you may be interested in modifying, 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

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

```
[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 you to the shell level. For example:

```
[node0]:: quitnode1#
```

hpc.conf: Cluster Configuration File

When the 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 six sections, which are summarized below and illustrated in Code Example 3–1.

- The `ShmemResource` section specifies the maximum amount of shared memory and swap space that jobs can allocate.
- The `Netif` section lists and ranks all network interfaces to which Sun HPC nodes may be connected.
- The `MPIOptions` section defines various MPI parameters that can affect the communication performance of MPI jobs.
- The `PFSFileSystem` section names and defines PFS file systems in the cluster.
- The `PFSServers` section names and defines I/O servers for the PFS file systems.
- The `HPCNodes` section is not used by the CRE. It applies only in an LSF-based runtime environment.

You can change any of these aspects of your cluster's configuration by editing the corresponding parts of the `hpc.conf` file. This section explains how to:

- Prepare for editing `hpc.conf`. See "Prepare to Edit `hpc.conf`" on page 27"Prepare to Edit `hpc.conf`" on page 27.
- Create one or more I/O servers for the PFS file systems. See "Create PFS I/O Servers" on page 28
- Create PFS file systems. See "Create PFS File Systems" on page 29.
- Specify various attributes of the network interfaces that your cluster nodes use. See "Set Up Network Interfaces" on page 30.
- Learn how to control MPI communication attributes. See "Specify MPI Options" on page 33.
- Update the CRE database. See "Update the CRE Database" on page 34.

Note - You may never need to make any other changes to `hpc.conf` than are described in this section. However, if you do want to edit `hpc.conf` further, see Chapter 7 for a fuller description of this file.

CODE EXAMPLE 3–1 General Organization of hpc.conf File

```
Begin ShmemResource
  :
End ShmemResource

Begin Netif
NAME    RANK    MTU    STRIPE    PROTOCOL    LATENCY    BANDWIDTH
  :      :       :        :         :           :           :
End Netif

Begin MPIOptions queue=hpc
  :
Begin HPCNodes
  :
End HPCNodes

Begin PFSFileSystem=pfs1
NODE            DEVICE            THREADS
  :               :                 :
End PFSFileSystem

Begin PFSServers
NODE            BUFFER_SIZE
  :               :                 End PFSServers

Begin HPCNodes

End HPCNodes
```

Prepare to Edit `hpc.conf`

Perform the steps described in “Stop the CRE Daemons” on page 27 and “Copy the `hpc.conf` Template” on page 28.

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.

Note - You can use one of the CCM tools (`cconsole`, `ctelnet`, or `crlogin`) to broadcast the nodal stop command to all the nodes from a single command entered on 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
```

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 3.0 distribution includes an `hpc.conf` template, which is stored, by default, in `/opt/SUNWhpc/examples/cre/hpc.conf.template`.

Copy the template from its installed location to `/opt/SUNWhpc/conf/hpc.conf` and edit it as described in “Create PFS I/O Servers” on page 28.

When you have finished editing `hpc.conf`, perform the steps described in“Update the CRE Database” on page 34 to update the CRE database with the new configuration information.

Create 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`. Code Example 3–2 shows a sample `PFSServers` section that includes three PFS I/O server nodes.

CODE EXAMPLE 3–2 PFSServers Section Example

```
Begin PFSServers
NODE            BUFFER_SIZE
hpc-node0        150
hpc-node1        150
hpc-node2        300
hpc-node3        300
End PFSServers
```

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

The second column specifies 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 number of buffers that you should specify will depend on the amount of I/O traffic you expect that server is likely to experience at any given time.

The optimal buffer size will vary with system type and load. Buffer sizes in the range of 128 to 512 provide reasonable performance on most Sun HPC Systems.

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.

Create 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. See "Parallel File System Name" on page 30.
- The hostname of each server node in the parallel file system. See "Server Node Hostnames" on page 30.
- The name of the storage device to be included in the parallel file system being defined. See "Storage Device Names" on page 30.
- The number of PFS I/O threads spawned to support each PFS storage device. See "Thread Limits" on page 30.

The following example shows sample `PFSFileSystem` sections for two parallel file systems, `pfs0` and `pfs1`.

CODE EXAMPLE 3–3 PFSFileSystem Section Example

```
Begin PFSFileSystem=pfs-demo0
NODE            DEVICE               THREADS
hpc-node0       /dev/rdsk/c0t1d0s2  1 hpc-node1       /dev/rdsk/
c0t1d0s2  1
hpc-node2       /dev/rdsk/c0t1d0s2  1
End PFSFileSystem

Begin PFSFileSystem=pfs-demo1
NODE            DEVICE               THREADS
hpc-node3       /dev/rdsk/c0t1d0s2  1
hpc-node4       /dev/rdsk/c0t1d0s2  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 file as are used for serial Solaris files.

Server Node Hostnames

The NODE column lists the hostnames of the nodes that function as I/O servers for the parallel file system being defined. The example configuration in Code Example 3–3 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

In the THREADS column, specify 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 will depend 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.

Set Up Network Interfaces

Edit the Netif section to specify various characteristics of the network interfaces that are used by the nodes in the cluster. Code Example 3–4 illustrates the default Netif section that is in hpc.conf.template. This section discusses the various network interface attributes that are defined in the Netif section.

CODE EXAMPLE 3–4 `Netif Section` Example

```
Begin Netif
NAME            RANK     MTU       STRIPE     PROTOCOL     LATENCY     BANDWIDTH
midnn           0        16384     0          tcp          20          150
idn             10       16384     0          tcp          20          150 scin          20      32768     1          t
  :             :          :       :           :           :           :
scid            40       32768     1          tcp          20          150
  :             :          :       :           :           :           :
scirsm          45       32768     1          rsm          20          150
  :             :          :       :           :           :           :
  :             :          :       :           :           :           :
smc             220      4096      0          tcp          20          150
End Netif
```

Interface Names

Add to the first column the names of the network interfaces that are used in your cluster. The supplied `Netif` section contains an extensive list of commonly used interface types to simplify this task.

By convention, network interface names include a trailing number as a way to distinguish multiple interfaces of the same type. For example, if your cluster includes two 100 Mbit/second Ethernet networks, include the names `hme0` and `hme1` in the `Netif` section.

Rank Attribute

Decide the order in which you want the networks in your cluster to be preferred for use and then edit the `RANK` column entries to implement that order.

Network preference is based on the relative value of a network interface's ranking, with higher preference being given to interfaces with lower rank values. In other words, an interface with a rank of 10 will be selected for use over interfaces with ranks of 11 or higher, but interfaces with ranks of 9 or less will have a higher preference.

Note - These ranking values are relative; their absolute values have no significance. This is why gaps are left in the default rankings, so that if a new interface is added, it can be given an unused rank value without having to change any existing values.

Decisions about how to rank two or more dissimilar network types are usually based on site-specific conditions and requirements. Ordinarily, a cluster's fastest network is given preferential ranking over slower networks. However, raw network bandwidth

is only one consideration. For example, an administrator might decide to dedicate a network that offers very low latency, but not the fastest bandwidth to all intra-cluster communication and use a higher-capacity network for connecting the cluster to systems outside the cluster.

MTU Attribute

This is a placeholder column. Its contents are not used at this time.

Stripe Attribute

If your cluster includes an SCI (Scalable Coherent Interface) network, you can implement scalable communication between cluster nodes by striping MPI messages over the SCI interfaces. In striped communication, a message is split into smaller packets and transmitted in two or more parallel streams over a set of network interfaces that have been logically combined into a *stripe-group*.

The `STRIPE` column allows the administrator to include individual SCI network interfaces in a *stripe-group pool*. Members of this pool are available to be included in logical stripe groups. These stripe groups are formed on an as-needed basis, selecting interfaces from this stripe-group pool.

To include the SCI interface in a stripe-group pool, set its `STRIPE` value to `1`. To exclude an interface from the pool, specify `0`. Up to four SCI network interface cards per node can be configured for stripe-group membership.

When a message is submitted for transmission over the SCI network, an MPI protocol module distributes the message over as many SCI network interfaces as are available.

Stripe-group membership is made optional so you can reserve some SCI network bandwidth for non-striped use. To do so, simply set `STRIPE = 0` on the SCI network interface(s) you wish to reserve in this way.

Protocol Attribute

This column identifies the communication protocol used by the interface. The `scirsm` interface employs the RSM (Remote Shared Memory) protocol. The others in the default list all use TCP (Transmission Control Protocol).

If you add a network interface of a type not represented in the `hpc.conf` template, you will need to specify the type of protocol the new interface uses.

Latency Attribute

This is a placeholder column. Its contents are not used at this time.

Bandwidth Attribute

This is a placeholder column. Its contents are not used at this time.

Specify 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 two templates with predefined option settings. These templates are shown in Code Example 3–5 and are discussed below.

General-Purpose, Multiuser Template – The first template in the `MPIOptions` section is designed for general-purpose use at times when multiple message-passing jobs will be running concurrently.

Performance Template – The second template 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 each template contains the phrase "`Queue=`*xxxx*." This is because the queue-based LSF workload manager runtime environment also uses this `hpc.conf` file.

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 you can see what options are most beneficial when operating in a multiuser mode.

CODE EXAMPLE 3–5 `MPIOptions` Section Example

