Sun™ HPC Software 2.0 System Administrator's Guide

THE NETWORK IS THE COMPUTER™

Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
650 960-1300 fax 650 969-9131

Part No.: 805-1555-10
Revision A, November 1997

Contents

Preface

The *Sun HPC Software 2.0 System Administrator's Guide* describes how to configure and manage a Sun HPC System running Sun HPC 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 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 Software 2.0 System Administrator's Guide*	805-1554-10
All	*Sun HPC Software 2.0 Release Notes*	805-2191-10
SCI	*Sun HPC SCI Guide*	805-1561-10
Installation	*Sun HPC Software 2.0 Installation Guide*	805-1562-10
Sun MPI Programming	*Sun MPI 3.0 Guide*	805-1556-10
Prism	*Prism 5.0 User's Guide*	805-1552-10
Prism	*Prism 5.0 Reference Manual*	805-1553-10
Sun HPF Programming	*Sun HPF 1.0 Guide*	805-1558-10
S3L	*S3L 2.0 Guide*	805-1557-10
LSF	*LSF User's Guide* *LSF User's Quick Reference* *LSF Administrator's Guide* *LSF Administrator's Quick Reference* *LSF Programmer's Guide*	No Sun part numbers are associated with LSF documentation.

Ordering Sun Documents

SunDocs[SM] is a distribution program for Sun Microsystems technical documentation. Contact SunExpress for easy ordering and quick delivery. You can find a listing of available Sun documentation on the World Wide Web.

TABLE P-4 SunExpress Contact Information

Country	Telephone	Fax
Belgium	02-720-09-09	02-725-88-50
Canada	1-800-873-7869	1-800-944-0661
France	0800-90-61-57	0800-90-61-58

TABLE P-4 SunExpress Contact Information *(Continued)*

Germany	01-30-81-61-91	01-30-81-61-92
Holland	06-022-34-45	06-022-34-46
Japan	0120-33-9096	0120-33-9097
Luxembourg	32-2-720-09-09	32-2-725-88-50
Sweden	020-79-57-26	020-79-57-27
Switzerland	0800-55-19-26	0800-55-19-27
United Kingdom	0800-89-88-88	0800-89-88-87
United States	1-800-873-7869	1-800-944-0661
World Wide Web: `http://www.sun.com/sunexpress/`		

Sun Documentation on the Web

The `docs.sun.com` web site enables you to access Sun technical documentation on the World Wide Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com`

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email or fax your comments to us. Please include the part number of your document in the subject line of your email or fax message.

- Email: `smcc-docs@sun.com`
- Fax: SMCC Document Feedback
  1-650-786-6443

LSF Technical Support

LSF 3.0, a product of Platform Computing Corporation, is part of the Sun HPC Software 2.0 Foundation Package. As such, it is supported by Sun as part of Sun HPC Software 2.0.

Sun HPC Software includes LSF Base and LSF Batch. However, LSF JobScheduler and LSF MultiCluster are not included and, therefore, not supported by Sun.

Information Sources for PVM and PETSc

TABLE P-5 lists organizations and resources for information about the publicly available libraries PVM and PETSc. This information is subject to change.

TABLE P-5 Information Sources for PVM and PETSc

Product	Contact
PVM	Copyright holders: University of Tennessee, Oak Ridge National Laboratory, Emory University Electronic mail: `pvm@msr.epm.ornl.gov` Newsgroup: `comp.parallel.pvm` Web site: `http://www.epm.ornl.gov/pvm/pvm_home.html`
PETSc	Developed and supported by the Mathematics and Computer Science Division of the Argonne National Laboratory.

CHAPTER 1

Introduction

This chapter:

- *Provides an overview of the Sun HPC System and outlines the system administrator's responsibilities — Section 1.1*
- *Describes the Sun HPC System hardware and software — Section 1.1.1*
- *Explains how users execute programs on the system — Section 1.2*
- *Helps you prepare to configure the system — Section 1.2.3*

1.1 The Sun HPC System

Note – In Sun HPC documentation, the terms *system* and *Sun HPC System* refer to the Sun Ultra HPC Server or Servers that are running Sun HPC Software.

The term used to describe an individual server (such as an SMP) within a cluster is *node*.

A Sun HPC System can be either a Sun UltraSPARC™-based symmetric multiprocessor (SMP) or a cluster of Sun UltraSPARC SMPs on which the Sun HPC Software is loaded. Sun HPC Software includes two resource managers,

- Load Sharing Facility (LSF)
- Sun HPC run-time environment (RTE)

Both the RTE and LSF provide support for load-balancing and management of batch and interactive throughput jobs in a cluster environment. The two resource managers accomplish this support in different ways, and for different purposes. See Section 1.1.2.2, "Sun HPC Software Foundation Package."

Sun HPC Software, which is layered on top of Solaris 2.5.1 or 2.6, provides a complete set of tools for developing, debugging, and executing serial and parallel programs on a Sun HPC System.

The system administrator's role is to install and configure the system to make it available for users. System administration responsibilities include:

- Stopping and restarting the run-time environment. See Chapter 4.
- Configuring authentication. See Chapter 4.
- Configuring and managing nodes, partitions, and queues. See Chapter 5 and Chapter 6.
- Operating and maintaining parallel file systems (PFS). See Chapter 7.
- Managing accounting records so you can monitor resource usage. See Chapter 8.
- Installing and upgrading software. See the *Sun HPC Software Installation Guide* (hardcopy only).

You can perform Sun HPC-specific system administration tasks on all nodes of the Sun HPC System at once, using one of the Cluster Console Manager programs (`cconsole`, `ctelnet`, or `crlogin`) described in Appendix A.

This manual provides an overview of Sun HPC Software and explains in detail how to configure the Sun HPC run-time environment (RTE). Once you have installed the Sun HPC Software, as explained in the *Sun HPC Software Installation Guide*, you will be ready to configure and use the system.

1.1.1 Sun HPC System Hardware

1.1.1.1 Nodes

A Sun HPC System consists of up to 256 CPUs, from 1 to 16 Sun Ultra™ HPC Server nodes, each node containing 1 to 64 CPUs. Using the Scalable Coherent Interface (SCI) interconnect, Sun Ultra HPC Clusters support up to 4 nodes. Sun HPC Software will also run on clusters of up to 16 nodes connected with any non-SCI, TCP/IP-enabled interconnect. The following SMP systems may be used as Sun Ultra HPC Servers:

- Sun Ultra HPC 2
- Sun Ultra HPC 450
- Sun Ultra HPC 3000
- Sun Ultra HPC 4000
- Sun Ultra HPC 5000
- Sun Ultra HPC 6000
- Sun Ultra HPC 10000

1.1.1.2 Networks

The official cluster interconnect for Sun HPC Software 2.0 is a Sun-supported implementation of the Scalable Coherent Interface (SCI). Sun HPC Software 2.0 supports SCI cluster sizes of from two to four nodes.

SCI is the preferred interconnection for use with parallel jobs on the cluster. Any other standard interconnect that enables TCP/IP and is supported by Sun can be used on clusters of Ultra HPC Servers.

Clustered Sun Ultra HPC Servers must have at least one non-SCI network linking the nodes in the cluster. The preferred interconnect is a 100-Base-T network. This network may be installed in addition to, or in place of, the SCI network. The non-SCI network must be installed and functioning before installing and configuring either the Sun HPC Software or SCI interconnect, if present.

To make using the Sun HPC System easier and more efficient, an additional, serial network is desirable. This option allows an administrator to monitor and control the boot console of each cluster node from a single point. This serial network must be configured so that each node's serial A port is wired to a network-connected terminal concentrator.

Note – The Sun Ultra HPC-10000 uses a special-purpose console, the *system service processor* (SSP), which cannot be replaced with a serial line connection to a terminal concentrator. Thus, the only Cluster Console Manager 2.0 applications that you can use with the Sun Ultra HPC 10000 are `ctelnet` or `crlogin`. See Appendix A for more information about Cluster Console Manager 2.0.

1.1.2 Sun HPC Software 2.0

Sun HPC Software is a suite of tools that performs resource management and system administration, and that supports the development of high-performance parallel applications. The software is available in two packages: the Sun HPC Software 2.0 Foundation package and the Parallel Development Environment (PDE).

1.1.2.1 Solaris Operating Environment

Sun HPC Software uses the Solaris 2.5.1 or 2.6 operating environment, extended by the Sun HPC run-time environment for Prism, Sun MPI, Sun HPF, the parallel file system, and related products. All programs that execute under Solaris 2.5.1 or 2.6 execute under Sun HPC Software.

Note – The Sun Ultra HPC Servers require Solaris 2.5.1 or 2.6. (The Sun Ultra HPC 10000 supports Solaris 2.5.1 and later releases).

1.1.2.2 Sun HPC Software Foundation Package

The Sun HPC Software Foundation Package consists of the following components.

Sun HPC Run-Time Environment 3.0

The Sun HPC run-time environment (RTE) includes system administration tools and a system of run-time daemons.

The system administration tools allow you to configure the system to meet the specific computing needs of your site and to monitor system usage. They are described in this manual.

The RTE controls program execution. It keeps track of system resources, ensuring that programs are executed on the nodes that best match their requirements. Using the RTE is described in the *Sun HPC Software 2.0 User's Guide*.