```
# Following is an example of the options that affect the runtime
# environment of the MPI library. The listings below are identical
# to the default settings of the library. The "queue=hpc" phrase
# makes it 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_numpostbox       15
# rsm_shortmsgsize    401
# rsm_maxstripe         2
```

(continued)

(Continuation)

```
# 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` template, do the following:

- Delete the "`Queue=performance`" phrase from the `Begin MPIOptions` line.
- Delete the comment character (#) from the beginning of each line of the performance template, including the `Begin MPIOptions` and `End MPIOptions` lines.

The resulting template should appear as follows:

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

Update 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.

CHAPTER 4

PFS Operations

This chapter applies only if you are implementing Sun PFS file systems.

A PFS I/O daemon must be running on each PFS I/O server, and a PFS proxy daemon must be running on each node that will access PFS file systems. The procedures for starting and stopping these daemons are described in the sections “Starting PFS I/O Daemons” on page 35 through “Stopping PFS Proxy Daemons” on page 37. The section “PFS Node or I/O Daemon Failures” on page 37 discusses PFS I/O failures.

Starting PFS I/O Daemons

PFS requires an I/O daemon to be running on each I/O server node. These daemons start automatically when node(s) on which they reside boot. However, if you want to start an I/O daemon on a newly created I/O server without rebooting, start the daemon manually on that node.

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

Once the daemons are running, execute the `mkfs` and `mount` commands in each parallel file system. For example, if `cities` is to be a 512-Mbyte file system.

```
# mkfs cities 512M# /opt/SUNhpc/bin/pfsmount cities
```

Alternatively, after executing the `mkfs` command, do the following on every node in the cluster.

```
# mount cities
```

The parallel file system `cities` is now ready to use.

Note - Once `mkfs` has been run on a file system, You should not attempt to change its configuration attributes in the CRE database. Doing so could result in the loss of file system contents.

Starting PFS Proxy Daemons

PFS also requires a proxy daemon to be running on each client node. Like the I/O daemons running on the I/O servers, PFS proxy daemons start automatically. If you want to start them manually (to avoid rebooting I/O client nodes), do the following on each node that requires it.

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

Alternatively, you can manually start both the PFS client and server daemons on every node in the cluster by executing the following command on any one node in the cluster.

```
# /opt/SUNWhpc/etc/pfs/pfsstart
```

Stopping PFS I/O Daemons

Stopping PFS is a two-step process.

Unmount each PFS file system. For example, to unmount the PFS file system `cities`, enter

```
# /opt/SUNWhpc/bin/pfsumount cities
```

on any node in the cluster. Alternatively, enter

```
# umount cities
```

on each client node.

Once you've unmounted the PFS file systems of interest, stop the I/O daemons on each I/O server. You do this by running the `sunhpc.pfs_server stop` command on each I/O server node. For example:

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

Note - PFS file systems are automatically unmounted and PFS I/O daemons are automatically stopped when a node is shut down or rebooted.

Stopping PFS Proxy Daemons

Shutting down client proxy daemons is similar to the proxy daemon startup process described in "Starting PFS Proxy Daemons" on page 36. For example, to manually stop a PFS proxy daemon a individual nodes, enter

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

Alternatively, you can manually stop both the PFS client and I/O server daemons on every node in the cluster by executing the following command on any one node in the cluster.

```
# /opt/SUNWhpc/etc/pfs/pfsstop
```

PFS Node or I/O Daemon Failures

If a PFS I/O daemon or PFS I/O node crashes, data stored in local memory buffers will be lost. This lost data may leave any mounted file systems in an inconsistent state. In such cases, use the `fsck` utility to restore file system consistency so they are safe to use.

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

Configuring a Sun SMP or cluster of SMPs for Sun HPC Software use 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 be 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 be 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.

For example, at a minimum, a Sun Ultra HPC 2 (two-processor) system should have 256 MBytes of memory.

Generally speaking, 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 3.0 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 the 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, the CRE provides enough configuration flexibility to allow the administrator to avoid many of these problems.

The following sections discuss some of the factors that should be considered 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 communications 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

The CRE uses the cluster's default network interconnect to perform communication between the daemons that perform resource management functions. The 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. The 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 the 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 will use the network that the CRE instructs it to use, which can be set by the system administrator. In general, the 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 will share bandwidth with administrative and CRE functions. This will result 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. The administrator 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 debugger is used to tune, debug and visualize Sun MPI programs running within the cluster. As Prism itself is a parallel program, starting it will generate the same sort of CRE traffic that invocation of other application generates.

Once Prism 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 Prism itself and is routed over the default network along with all other administrative traffic. In general, the amount of traffic generated by Prism 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 will do so. The administrator needs 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 will generate traffic on the Sun MPI network, while most other types of I/O operations will involve only the default network.

Applications that use parallel I/O to read from and write to PFS file systems will use the network specified by the CRE. In a one-network cluster, this means that parallel I/O traffic will be 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, the administrator would normally configure the CRE so that this type of parallel I/O would be routed over the parallel application network. A Sun HPC cluster can be configured to allow parallel I/O traffic to be routed by itself over a dedicated third network if that amount of traffic segregation is desired.

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 can often 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 SCI, 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 will account 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 some as 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 SCI, 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.

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 will depend 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, such as VxFS, 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 all data in 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 (such as Sun Enterprise Volume Manager or RAID 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 today's Sun SMP nodes, this assumption is less likely to 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, of 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 sixteen-process job is scheduled on four SMP nodes, then the limiting factor will be the four network adaptors 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 45.

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 sixteen-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 FastEthernet, the bandwidth out of any node is limited to 100 Mbits/s.

A single SPARC Storage Array (SSA) can generate more than twice that bandwidth. Since the network is effectively half the bandwidth of the node, adding a second SSA to the node will not lead to any improvement in performance. On the other hand, adding an SSA to a node that is not currently being used as a PFS server may well boost the overall PFS performance.

CHAPTER 6

mpadmin: Detailed Description

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

- mpadmin syntax, the subcommands it supports, and other aspects of mpadmin functionality
- how to use mpadmin in performing 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 --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. See “Command-Line Options” on page 48 for descriptions of the mpadmin command-line options.

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. See “mpadmin Command Overview” on page 51 for a description of the interactive mpadmin mode.

Note - For the rest of this discussion, `mpadmin` subcommands will be referred to as `mpadmin` commands or simply as commands.

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. See
`--f` *file-name*	Take input from specified file.
`--h`	Display help / usage text.
`--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.

`--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 51 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 will be 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.
- The net work interfaces used by the nodes.

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 - The 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 four *contexts*, which correspond to the four types of mpadmin objects. These contexts are illustrated in Figure 6–1 and summarized below.