LSF 3.0

Load Sharing Facility (LSF) is a resource-management system that provides load sharing and distributed batch queueing. You can do every batch submission, configuration, or monitoring task via a graphical user interface, complete with hypertext-based online help. Sun HPC Software 2.0 includes LSF Base and LSF Batch (LSF JobScheduler and LSF MultiCluster are not included). They are described in the *LSF User's Guide*, the *LSF System Administrator's Guide* and the *LSF Programmer's Guide*. See Section 1.2 for a comparison of LSF and the RTE.

Sun MPI 3.0

Sun MPI is Sun Microsystems' implementation of MPI, the industry-standard specification for writing message-passing programs. Message passing is a programming model that gives the programmer explicit control over interprocess communication.

Sun MPI includes the Sun HPC MPI I/O, which supports parallel I/O for message-passing programs. Sun MPI I/O supports I/O to and from both UNIX file systems and parallel file systems. For more information about Sun MPI and Sun MPI I/O, see the *Sun MPI User's Guide*.

Parallel File System (PFS) 1.0

The Sun HPC System supports parallel I/O. Two or more disk storage devices, each connected to a separate node in the Sun HPC System, operate together to provide the basis for a parallel file system. These nodes are configured in the run-time environment database as I/O servers.

Under PFS, individual files are distributed across multiple disks and I/O servers. Data are read from and written to multiple devices in parallel, promoting higher overall I/O performance. PFS is optimized for the complex data access patterns that characterize parallel scientific applications. For more information, see Section 6.10, "Parallel File Systems," and Chapter 7, "PFS Operations and Utilities."

PVM 3.3.11

PVM (Parallel Virtual Machine) is a publicly available library of routines for message-passing used widely in academic and research computing. PVM is provided at no cost to the user and is not supported by SunService or covered under any maintenance agreement.

Cluster Console Manager 2.0

The Cluster Console Manager is suite of applications (`cconsole`, `ctelnet`, and `crlogin`) supplying a graphical user interface that enables you to execute commands on multiple nodes simultaneously, thus simplifying the tasks of administering Sun HPC Clusters. At startup, the selected Cluster Console Manager application opens a window that is common to every node in the cluster. One or more terminal windows open at the same time, each of which is connected to a different node. For further information about Cluster Console Manager applications, see Appendix A.

Switch Management Agent 2.0

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

1.1.2.3 Sun HPC Parallel Development Environment

The Parallel Development Environment (PDE) is an optional package of software for use on Sun HPC Systems.

Prism 5.0

Prism is a software environment for developing and debugging programs and for visualizing data. It is described in the *Prism User's Guide* and *Prism Reference Manual*.

Sun HPF 1.0

Sun's High Performance Fortran (Sun HPF) compiler is an implementation of Subset HPF. Sun HPF produces parallel codes that will run on Sun HPC Systems. The Sun HPF compiler includes parallel I/O capabilities that allow users to take advantage of the parallel file system. Sun HPF requires Sun Performance WorkShop™ Compiler Fortran v.4.0 or v4.2. (Sun Performance WorkShop Fortran v3.0, described below, includes the v4.2 compilers.) Sun HPF is described in the *Sun HPF Guide.*

S3L 2.0

The Sun Scientific Subroutine Library (S3L) provides a set of parallel and scalable functions and tools used widely in scientific and engineering computing. S3L is described in the *S3L User's Guide.*

S3L uses the Sun Performance Library™ for nodal computation. See Section 1.1.2.4.

Note – The Sun Performance Library is made available to S3L users as part of either WorkShop Compilers Fortran v4.2 or Performance WorkShop Fortran v3.0.

PETSc 2.0.17

The Portable, Extensible Toolkit for Scientific Computation (PETSc) is an extensive package of mathematical library routines for sparse iterative linear and nonlinear solvers. PETSc is provided at no cost to the user and is not supported by SunService or covered under any maintenance agreement. Developed and supported by the Mathematics and Computer Science Division of the Argonne National Laboratory, PETSc is targeted at the resolution of partial differential equations.

1.1.2.4 Sun Performance WorkShop Fortran v3.0

Sun Performance WorkShop Fortran is a collection of compilers, debuggers, libraries, and tools for both single processor and SMP systems. Packaged with Sun Ultra HPC Servers, some Fortran components of the Sun Performance Workshop are required by Sun HPC Software.

1.2 Comparing LSF and the RTE

Sun HPC Software provides two alternative resource management systems: LSF and the RTE. These systems do not interoperate. This section describes and compares both systems.

1.2.1 Accessing the System

1.2.1.1 Load Sharing Facility (LSF)

Users can access the LSF environment via a load-sharing login session, using the `lslogin` command. Every job submission, configuration, or monitoring task can be accomplished using a graphical user interface (GUI). The LSF GUI includes hypertext-based online help. LSF also supports AFS and Kerberos authentication of users.

Note – The binary files required for installing LSF with AFS can be downloaded from the Platform Computing website: `http://www.platform.com`.

For further information on accessing LSF, see the *LSF User's Guide*, which is provided with the Sun HPC Software documentation set.

1.2.1.2 Run-time Environment

Users access the run-time environment (RTE) of their Sun HPC System directly via the `tmlogin`, or `tmtelnet` commands, as described in the *Sun HPC User's Guide*. The RTE supports Kerberos and DES (public key–based) daemon authentication, as well as user authentication. Remote access is available via the `-s` (*system name*) option to the `tmadmin`, `tmrun`, and `tmsub` commands, as described in Chapter 5, "System Administration Tools," and in the *Sun HPC User's Guide*.

Note – In Sun HPC documentation, the terms *system* and *Sun HPC System* refer to the Sun Ultra HPC Server or Servers that are running Sun HPC Software. A Sun HPC System (or system) may consist of a cluster of servers or a single server, depending on your local configuration.

The term used to describe an individual server (such as an SMP) within a cluster is *node.*

The difference between `tmlogin` and `tmtelnet` is the protocol they use. `tmlogin` behaves like `rlogin`, and `tmtelnet` behaves like `telnet`. If you use `tmlogin` or `tmtelnet`, you can also specify the partition you want to log in to. `rlogin` and `telnet` simply log you in to the specified host without the benefit of load-balancing.

To make `tmlogin` available to users outside the cluster, install `tmlogin` on systems from which users will log in to the Sun HPC System.

1.2.2 Program Execution

Both resource managers support detailed resource requirement strings. Both offer selectable submission parameters for batch processing. LSF and the RTE offer comparable data for monitoring loads, static resources, hosts, clusters, and several other categories.

1.2.2.1 Load Sharing Facility

LSF supports both interactive (`lsrun`) and batch program execution (`bsub`), as well as allowing users to run interactive jobs within the batch system.

LSF has an extensive array of batch features, including:

- Specifiable job start and end times; time windows
- Exclusive jobs
- Checkpointing and migration
- Rerunnable jobs
- Job dependency options—jobs scheduled to run conditionally, in series
- File transfers for nonuniform file systems
- Maximum and minimum numbers of processors
- Job migration on demand when threshold parameters are exceeded

LSF provides more varieties of queue support than the RTE, including:

- Processor reservation
- Tracking and managing jobs after submission to the queue
- Resource limit enforcement

- Access control
- Fairshare scheduling policy (also preemptive and exclusive policies)
- Job migration thresholds
- Pre- and post-execution commands for configuring hosts, per queue
- Requeueing when a job exits with a certain status
- Configurable job control actions (suspend, resume, or terminate)

For further information on executing programs within LSF, see the *LSF User's Guide*.

1.2.2.2 Run-Time Environment (RTE)

Users of the run-time environment (RTE) execute their programs in one of two ways: interactive or batch.

Interactive programs are initiated via the `tmrun` command, which is described in the *Sun HPC Software User's Guide*. They are executed immediately, provided the requested resources are available. If the necessary resources are not available, the user must keep trying to execute the task until they are.

Batch jobs are initiated via `tmsub`, which is also described in the *Sun HPC Software User's Guide*. They are put in a batch queue, where they await resources. Programs submitted to a batch queue are executed in order of submission. If there is high demand for a particular resource—for example, a dedicated partition or a node with special hardware, such as a disk array—the most equitable way to manage access to it is through the use of batch queues.

Options to `tmrun` and `tmsub` allow users to specify the computing resources they need for a program. For example, a user can specify a partition, specific hardware, or memory requirements.

The RTE maintains the state of the system, continually monitoring system resources, such as the loads on all nodes. When a program is executed, the RTE determines where it will run. A program is run on the least-loaded node or on nodes that meet user-specified criteria.

Once a program is started, the RTE continues to control its execution automatically. An instance of a running program is known as a *task*. In a message-passing program, the task can consist of multiple *processes*. A task can consist of up to 256 processes. These processes can run on multiple nodes or on a single node. To control a message-passing program, the nodes have to be able to communicate. Communication setup takes place via the RTE.

1.2.3 Summary

1.2.3.1 Interactive or Batch Configurations

Both LSF and the RTE allow users to run programs interactively, typically in order to debug them. LSF also lets users run interactive sessions within the more regulated environment of the LSF Batch system. However, LSF does not support the Prism development environment. If users want to debug in Prism, they must use the RTE.