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

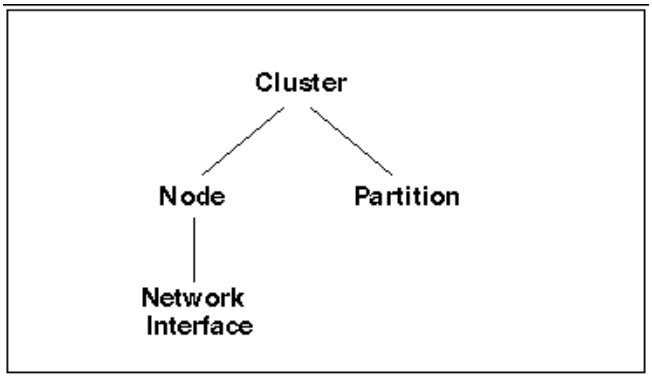

Figure 6–1 The mpadmin Contexts

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

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.
[*cluster-name*]N(*node-name*) Network::	Current context = Network Interface, but not a specific network interface.
[*cluster-name*]N(*node-name*) I(*net-if-name*)::	Current context = a specific network interface.

Note - When the prompt indicates a specific network interface, it uses `I` as the abbreviation for `Network Interface` to avoid being confused with the `Node` abbreviation `N`.

`mpadmin` Command Overview

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, partitions, network interfaces); see “Configuration Control” on page 52.
- Attribute control – These commands are used to set and reset attribute values; see “Attribute Control” on page 53.

- Context navigation – These commands are used to change the current context to a different context; see “Context Navigation” on page 55.
- Information retrieval – These commands are used to display object and attribute information; see “Information Retrieval” on page 59.
- Miscellaneous – See “Miscellaneous Commands” on page 62.

Configuration Control

A Sun HPC cluster contains one or more named partitions. Each partition contains some number of specific nodes. Likewise, each node includes one or more network interfaces that it uses for internode communication.

The CRE automatically creates the cluster, node, and network interface objects based on the contents of the `hpc.conf` file. Partitions are the only kind of object that the system administrator is required to create and manage.

Use the `delete` command to remove partitions, but no other types of cluster objects. Your remove nodes and network interfaces from a Sun HPC cluster by editing the `hpc.conf` file.

`create`

Usage:

```
:: create object-name
```

Available In:

Node, Partition, Network

The `create` command creates a new object with the name *object-name* and makes the new object the current context.

Note, 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, Network

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 - Although you *can* use the `set` and `unset` commands to change any cluster attribute, the 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, Network

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 node=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 node=+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` will ignore the value assignment and will simply consider the attribute to be active.

---

unset

Usage:

```
:: unset attribute
```

Available In:

Cluster, Node, Partition, Network

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.

Example:

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

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

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` will list all the nodes in the cluster. If `list` is invoked in the Partition context, it will list 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, Network

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, node, or network interface, it must have been created by the 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, Network

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, Network

The `up` command moves you up one level from the current context. The following example moves from the Network context to the context of node `node2`.

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

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, Network

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

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

`network`

Usage:

:: network

Available In:

Node

The `network` command moves you from a specific Node context to the Network context associated with that node.

[node0]
N[node2]:: **network**[node0] N[node2] Network::

Information Retrieval

This set of commands displays 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 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 don't 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, Network

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, Network

The `show` command displays the current state of the attributes of the specified object *object-name*. 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 the object whose attributes you want to see is in the current context, you don't have to specify its name. For example:

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

Miscellaneous Commands

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. See "`--s` *cluster-name* – Connect to Specified Cluster" on page 49 for a description of the `--s` option.

Note - When the 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.

If, for some reason, you want to use the `connect` command, see the following example. It shows the 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, Network

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 part1Enabling part0 and part1
[node0]::
```

help

Usage:

```
:: help [command]
```

Available In:

Cluster, Node, Partition, Network

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[node
```

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

If you specify a context command (`node`, `partition`, or `network`), `mpadmin` lists the commands available within that context. Note that you can specify `network` as an argument to `help` only at the node level.

```
[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 and its network

   interfaces
set <attribute>[=value]   set the current
node's attribute
unset <attribute>   delete the current node's attribute
network   enter the network interface command mode
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[node
```

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

`quit`/`exit`

Usage:

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

Available In:

Cluster, Node, Partition, Network

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 are input.

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 consists of the following sections:

- “Introductory Notes” on page 67
- “Log In to the Cluster” on page 67
- “Customizing Cluster-Level Attributes” on page 69

Introductory Notes

This section contains information about various `mpadmin` topics that you will find useful when reading about cluster administration tasks in later sections.

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 86.

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 the CRE.

Separate Name Spaces

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

Log In to 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 that is not the case, you must be connected to the target cluster through one of the following alternative methods:

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 describedbelow.

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).
`lock_max_age`	Value	Specifies the maximum amount of time a lock can remain set (the value is in seconds)

`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 Cluster Runtime Environment 1.0 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`, the 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 will fail.

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 will fail.
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 into any partition. If so, execute the program in that partition.
4. Check to see if the user is logged into a partition. Execute the program in that partition.
5. If none of these checks yield a partition name, check for the existence of the default_interactive_partition attribute. If it specifies a partition, execute the program in that partition.

The SUNHPC_PART environment variable is described in "CRE Environment Variables" on page 16. The MPRUN_FLAGS environment variable is described in the *Sun MPI 4.0 User's Guide: With CRE*.

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 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.

---

administrator

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

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

Note the use of double quotes.

lock_max_age

The CRE uses locks for internal purposes. The `lock_max_age` attribute specifies the length of time that the CRE will wait before removing a lock. For example, to set the maximum lock interval to two minutes, enter the following:

```
[node0]:: set lock_max_age="2 minutes"
```

The default is 10 minutes.

Nodes and Network Interfaces

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 the CRE automatically or are controlled via partition-level attributes.

There are no administrative actions required by network interface attributes. They are all controlled by the CRE. The only actions you might want to take with respect to network interfaces is to list them or display their attribute values.

Node Commands

Table 6–5 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.

TABLE 6–5 Node-Level `mpadmin` Commands *(continued)*

Command	Synopsis
`list`	List all the defined nodes.
`show` [*node*]	Show a node's attributes.
`dump` [*node*]	Show the attributes of the node and its network interfaces.
`set` *attribute*[=*value*]	Set the specified attribute of the current node.
`unset` *attribute*	Delete the specified attribute of the current node.
`network`	Move to the network interface command level.
`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, the `enabled` and `max_total_procs` are the only node attributes that you can safely modify. See "`enabled` " on page 75 and "`max_total_procs`" on page 76 for details.

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_swap`	Value	Percent of time CPU spent waiting for swap.
`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).
`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, for example, `SunOS`.
`os_release`	Value	Operating system's release number, for example, 5.5.1
`os_release_maj`	Value	Operating system's major release number, for example, 5.

TABLE 6–6 Node Attributes That Cannot Be Set by the System Administrator *(continued)*

Attribute	Kind	Description
os_release_min	Value	Operating system's minor release number, for example, 5 or 6.
os_version	Value	Operating system's version, for example, GENERIC.
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.
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).
name	Value	Name of the node; this is predefined and must not be set via mpadmin.

TABLE 6–7 Node Attributes That Can Be Set by the System Administrator *(continued)*

Attribute	Kind	Description
`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 the CRE daemons are start up 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. The 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 will be able to 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.

Solaris 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)::
```

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

name

A node's name is predefined by the `hpc.conf` file. You must not change it by setting this attribute.

partition

Note - There is no need to set this attribute. The CRE sets it automatically if the node is included in any partition configuration(s). See "Creating Partitions" on page 79 for additional details.

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 will be able to 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.

```
[node0]
N(node1):: set shmem_minfree=0.2
```

`shmem_minfree` must be set to a value between 0.0 and 1.0. When `shmem_minfree` is unset, it defaults to 0.1.

This attribute can be set on both nodes and partitions. If both 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, you should first

- 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 81 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 Cluster Runtime Environment 1.0 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::
```

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.

The CRE performs load balancing on shared partitions. When you use `mprun` to execute a program on a shared partition, the 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.