Both LSF and the RTE provide the tools to maintain a production environment, one in which programs don't need attention while they run. LSF has extensive batch and queue features. Also, LSF allows you to specify non-interactive batch operation, ensuring uninterrupted batch processing. The RTE allows you to create batch-only partitions. Batch partitions allow jobs to be run only via queues.

1.2.3.2 Shared or Dedicated Configurations

Both LSF and the RTE allow users to run large serial jobs. However, if throughput of multiple serial jobs is a high priority, LSF is the preferred environment. To maximize throughput within the RTE, you can create a shared partition. In a shared partition, multiple programs can run simultaneously.

Both LSF and the RTE provide support for parallel jobs. However, if maximum performance from parallel jobs is a high priority, the RTE is the preferred environment. To ensure the fastest possible execution of parallel jobs, create a dedicated partition. In a dedicated partition, only one program has access to the nodes in the partition at a time.

LSF is not compatible with Sun HPF, Sun MPI, PFS, or Prism.

1.2.3.3 Varying Configurations

If usage patterns will vary over time—for example, if users run programs interactively during the day and submit large batch jobs at night—you can create a mix of interactive partitions and batch-only (RTE) partitions or non-interactive batch partitions (LSF), enabling only those partitions that are needed at a given time. You can automate the enabling and disabling of the partitions by creating a script that runs first thing in the morning and after the end of the workday.

If your usage matches only one of the patterns listed above, you might want to create only one partition.

Section 6.8 describes common types of RTE partitions and how to configure them.

CHAPTER 2

Getting Started

This chapter assumes that the installation process has concluded successfully and the `part_initialize` script (introduced at the end of the installation process) has been run. The discussions in this chapter describe the post-installation configuration phase—the steps needed to get your Sun HPC System ready for use. The steps covered in this chapter include

- Starting the RTE and LSF daemons
- Verifying system readiness
- Adding a parallel file system (PFS)
- Testing MPI communications

2.1 Start the RTE and LSF Daemons

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

Note that the master node is specified in the `hpc_config` file in the line `MASTER_NODE="`*hostname*`"`

1. **On the master node, start the RTE master daemons as root.**

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

2. **On all the nodes in the cluster, start the RTE nodal daemons and LSF daemons.**

   Note, if available, you may want to use one of the Cluster Console tools for this step. This would allow you to broadcast the command to all the nodes at once.

```
# /etc/init.d/rte.node start
# /etc/init.d/lsf start
```

2.2 Verify Basic Functionality

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

Note – This procedure assumes that the RTE master and node daemons and LSF daemons have been started, as described in Section 2.1.

2.2.1 Run `tminfo`

Run `tminfo -N` as a first test.

`tminfo` is a Sun HPC inquiry command that provides information about the system's configuration and resource usage. It supports various options that determine which aspect of the system will be probed: partitions, queues, nodes, or all.

For this test, specify the `-N` option to examine the state of the cluster's nodes. Note, the following procedure assumes that `/opt/SUNWhpc/bin` is in your path.

```
# tminfo -N
NAME         UP PARTITION   OS       OSREL NCPU   FMEM   FSWP LOAD1 LOAD5 LOAD15
dev-node18    y -           SunOS    5.5.1    1   7.17  74.76  0.03  0.04  0.05
dev-node19    y -           SunOS    5.5.1    1  34.70  38.09  0.06  0.02  0.02
```

FIGURE 2-1 Sample `tminfo -N` Output for a Two-Node System

Note – If `part_initialize` has been run at the conclusion of the installation process, nodes will be in partition `all`, as shown in FIGURE 2-2.

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 node daemons

2.2.2 Verifying the RTE Setup

After you have configured the RTE into the partition `all`, run `tminfo -N` again. This time, its output should show the nodes are in a partition named `all`.

```
# tminfo -N
NAME         UP PARTITION   OS       OSREL NCPU   FMEM   FSWP LOAD1 LOAD5 LOAD15
dev-node18    y all         SunOS    5.5.1    1   8.26  74.68  0.00  0.01  0.03
dev-node19    y all         SunOS    5.5.1    1  34.69  38.08  0.00  0.00  0.01
```

FIGURE 2-2 `tminfo` Output for the Sample Partition `all`

2.2.3 Run `tmrun` –`Ns`

If the RTE initialization script completed successfully, you can use `tmrun` to verify the RTE setup. Note that the RTE does not sort or rank the output of `tmrun` by default, so host name ordering may vary between runs. You should see all the host names in your cluster printed one-per-line.

```
# tmrun -Ns -np 0 hostname
dev-node9
dev-node18
```

FIGURE 2-3 `tminfo` Output for a Two-Node System

2.2.4 Verify That RTE and LSF Execute Jobs

To ensure that the RTE and LSF are ready to execute your programs, run the tests in the following examples:

```
% tmrun -np 0 uname -a
% tmsub -np 0 uname -a
% lsrun uname -a
% bsub uname -a
```

2.3 Adding a PFS and Storage Objects

This section describes two methods to accomplish the same objective, creating a parallel file system (PFS).