TABLE 6–8 Partition-Level `mpadmin` Commands *(continued)*

Command	Synopsis
`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*.

Creating Partitions

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.

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. “Naming Partitions and Custom Attributes” on page 67 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. See “Configuring Partitions” on page 80 for details.

Configuring Partitions

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

TABLE 6–9 Common Partitions and Their Attributes

Partition Type	Relevant Attributes	Recommended Value
Login	no_logins	not set
Login	max_total_procs	not set or set greater than
Dedicated	max_total_procs no_logins	=1 set
Serial	no_mp_tasks	set
Parallel	no_mp_task	not set

You can combine the attributes listed in Table 6–10 in any way that makes sense for your site. See “Configuring Partitions” on page 80 for suggestions about how to configure your partitions.

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

Note - You should not change these partition attributes. See "max_locked_mem and min_unlocked_mem" on page 75 for a description of their effects.

---

max_total_procs

To limit the number of simultaneously running mprun processes allowed on all nodes in a partition or in all partitions, set the max_total_procs attribute in a specific node context or in the general 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. By default, max_total_procs is unset.

The 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 "Naming Partitions and Custom Attributes" on page 67 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_tasks

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

```
[node0]
P(part1):: set no_mp_tasks[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=node1mpadmin: node1 must
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.

```
[node0]
P(part1):: set shmem_minfree=0.2
```

`shmem_minfree` must be set to a value between 0.0 and 1.0. When `shmem_minfree` is unset, it defaults to 0.1.

This attribute can be set 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.

Prerequisite

Before enabling a partition, you must disable any partitions that share nodes with the partition that you are about to enable.

Setting `enabled`

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 enabledmpadmin: 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 are running on a partition when it is disabled will continue to run. After disabling a partition, you should either wait for ay running jobs to terminate or stop them using the `mpkill` command. This is described in the *Sun HPC Cluster Runtime Environment 1.0 User's Guide.*

Deleting Partitions

Delete a partition when you don't 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 Cluster Tools 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.

See “Naming Partitions and Custom Attributes” on page 67 for restrictions on attribute names.

CHAPTER 7

hpc.conf: Detailed Description

This chapter discusses the Sun HPC 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/conf`.

`hpc.conf` is organized into six sections, which are summarized below and illustrated in Code Example 7–1.

- The `ShmemResource` section defines certain shared memory attributes. See "`ShmemResource` Section" on page 90 for details.
- The `Netif` section lists and ranks all network interfaces to which Sun HPC nodes are connected. See "`Netif` Section" on page 92 for details.
- The `MPIOptions` section allows the administrator to control certain MPI parameters by setting them in the `hpc.conf` file. See "`MPIOptions` Section" on page 94 for details.
- The `PFSFileSystem` section names and defines all parallel file systems in the Sun HPC cluster. See "`PFSFileSystem` Section" on page 99 for details.
- The `PFSServers` section names and defines all parallel file system servers in the Sun HPC cluster. See "`PFSServers` Section" on page 101 for details.
- The `HPCNodes` section is not used by the CRE. It applies only in an LSF-based runtime environment.

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.

Sun HPC ClusterTools 3.0 software is distributed with an `hpc.conf` template, which is installed by default in `/opt/SUNWhpc/examples/cre/hpc.conf.template`. You should copy this file to `/opt/SUNWhpc/conf/hpc.conf` and edit it to suit your site's specific configuration requirements.

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 "To Shut Down the CRE Without Shutting Down Solaris" on page 10. If you change `PFSFileSystem` or `PFSServers` sections, you must also unmount any PFS file systems first.

CODE EXAMPLE 7–1 General Organization of hpc.conf File

```
Begin ShmemResource
  :     End ShmemResource
Begin Netif
NAME    RANK    MTU    STRIPE    PROTOCOL    LATENCY    BANDWIDTH
  :      :       :        :         :           :           :
End Netif
Begin MPIOptions queue=hpc
  :
End MPIQOptions

Begin HPCNodes
  :
End HPCNodes

Begin PFSFileSystem=pfs1
NODE            DEVICE            THREADS
  :               :                 :
End PFSFileSystem

Begin PFSServers
NODE            BUFFER_SIZE
  :               :                 End PFSServers

Begin HPCNodes

End HPCNodes
```

`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.

Code Example 7–2 shows the `ShmemResource` template that is in the `hpc.conf` file that is shipped with Sun HPC ClusterTools 3.0 software.

CODE EXAMPLE 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.

The following section explains how to set these limits.

Guidelines for Setting Limits

Sun HPC's 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 will be 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 will be 90% of available physical memory.

The following Sun HPC ClusterTools components use shared memory:

- The 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 the CRE when it starts up and is not affected by `MaxAllocMem` and `MaxAllocSwap` settings. This ensures that the 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. The amount of shared memory allocated by a job can be controlled by MPI environment variables.
- 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 will allow 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 will reduce the risk of having a single application lock up physical memory. How much below 50% you choose to set it will depend 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.

`Netif` Section

The `Netif` section identifies the network interfaces supported by the Sun HPC cluster and specifies the rank, and striping attributes for each interface. The `hpc.conf` template that is supplied with Sun HPC ClusterTools 3.0 software contains a default list of supported network interfaces as well as their default ranking. Code Example 7–3 represents a portion of the default `Netif` section.

CODE EXAMPLE 7–3 Netif Section Example

```
Begin Netif
NAME          RANK     MTU       STRIPE     PROTOCOL     LATENCY WIDTH
midnn         0        16384     0          tcp          20          150
idn           10       16384     0          tcp          20          150
scin          20       32768     1          tcp          20          150
  :           :          :       :           :           :           :
scid          40       32768     1          tcp          20          150
  :           :          :       :           :           :           :
scirsm        45       32768     1          rsm          20          150
  :           :          :       :           :           :           :
```

(continued)

(Continuation)

```
  :           :         :       :          :          :           :
smc           220       4096    0          tcp        20          150End Netif
```

Interface Names

The first column lists the names of possible network interface types.

Rank Attribute

The rank of an interface is the order in which that interface is to be preferred over other interfaces. 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 cluster intra-cluster communication and use a higher-capacity network for connecting the cluster to other systems.

MTU Attribute

This is a placeholder column. Its contents are not used at this time.

Stripe Attribute

Sun HPC ClusterTools 3.0 software supports scalable communication between cluster nodes through striping of MPI messages over SCI interfaces. In striped communication, a message is split into smaller packets and transmitted in two or more parallel streams over a set of network interfaces that have been logically combined into a *stripe-group*.

The STRIPE column allows the administrator to include individual SCI network interfaces in a *stripe-group pool*. Members of this pool are available to be included in logical stripe groups. These stripe groups are formed on an as-needed basis, selecting interfaces from this stripe-group pool.

To include the SCI interface in a stripe-group pool, set its STRIPE value to 1. To exclude an interface from the pool, specify 0. Up to four SCI network interface cards per node can be configured for stripe-group membership.

When a message is submitted for transmission over the SCI network, an MPI protocol module distributes the message over as many SCI network interfaces as are available.

Stripe-group membership is made optional so you can reserve some SCI network bandwidth for non-striped use. To do so, simply set STRIPE = 0 on the SCI network interface(s) you wish to reserve in this way.

Protocol Attribute

This column identifies the communication protocol used by the interface. The scirsm interface employs the RSM (Remote Shared Memory) protocol. The others all use TCP (Transmission Control Protocol).

Latency Attribute

This is a placeholder column. Its contents are not used at this time.

Bandwidth Attribute

This is a placeholder column. Its contents are not used at this time.

MPIOptions Section

Overview

This section provides a set of options that control MPI communication behavior in ways that are likely to affect message-passing performance. It contains two templates with predefined option settings. These templates are shown in Code Example 7–4 and discussed below.

General-Purpose, Multiuser Template – The first template in the `MPIOptions` section is designed for general-purpose use at times when multiple message-passing jobs will be running concurrently.

Performance Template – The second template 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 each template contains the phrase "`Queue=`*xxxx*." This is because the queue-based LSF workload management runtime environment uses the same `hpc.conf` file as the CRE.

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 you can see what options are most beneficial when operating in a multiuser mode.

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

- Delete the "`Queue=performance`" phrase from the `Begin MPIOptions` line.
- Delete the comment character (#) from the beginning of each line of the performance template, including the `Begin MPIOptions` and `End MPIOptions` lines.

The resulting template should appear as follows:

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

Table 7–1 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 7–1 names the environment variable

CODE EXAMPLE 7–4 `MPIOptions` Section Example

```
# Following is an example of the options that affect the runtime
# environment of the MPI library. The listings below are identical
# to the default settings of the library. The "queue=hpc" phrase
# makes it 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.
```

(Continuation)

```
#
# Begin MPIOptions queue=hpc
# coscheduling  avail
# pbind         avail
# spindtimeout   1000
# progressadjust   on
# spin            off
#
# shm_numpostbox       16
# shm_shortmsgsize    256
# rsm_numpostbox       15
# rsm_shortmsgsize    401
# rsm_maxstripe         2
# 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 7–1 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`.

TABLE 7–1 MPI Runtime Options *(continued)*

Option	Values		Description
	Default	**Other**	
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.
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.

TABLE 7–1 MPI Runtime Options *(continued)*

Option	Values		Description
	Default	**Other**	
		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 value specifying the desired number.
`rsm_shortmsgsize`	401		Sets to 401 the maximum number of bytes a short message can contain. 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.
`rsm_maxstripe`	2		Sets to 2 the maximum number of stripes that can be used. This equals the value to which the environment variable `MPI_RSM_MAXSTRIPE` is set.

TABLE 7–1 MPI Runtime Options *(continued)*

Option	Values		Description
	Default	**Other**	
		integer	To change the maximum number of stripes that can be used, enter an integer specifying the desired limit. This value cannot be greater than 2.
`spin`	`off`		Sets the MPI library to avoid spinning while waiting for status. This equals the value to which the environment variable `MPI_SPIN` is set.
		`on`	Sets the MPI library to spin.

`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 hostname of each server node in the parallel file system.
- The name of the storage device to be included in the parallel file system being defined.
- The number of PFS I/O threads spawned to support each PFS storage device.

A separate `PFSFileSystem` section is needed for each parallel file system that you want to create. Code Example 7–5 shows a sample `PFSFileSystem` section with two parallel file systems, `pfs0` and `pfs1`.

CODE EXAMPLE 7–5 PFSFileSystem Section Example

```
Begin PFSFileSystem=pfs-demo0
NODE            DEVICE              THREADS
hpc-node0       /dev/rdsk/c0t1d0s2  1
hpc-node1       /dev/rdsk/c0t1d0s2  1
hpc-node2       /dev/rdsk/c0t1d0s2  1
End PFSFileSystem
```

(continued)

(Continuation)

```
Begin PFSFileSystem=pfs-demo1
NODE            DEVICE              THREADS
hpc-node3       /dev/rdsk/c0t1d0s2  1
hpc-node4       /dev/rdsk/c0t1d0s2  1
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 Hostnames

The `NODE` column lists the hostnames of the nodes that function as I/O servers for the parallel file system being defined. The example configuration in Code Example 7–5 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 hostname `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 second column gives the device name associated with each member node. This name follows Solaris device naming conventions.

Thread Limits

The `THREADS` column allows the administrator to specify how many 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 will depend 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.

PFSServers Section

A PFS I/O server is a Sun HPC node that is

- Listed in the PFSServers section of hpc.conf, as shown in Code Example 7–6.
- Connected to one or more disk storage units that are listed in a PFSFileSystem section of the hpc.conf file.

CODE EXAMPLE 7–6 PFSServers Section Example

```
Begin PFSServers
NODE            BUFFER_SIZE
hpc-node0        150
hpc-node1        150
hpc-node2        300
hpc-node3        300
End PFSServers
```

In addition to being defined in hpc.conf, a PFS I/O server also differs from other nodes in a Sun HPC cluster in that it has a PFS I/O daemon running on it.

PFS I/O Server Hostnames

The left column lists the hostnames of the nodes that are PFS I/O servers. In this example, they are ios0 through node2 and node3.

Buffer Size

The second column specifies the amount of memory the PFS I/O daemon will have for buffering transfer data. This value is specified in units of 32-Kbyte buffers. The number of buffers that you specify will depend on the amount of I/O traffic you expect that server is likely to experience at any given time.

The optimal buffer size will vary with system type and load. Buffer sizes in the range of 128 to 512 provide reasonable performance on most Sun HPC Systems. You can use pfsstat to get reports on buffer cache hit rates. This can be useful for evaluating how well suited the buffer size is to the cluster's current I/O activity.

HPCNodes Section

This section is used only in a cluster that is using LSF as its workload manager, not CRE. The CRE ignores the HPCNodes section of the hpc.conf file.

Propagate 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
```

If PFS file systems were unmounted and PFS I/O daemons were stopped, restart the I/O daemons and remount the PFS file systems .

CHAPTER 8

Troubleshooting

System administrators can control some conditions that can cause errors by performing periodic maintenance. Examples of such preventive maintenance are described in “Cleaning Up Defunct CRE Jobs” on page 103. “Diagnostics” on page 106 describes procedures for troubleshooting various kinds of problems. “Error Conditions and Troubleshooting Tips” on page 107 discusses various error conditions and troubleshooting tips. Finally, “Procedures for Recovery” on page 110 describes a procedure for recovering the CRE database when a system failure occurs.

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` states.

- A Prism *main* window that won't 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 the 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
```

1. **If necessary, remove the job object from the CRE database.**

   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 root, from the master node.

```
# mpkill --C jid
```

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, the CRE will handle 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 the 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 all defunct jobs, data about the jobs may remain in the CRE database. As root from the master node, use `mpkill` –C to remove this residual data.

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 and CPU cycles.

Symptoms of orphaned processes can be detected by examining error log files or `stdout`, if you're 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`.

Diagnostics

The following sections describe Solaris diagnostics that may be useful in troubleshooting various types of error conditions.

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
```

will test (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 the 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 3 seconds.

Use `/usr/bin/iostat` to display disk and system usage. For example:

```
% iostat --c 2
```

displays percentage utilizations every 2 seconds.

Use `/usr/bin/vmstat` to generate additional information about the virtual memory system. For example:

```
% vmstat --S 5
```

reports on swapping activity every 5 seconds.

It can be useful to run these diagnostics periodically, monitoring their output for multiple intervals.

Error Conditions and Troubleshooting Tips

The following sections include sample error messages and their interpretations, as well as guidelines for anticipating common problems.

Error Messages

- The following error message usually indicates that all the nodes in an CRE partition are marked down—that is, their node daemons are not running.

```
No nodes in partition satisfy RRS:
```

This could happen, for example, if CRE was unable to check out its licenses. This message can also indicate an error in the construction of an RRS.