2.3.1 Creating a PFS Interactively

The Sun HPC parallel file system offers clustered servers a fast, high-capacity means for supporting parallel applications within the RTE. To set up a PFS, start `tmadmin` as root (for information about `tmadmin`, see Chapter 5 and Chapter 6) and take the following steps:

1. **Identify which nodes will function as PFS I/O servers for the new PFS, and update their node objects in the RTE database with two pieces of additional information.**

   The following example illustrates three nodes being configured for use as PFS I/O servers. For each, 128 buffers are allocated to I/O cache use, and the SCI interface is specified as the communication network. Their names, `io-node8, io-node9`, and so on, were given to them when they were created.

   ```
   [gauss]:: node current io-node8
   [gauss] N(io-node8):: set pfs_buffers = 128
   [gauss] N(io-node8):: set pfs_network = fa0
   [gauss] N(io-node8):: current io-node9
   [gauss] N(io-node9):: set pfs_buffers = 128
   [gauss] N(io-node9):: set pfs_network = fa0
   [gauss] N(io-node9):: current io-node10
   [gauss] N(io-node10):: set pfs_buffers = 128
   [gauss] N(io-node10):: set pfs_network = fa0
   ```

   See Section 6.6.4 for more complete details.

2. **Use the `tmadmin create` command to name a new file system and to add that name to the RTE database.**

   Use the `create` command to assign a name to a new parallel file system. The name is added to the RTE database. For example, to add the PFS `cities` to the database:

   ```
   [gauss] N(io-node10):: top
   [gauss]:: pfs
   [gauss] F():: create cities
   [gauss] F(cities)::
   ```

   See Section 6.10.4 for more complete details.

3. **Describe the new PFS's storage objects in the RTE database.**

   The following example uses the same three I/O servers defined in Step 1. Note that it begins in the context of the new PFS, `cities`.

```
[gauss] F(cities):: disk create 1
[gauss] F(cities) D(001):: set node = io-node8
[gauss] F(cities) D(001):: set device = "/dev/rdsk/c0t1d0s2"
[gauss] F(cities) D(001):: disk create 15
[gauss] F(cities) D(015):: set node = io-node9
[gauss] F(cities) D(015):: set device = "/dev/rdsk/c0t1d0s2"
[gauss] F(cities) D(015):: disk create 30
[gauss] F(cities) D(030):: set node = io-node10
[gauss] F(cities) D(030):: set device = "/dev/rdsk/c0t1d0s2"
```

   See Section 6.11.3 for more complete details.

2.3.2 Creating a PFS from a File

You can use the –`f` option to `tmadmin` to create a new PFS.

For example, if you wish to install a PFS named *cities*, you could execute the following command using a file (written using the `tmadmin dump` format) called cities.txt:

```
# tmadmin -f cities.txt
```

The contents of `cities.txt`:

```
top
node current io-node8
set pfs_buffers = 128
set pfs_network = fa0

top
node current io-node9
set pfs_buffers = 128
set pfs_network = fa0

top
node current io-node10
set pfs_buffers = 128
set pfs_network = fa0

#
# Parallel File Systems
#

top
pfs create cities
```

```
#
# PFS pfs disks
#
top
pfs current cities disk create 1
set node = io-node8
set device = /dev/rdsk/c0t1d0s2
top
pfs current cities disk create 15
set node = io-node9
set device = /dev/rdsk/c0t1d0s2
top
pfs current cities disk create 30
set node = io-node10
set device = /dev/rdsk/c0t1d0s2
```

For more information about `tmadmin`, see Chapter 5 and Chapter 6.

2.4 Running Basic Tests

2.4.1 Verify MPI Communications

The program `mpptest` is a performance test suite originally distributed with MPICH, a public domain version of MPI. Sun provides this program as an unsupported convenience utility. The MPICH `readme` file for `mpptest` is included with the `mpptest` source files.

If the tests are run as suggested, they can provide you with some level of confidence that the installed version of Sun MPI is functioning. Note that `mpptest` was intended to measure performance; therefore, no error checking is performed.

It is assumed that `cc` and `make` are in your path and running as root.