- Under certain circumstances, when a user attempts to kill a job, the 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
```

If these can be correlated to jobs being killed, these errors 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.

Troubleshooting Tips

- When running multiprocess jobs (`--np` equal to 0 or greater than 1) in an 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 will be 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 will reduce 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, find out which processes are use `mpkill --9` *jid* to kill such a job.

- The 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.

- *RPC timeouts* – 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, you should 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 the 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 won't 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 system 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 the behavior of your system suggests that you've run out of swap space (after executing `vmstat` or `df` on `/tmp`), you may need to increase the limit on the CRE's `shmem_minfree` attribute.

- 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.

Procedures for Recovery

Re-creating the CRE Database

The `rte.master reboot` and `rte.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 rebootthe CRE:

1. **Run `rte.master reboot` on the master node.**

```
# /etc/init.d/rte.master reboot
```

1. **Run `rte.node reboot` on all the nodes (including the master node if it's running `tm.spmd` and `tm.omd`).**

```
# /etc/init.d/rte.node reboot
```

The procedure will attempt 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 of the nodes in the cluster simultaneously. For information about the Cluster Console Manager applications, see Appendix A "Cluster Management Tools".

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`. `rdb-log` is a running log of the resource database activity and `rdb-save` is a snapshot of the database taken at regular intervals.

APPENDIX **A**

Installing and Removing the Software

This appendix includes instructions for

- *Installing the software at the command line* – “Installing at the Command Line” on page 111
- *Removing the software* – “Removing the Software” on page 127
- *Removing and installing individual packages* – “Removing and Reinstalling Individual Packages” on page 129

Installing at the Command Line

The easiest way to configure and install Sun HPC ClusterTools 3.0 software is to use the configuration tool, `install_gui`, as described in the *Sun HPC ClusterTools 3.0 Installation Guide*. If you prefer, however, you may install the software from the command line as described in this appendix, with a few references to the installation guide.

Figure A–1 summarizes the steps involved. The solid lines identify tasks that are always performed. The dashed lines indicate special-case tasks.

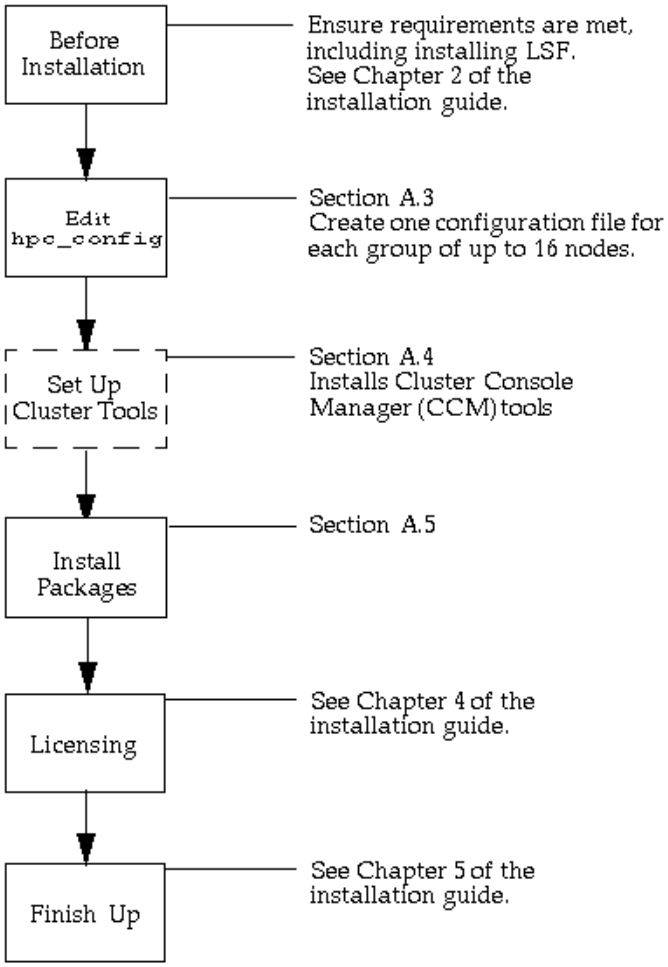

Figure A–1 Installing Sun HPC ClusterTools 3.0 Software at the Command Line

Before Installation

Before installing Sun HPC ClusterTools 3.0 software, you need to ensure that the hardware and software that make up your cluster meet certain requirements. You must have already installed LSF 3.2.3. Further requirements are outlined in the *Sun HPC ClusterTools 3.0 Installation Guide*. Review them before proceeding with the instructions in this appendix.

If you are installing the software on a cluster of more than 16 nodes, you will probably want to use the CCM tools to make installation easier. You can use these tools to install on up to 16 nodes at a time. If you need to install software on a cluster of more than 16 nodes, you must install it first on a group of up to 16 nodes, then add more nodes by repeating the installation process on additional groups of up to 16 until you have installed the software on all the nodes in the cluster. For each group of nodes, you will need to create a separate configuration file, each with a unique file name, such as `hpc_config1`, `hpc_config2`, and so on.

The `hpc_config` File

Many aspects of the Sun HPC ClusterTools 3.0 installation process are controlled by a configuration file called `hpc_config`, which is similar to the `lsf_config` file used to install LSF 3.2.3.

Instructions for accessing and editing `hpc_config` are provided in “Accessing `hpc_config`” on page 113 and “Editing `hpc_config`” on page 114.

Accessing `hpc_config`

Use a text editor to edit the `hpc_config` file directly. This file must be located in a directory within a file system that is mounted read / write / execute accessible on all the other nodes in the cluster. A template for `hpc_config` is provided on the Sun HPC ClusterTools 3.0 distribution CD-ROM to simplify creation of this file.

Before starting the installation process, you should copy this template to a directory on the node chosen to be the installation platform and edit it so that it satisfies your site-specific installation requirements. Choose a node to function as the installation platform and a home directory on that node for `hpc_config`.

Note - The directory containing `hpc_config` must be read / write / execute accessible (777 permissions) by all the nodes in the cluster.

The `hpc_config` template is located in

```
/cdrom/hpc_3_0_ct/Product/Install_Utilities/config_dir/hpc_config
```

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 distribution CD-ROM in the CD-ROM drawer.**

3. **Copy the configuration template onto the node.**

```
# cd config_dir_install
# cp /cdrom/hpc_3_0_ct/Product/Install_Utilities/config_dir/hpc_config .
```

config_dir_install is a variable representing the directory where the configuration files will reside; all cluster nodes must be able to read from and write to this directory.

4. **Edit the** `hpc_config` **file according to the instructions provided in the next section.**

If You Have Already Installed the Software

If you have already installed the software, you can find a copy of the `hpc_config` template in the directory `/opt/SUNWhpc/bin/Install_Utililites/config_dir`.

If you are editing an existing `hpc_config` file after installing the software using the graphical installation tool, the `hpc_config` file created by the tool will not contain the comment lines included in the template.

Editing `hpc_config`

Code Example A–1 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 following Code Example A–1. Two examples of edited `hpc_config` files follow the general description of the template.

The template comprises five sections:

- *Supported Software Installation* – All installations must complete this first section. Since you will be using LSF, complete only Part A of this section.
- *General installation information* – All installations must complete this section. If you are installing the software locally on a single-node cluster, you can stop after completing this section.
- *Information for NFS and cluster-local installations* – If you are installing the software either on an NFS server for remote mounting or locally on each node of a multinode cluster, you need to complete this section, too.

CODE EXAMPLE A–1 hpc_config Template (With Most Comment Lines Removed)

```
Section I - Supported Software Information

LSF_SUPPORT="<choice>"

# PART A: Running HPC 3.0 ClusterTools software with LSF software.

# Do you want to modify LSF parameters to optimize HPC job launches?

MODIFY_LSF_PARAM="<choice>"

# Name of the LSF Cluster
LSF_CLUSTER_NAME="<clustername>"

# 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_3_0_ct"

#Section III - For Cluster-Local and NFS Installation

# Installation Method
INSTALL_METHOD="<method>"

# Hardware Information
NODES="<hostname1> <hostname2> <hostname3>"

# SCI Support
INSTALL_SCI="<choice>"

# Section IV - For NFS Installation

# NFS Server NFS_SERVER=""

# Location of the Software Installed on the NFS ServerINSTALL_LOC_SERVER=""
```

- *Information for NFS installations* – You need to complete this section only if you are installing the software on an NFS server.

For the purposes of initial installation, ignore the fifth section.

Supported Software Installation

LSF Support

You will be using the software with LSF, so enter `yes` here.

```
LSF_SUPPORT="yes"
```

Since you will be using LSF, complete only Part A of this section.

LSF Parameter Modification

Allowing the Sun HPC installation script to modify LSF parameters optimizes HPC job launches. Your choice for this variable must be `yes` or `no`.

```
MODIFY_LSF_PARAM="choice"
```

Name of the LSF Cluster

Before installing Sun HPC ClusterTools software, you must have installed LSF 3.2.3. When you installed the LSF software, you selected a name for the LSF cluster. Enter this name in the `LSF_CLUSTER_NAME` field.

```
LSF_CLUSTER_NAME="clustername"
```

General Installation Information

All installations must complete this section. If you are installing the software locally on a single-node cluster, you can stop after completing this section.

Type of Installation

Three types of installation are possible for Sun HPC ClusterTools 3.0 software:

- `nfs` – Install the software on an NFS server for remote mounting.
- `smp--local` – For single-node clusters only: Install the software locally on the node.
- `cluster--local` – For multinode clusters only: Install the 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.

- *Local Installations* – For local installations (`smp--local` or `cluster--local`), `INSTALL_LOC` is the path where the packages will actually be installed.
- *NFS installations* – 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`. The location must have set (or mounted, if this is an NFS installation) read / write (755) permission 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 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_3_0_ct`. For example:

```
CD_MOUNT_PT="/cdrom/hpc_3_0_ct"
```

Information for NFS and Cluster-Local Installations

If you are installing the software either on an NFS server for remote mounting or locally on each node of a multinode cluster, you need to complete this section.

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 greatly facilitate the installation of Sun HPC ClusterTools 3.0 software on all of the nodes in your cluster in parallel. See Appendix B for information about using CCM tools.

Note - You can use the CCM tools to install on up to 16 nodes at a time. For clusters with more than 16 nodes, you will have to repeat the installation process on groups of up to 16 nodes at a time until you have installed the software on the entire cluster.

- `rsh` – If you choose the `rsh` method, the software will be installed serially on the cluster notes 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 this method requires that all nodes are 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  ..."
```

Every node in your Sun HPC cluster must also be in the corresponding LSF cluster. See the discussion of the `lsf.cluster.`*clustername* configuration file in the *LSF Batch Administrator's Guide* for information on LSF clusters.

Note - If you will not be using the CCM tools, you can allow the installation script to derive the node list from the LSF configuration file `lsf.cluster.`*clustername*. To do this, either set the `NODES` variable to `NULL` or leave the line commented out. You must be installing from one of the nodes in the LSF cluster.

SCI Support

This section tells the script whether to install the SCI-related packages. If your cluster includes SCI, replace *choice* with `yes`; otherwise, replace it with `no`.

```
INSTALL_SCI="yes"
```

A `yes` entry causes the three SCI packages and two RSM packages to be installed in the `/opt` directory. A `no` causes the installation script to skip the SCI and RSM packages.

Note - The SCI and RSM packages are installed locally on every node, not on an NFS server.

Information for NFS Installations Only

You need to 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 software—it must be included in the `NODES` field described in “Hardware Information” on page 118. 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

If you want to install the software on the NFS server in the same directory as the one specified in `INSTALL_LOC`, leave `INSTALL_LOC_SERVER` empty (`""`). If you prefer,

you can override INSTALL_LOC by specifying an alternative 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.

Sample hpc_config Files

Code Example A–2 and Code Example A–3 illustrate the general descriptions in the preceding sections with edited hpc_config files representing two different types of installations.

Local Install – Code Example A–2 shows how the file would be edited for a *local* installation on every node in a cluster. The main characteristics of the installation illustrated by Code Example A–2 are summarized below:

- Section 1:
  - The software will be used with LSF.

CODE EXAMPLE A–2 Sample Completed hpc_config File—Local Install via rsh

```
Section I - Supported Software Information

LSF_SUPPORT="yes"

# PART A: Running the software with LSF

# Do you want to modify LSF parameters to optimize HPC job
launches?
MODIFY_LSF_PARAM="yes"

# Name of the LSF Cluster
LSF_CLUSTER_NAME="italy"

#
Section II - General Installation Information

# Type of Installation Configuration
INSTALL_CONFIG="cluster-local"

# Installation Location
INSTALL_LOC="/export/home/opt2"

# CD-ROM Mount Point
CD_MOUNT_PT="/cdrom/hpc_3_0_ct"

#
```

(continued)

(Continuation)

```
Section III - For Cluster-Local and NFS Installation

# Installation Method
INSTALL_METHOD="rsh"

# Hardware Information
NODES="napoli pisa milano"

# SCI Support
INSTALL_SCI="no"#Section IV - For NFS Installation

# NFS Server
NFS_SERVER=""

# Location of the Software Installed on the NFS Server
INSTALL_LOC_SERVER=""
```

- The installation script will modify LSF parameters to optimize HPC launches.
- The set of clustered nodes that will be running LSF jobs is named `italy`.
- Section 2:
  - The packages will be installed locally. This means every node in the system will contain a copy of the packages that make up Sun HPC ClusterTools 3.0 software.
  - The base directory where the software will be installed is `/export/home/opt2/`.
  - The mount point for the software CD-ROM is `/cdrom/hpc_3_0_ct`.
- Section 3:
  - The software will be installed using the UNIX utility `rsh`.
  - The nodes are not connected to a terminal concentrator, so only the node host names are listed in the `Hardware Information` section.
  - The cluster in this example does not include SCI hardware.
- Section 4 is *not* completed, as the software is being installed locally on every node.

For the purposes of initial installation, ignore the fifth section.

NFS Install – Code Example A–3 shows an `hpc_config` file for an *NFS* installation. The main features of this installation example are summarized below:

- Section 1:
  - The software will be used with LSF.
  - The installation script will modify LSF parameters to optimize HPC launches.
  - The set of clustered nodes that will be running LSF jobs is named `italy`.
- Section 2:
  - The packages are being installed on an NFS server.
  - The mount point for the software on the NFS clients is `/opt/`.
  - The mount point for the software CD-ROM is `/cdrom/hpc_3_0_ct`.
- Section 3:
  - The software will be installed using Cluster Console Manager tools.
  - The nodes are connected to a terminal concentrator, and the cluster console facility will be used. Consequently, each host name must be part of a triplet entry that also includes the name of the terminal concentrator and the ID of the terminal concentrator port to which the node is connected.

This example shows the nodes `venice`, `napoli`, and `pisa` all connected to the terminal concentrator `rome` via ports 5002, 5003, and 5004.

- The cluster in this example does not include SCI hardware.

CODE EXAMPLE A–3 Sample Completed hpc_config File—NFS Install via cluster–tool

```
Section I - Supported Software Information

LSF_SUPPORT="yes"

# PART A: Running the software with LSF

# Do you want to modify LSF parameters to optimize HPC job launches?
MODIFY_LSF_PARAM="yes"

# Name of the LSF Cluster
LSF_CLUSTER_NAME="italy"

#
Section II - General Installation Information
# Type of Installation Configuration
INSTALL_CONFIG="nfs"
```

(continued)

(Continuation)

```
# Installation Location
INSTALL_LOC="/opt"

# CD-ROM Mount Point
CD_MOUNT_PT="/cdrom/hpc_3_0_ct"

#
Section III - For Cluster-Local and NFS Installation

# Installation Method
INSTALL_METHOD="cluster--tool"

# Hardware Information
NODES="venice/rome/5002 napoli/rome/5003 pisa/rome/5004"

# SCI Support
INSTALL_SCI="no"

#
Section IV - For NFS Installation

# NFS Server
NFS_SERVER="mars/rome/5005"

# Location of the Software Installed on the NFS Server
INSTALL_LOC_SERVER="/export/home/opt2"
```

- Section 4:
  - The host name of the NFS server must be supplied (in this example, `mars`). Because the NFS server is connected to a terminal concentrator, its host name must also be part of a triplet entry analogous to the entries in `NODES`.

In this case, the NFS server is not one of the nodes in the Sun HPC cluster. All the nodes in the cluster must be able to communicate with it over a network.

- The software will be installed on `mars` in the directory `/export/home/opt2`.

For the purposes of initial installation, ignore the fifth section.

Run cluster_tool_setup

Note - You can use the CCM tools to install on up to 16 nodes at a time. For clusters with more than 16 nodes, you will have to repeat the installation process on groups of up to 16 nodes at a time until you have installed the software on the entire cluster.

This step is optional. If you have chosen the `cluster--tool` method of installation 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 machine and creates a cluster configuration file that is used by CCM applications. See Appendix B 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 other two, `ctelnet` and `crlogin`, do not.