1. **Go to the directory containing the test routines.**

   # **`cd /opt/SUNWhpc/examples/mpi`**

2. **Compile the test software.**

   # **`make mpptest`**

   Note: If the Sun HPC Software was not NFS-mounted, then the executable, `mpptest`, needs to be copied to the same location on all other nodes in the system.

3. **Execute some of the following tests:**

   Point-to-point tests (between pairs of nodes):

   - To test blocking sends and receives (`MPI_Send`, `MPI_Recv`)

   # **/opt/SUNWhpc/bin/tmrun –np 2 mpptest –sync**

   - To test non-blocking receives (`MPI_Irecv`)

   # **/opt/SUNWhpc/bin/tmrun –np 2 mpptest –async**

   - To test ready send (`MPI_Rsend`, `MPI_Irecv`)

   # **/opt/SUNWhpc/bin/tmrun –np 2 mpptest –force**

   - To test persistent requests (`MPI_Send_init`, `MPI_Recv_init`, `MPI_Start`, `MPI_Startall`, `MPI_Wait`, `MPI_Waitall`)

   # **/opt/SUNWhpc/bin/tmrun –np 2 mpptest –persistent**

   Collective operations tests (one test per node):

   - To test `MPI_Allreduce` using `MPI_DOUBLE`

   # **/opt/SUNWhpc/bin/tmrun –np 0 mpptest –gop –dsum**

   - To test `MPI_Allreduce` using `MPI_INT`

   # **/opt/SUNWhpc/bin/tmrun –np 0 mpptest –gop –isum**

   - To test `MPI_Barrier`

   # **/opt/SUNWhpc/bin/tmrun –np 0 mpptest –gop –sync**

   - To test `MPI_Bcast`

   # **/opt/SUNWhpc/bin/tmrun –np 0 mpptest –gop –bcast**

CHAPTER 3

System Architecture

This chapter discusses some principles of Sun HPC System architecture, with suggestions and rules of thumb for configuring a Sun HPC System.

Note – In Sun HPC documentation, the terms system and Sun HPC System refer to the Sun Ultra HPC Server or Servers that are running Sun HPC Software. A Sun HPC System (or system) may consist of a cluster of servers or a single server, depending on your local configuration.

The term used to describe an individual server (such as an SMP) within a cluster is node

3.1 Nodes

Configuring an SMP or cluster of SMPs for Sun HPC Software use involves many of the 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 System have such a large effect on the optimal settings of these parameters, the following discussion is necessarily general in nature.

3.1.1 Number of CPUs

Since both Sun MPI and Sun HPF 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 system. This is not a necessary condition since Sun

MPI and Sun HPF 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.

3.1.2 Memory

Generally, the amount of installed memory should be proportional to the number of CPUs in the system, 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.

3.1.3 Swap Space

Because Sun HPC applications are, on average, larger than those typically run on compute servers, the swap space allocated to Sun HPC systems should be correspondingly larger. The amount of swap should be proportional to number of CPUs and to the amount of installed memory. In addition, if Sun MPI or Sun HPF jobs are to be run on the system, additional swap should be configured to act as backing store for the shared memory communication areas used by Sun HPC Software 2.0 in these situations.

Sun MPI (and, by extension, Sun HPF) tasks 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 task (with both processes running within the same SMP) requires shared memory files of approximately 35 Mbytes. A 16-process task (all processes running within the same SMP) requires shared memory files of approximately 85 Mbytes. A 256-process task (all processes running within the same SMP) requires shared memory files of approximately 210 Mbytes.

3.2 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 Sun HPC run-time environment (RTE).

In a broad sense, an 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. Here, this kind of network traffic is referred to as *administrative* traffic.

Administrative traffic has the potential to tax system resources. This can result in severe performance losses for some Sun HPC applications unless these resources are somehow protected from this traffic. Fortunately, the Sun HPC Software suite has 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.

3.2.1 Sun HPC Software Requirements

Several components of the Sun HPC Software suite generate internode communication. It is important to understand the nature of these communications in order to make informed network configuration decisions.

3.2.1.1 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 occurs as the executable is read from a (typically) NFS-mounted disk and paged into a node's memory. In contrast, when a 16- or 32-process parallel program is invoked, the NFS server can be hit quite hard by demands from multiple nodes—each pulling pages of the executable to its own memory. Such requests are likely to occur at about the same time, resulting in large amounts of network traffic—depending on the number of processes in the parallel job, the size of the executable, and so forth.

The addition of other communication from parallel applications, such as message-passing traffic from large Sun MPI programs, to the traffic load on the default network can degrade performance. In later sections of this discussion, we will introduce potential answers to such communication-related questions.