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 if from a cluster node, it will be disabled when the node from which it is launched reboots.

Perform the following steps, as root, to run `cluster_tool_setup`.

1. **Go to the `Product` directory on the Sun HPC ClusterTools 3.0 distribution CD-ROM.**

   Note that this directory must be mounted on (accessible by) all nodes in the cluster.

   ```
   # cd /cdrom/hpc_3_0_ct/Product/Install_Utilities
   ```

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, plus a trailing `--f` tag.

```
# ./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
```

(This example uses C-shell syntax.)

4. **Invoke the CCM tool of your choice: `cconsole` (if the nodes are connected to a terminal concentrator), `ctelnet`, or `crlogin`.**

All three tools reside in `/opt/SUNWcluster/bin`. For example,

```
# /opt/SUNWcluster/bin/ctelnet clustername
```

where *clustername* is the name of the LSF cluster. 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 remove previous release packages, as described in “Removing and Reinstalling Individual Packages” on page 129, or to install the software packages, as described in “Installing Software Packages” on page 125.”

Installing Software Packages

This section describes the procedure for installing the Sun HPC ClusterTools packages. Note that the exact procedure for each step will depend 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 B 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 its configuration file as part of the package installation. If the installation process stops prematurely—if, for example, 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.**

2. **Go to the Product directory on the Sun HPC ClusterTools 3.0 distribution CD-ROM.**

   Note, this directory must be mounted on (accessible by) all nodes in the cluster.

   ```
   # cd /cdrom/hpc_3_0_ct/Product/Install_Utilities
   ```

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.

   Note - If the `hpc_config` file contains an `INSTALL_SCI="yes"` entry, `hpc_install` will install the three SCI software packages along with the other Sun HPC ClusterTools packages. When the SCI packages are installed, the installation script will display a message telling you to reboot the nodes. Ignore this message. You must reboot the nodes only after any SCI interface cards are configured. If the system does not include SCI hardware, the nodes do not need to be rebooted.

Removing the Software

To remove LSF, see the documentation that came with the software.

The easiest way to remove Sun HPC ClusterTools 3.0 software is by using the configuration tool, `install_gui`. See the next section for details. If you prefer to remove the software at the command line, you may do so using the provided removal scripts. See “Removing the Software: Command Line” on page 128 for instructions.

Removing the Software: Configuration Tool

1. **Locate a configuration file or files for the cluster.**

   To remove the software from your cluster, you will need a configuration file that describes the cluster. Ideally you should use the configuration file you created when installing the software. If you cannot locate that file, you will have to create one. You can use the configuration tool to create the file. (See Chapter 3 of the *Sun HPC ClusterTools 3.0 Installation Guide*.)

   Note - The configuration tool will remove the software from up to 16 nodes at once. If you need to remove software from a cluster of more than 16 nodes, you must remove it first from a group of up to 16 of the nodes in your cluster. Then remove from more nodes by repeating the removal process on additional groups of nodes until you have removed the software from all the nodes in the cluster. The procedure is similar to installing the software on a cluster of more than 16 nodes. See Section 3.1.2 of the installation guide for more information.

   ---

2. **Load the Sun HPC ClusterTools 3.0 CD-ROM in the CD-ROM drawer.**

   The CD-ROM mount point must be mounted on all the nodes in the cluster.

3. **Enable root login access.**

   By default, most systems allow logins by root only on their console devices. To enable root login access during software removal, you must edit the `/etc/default/login` file on each node in the cluster. In this file on each node, find this line:

   ```
   CONSOLE=/dev/console
   ```

and make it into a comment by adding a # before it:

```
#CONSOLE=/dev/console
```

After removing the software, you should disable root login access again if your site's security guidelines require it.

1. **As root, launch the `install_gui` tool with the configuration file.**

   You can load the configuration file either from the command line or from within the tool after it has been launched.

   - At the command line, launch the configuration tool using the name of the configuration file as an argument:

     ```
     # /cdrom/hpc_3_0_ct/Product/Install_Utilities/install_gui hpc_config
     ```

   - Alternatively, you can load the configuration file after launching the tool by choosing Load from the File menu.

2. **Select the Remove task and click on the Go button.**

For help using the configuration tool, choose Help with Configuration Tool from the Help menu.

Removing the Software: Command Line

1. **Locate a configuration file or files for the cluster.**

   To remove the software from your cluster, you will need a configuration file that describes the cluster. Ideally you should use the configuration file you created when installing the software. If you cannot locate that file, you will have to create one.

   Note - You can use the CCM tools to install on up to 16 nodes at a time. For clusters with more than 16 nodes, you will have to repeat the installation process on groups of up to 16 nodes at a time until you have installed the software on the entire cluster.

   ---

2. **Place the Sun HPC ClusterTools 3.0 distribution CD-ROM in the CD-ROM drive.**

3. **Go to the directory on the CD-ROM containing the release packages.**

   This directory must be mounted with read/execute permissions (755) on all the nodes in the cluster:

```
# cd /cdrom/hpc_3_0_ct/Product/Install_Utilities/
```

4. **Run** `hpc_remove`**; use the** `--c` **option to specify the directory containing the** `hpc_config` **file.**

```
# ./hpc_remove --c /config_dir_install
```

The `--c` tag 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` tag.

Removing and Reinstalling Individual Packages

To remove a single package and install (or reinstall) another package in its place, perform the following steps:

```
#./hpc_remove --c hpc_config_file_path -d PACKAGE_NAME
#./hpc_install -c config_dir -d location_of_package/PACKAGE_NAME
```

For example:

```
# cd /cdrom/hpc_3_0_ct/Product
#./hpc_remove --c /home/hpc_admin -d SUNWhpmsc
#./hpc_install -c /home/hpc_admin -d /cdrom/hpc_2_0_sw/Product/SUNWhpmsc
```

APPENDIX **B**

Cluster Management Tools

This appendix describes a set of cluster administration tools that are installed with the Sun HPC ClusterTools 3.0 release.This toolset, called the Cluster Console Manager (CCM), 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 3.0 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 will be done as superuser. Likewise, if `crlogin` is launched from an ordinary user prompt, all remote logins will be 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 (for example, the Sun HPC 10000 has no provision for a terminal concentrator), only `ctelnet` and `crlogin` can be used, not `cconsole`.

Launching Cluster Console Tools

All CCM tools are launched using the same command-line form:

```
% tool_name clsluster_name
```

where *tool_name* is `cconsole`, `ctelnet`, or `crlogin`, and *cluster_name* is a name given to the cluster at installation time. 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 if 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.

Note that all three CCM 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 CCM is launched.

Menu Bar

The menu bar has three menus:

- Hosts
- Options
- Help

Note that in this manual, the CCM 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 B–1 describes these menu choices.

TABLE B–1 CCM 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 CCM. ON — Enables input from the Common Window text field to the node. OFF — Disables input from the CCM.
Select Hosts	Displays the Select Hosts dialog window. See “Select Hosts Dialog Box” on page 133” for details.
Exit	Quits the CCM program.

Select Hosts Dialog Box

The Select Hosts dialog enables you to add or delete nodes during the current CCM session. The scrolled text window in the Select Hosts dialog displays a list of the nodes that are currently connected to CCM.

There are three Select Hosts dialog buttons, which are described in Table B–2

TABLE B–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.

▼ To Add 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.

▼ To Add All Nodes in a Cluster

1. **Enter the *clustername* in the Hostname text field.**

2. **Select Insert.**

   CCM 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. CCM 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.

▼ To Remove 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 B–3 for a description.

TABLE B–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 B–4 for a description

TABLE B–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 CCM 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 you want 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.

CCM 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 B–5

TABLE B–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 CCM

To issue commands to multiple nodes simultaneously:

1. **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:

1. **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 CCM tool: `clusters` and `serialports`. These files are created automatically by `cluster_tool_setup`, which places them in `/etc`. (These files are not updated automatically; if you later change cluster characteristics, you must update these files manually.)

The `clusters` File

The `clusters` configuration file maps a cluster name to the list of host names 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.