3.2.1.2 RTE Traffic

The run-time environment (RTE) uses the cluster's default network interconnect to perform communication between the daemons that perform resource management functions. The RTE makes heavy use of this network when parallel jobs are started, with the *load* being roughly proportional to the size (number of processes) of the parallel jobs. This load is in addition to the start-up load described in the previous section. The RTE will generate a similar load during job termination as accounting information is gathered and as the RTE database is updated to reflect the departed parallel task.

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

Note – Parallel programs that communicate on standard I/O channels can also add to network load, since this data must also be routed over the network and back to the user's terminal session. In most cases, this traffic is insignificant. Note especially that jobs submitted to the RTE's batch system do not generate this kind of standard I/O network traffic.

3.2.1.3 LSF Traffic

LSF uses the cluster's default network interconnect for all of its communication. LSF does not interact with either a parallel applications network or PFS network, if either is available.

3.2.1.4 Sun MPI Traffic

Parallel programs use Sun MPI (message-passing interface) 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 RTE instructs it to use, which can be set by the system administrator. In general, the RTE instructs Sun MPI to use the fastest network available so that parallel programs obtain the best possible performance. For information about the way the RTE manages multiple network interfaces, see Section 6.7.2.2.

If the cluster has only one network, then parallel application traffic will share bandwidth with administrative and RTE 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 many parallel applications do. The administrator should understand the communication requirements associated with the types of applications to be run on the Sun HPC System 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 benefit overall performance significantly.

Note – Some Sun MPI (and Sun HPF) programs also make use of the parallel I/O capabilities of Sun HPC 2.0. Parallel I/O can generate especially large amounts of network traffic due to the large files typically accessed via parallel I/O. For information about parallel I/O, see Section 3.2.1.7, “Parallel I/O Traffic.”

3.2.1.5 Sun HPF Traffic

The advice given in the Sun MPI section applies to Sun HPF (High Performance Fortran) programs as well. This is because they are capable of generating the same kinds of network traffic.

3.2.1.6 Prism Traffic

The Prism debugger is used to tune, debug and visualize Sun HPF and Sun MPI programs running within the cluster. As Prism itself is a parallel program, starting it will generate the same sort of RTE traffic that other application invocations generate.

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 HPF or 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.

3.2.1.7 Parallel I/O Traffic

Both Sun MPI and Sun HPF parallel programs can make use of the parallel I/O capabilities of Sun HPC Software, but not all such programs will do so. The administrator needs to understand how parallel applications that are run on the Sun HPC System will make use of parallel I/O, if at all, to understand the ramifications for network load.

Applications can use parallel I/O in two different ways, and the choice is made by the writer of the application. Applications that use parallel I/O to read from and write to standard UNIX file systems can generate NFS traffic on the default network, traffic on the same network being used by the Sun MPI component, or some combination of both (See Section 3.2.1.4). 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 Sun HPC Software's parallel file system (PFS) will use the network that the RTE specifies. 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 RTE so that this type of parallel I/O would be routed over the parallel application network. A Sun HPC System 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. For information about PFS network interconnects, see Section 6.6.4.

3.2.2 Network Characteristics

Bandwidth and latency are the two most important network characteristics to consider when choosing interconnects for a Sun HPC Cluster. Another important network characteristic is performance under load.

3.2.2.1 Bandwidth

Bandwidth should be matched to expected load as closely as possible. If the intended parallel applications have only modest communication requirements and no significant parallel I/O requirements, then a fast, expensive interconnect is

unnecessary. On the other hand, many parallel applications can often benefit from *large pipes* (high-bandwidth interconnects), and clusters meant to handle such applications should use interconnects with appropriate bandwidths. Significant use of parallel I/O would also increase the importance of having a high-bandwidth interconnect.

In addition to matching bandwidth to load, connecting larger nodes (nodes with larger numbers of CPUs) with higher bandwidth interconnects in order to keep communication capabilities in balance with computational capabilities is a good rule of thumb.

An example of a low-bandwidth interconnect would be 10 Mbit/s Ethernet. Examples of higher-bandwidth interconnects include SCI, ATM, and switched FastEthernet.

3.2.2.2 Latency

The *latency* of the network (the delay incurred before a message arrives at the receiving side) is another important network characteristic, but its relevance varies according to the application. Latency is most important when the messages being sent over the network are small in size and where the latency becomes a large fraction of the total time spent transmitting the entire message.

Whether small or large messages are used by parallel programs varies according to both applications and algorithms. The site administrator should understand the communications characteristics of the relevant parallel applications when making network configuration decisions.

Parallel I/O operations are not as susceptible to latency as small-message traffic because the messages transferred by parallel I/O operations tend to be large (often 32 Kbytes or larger).

3.2.2.3 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.

3.3 Storage and the Parallel File System

The performance of parallel applications can be enhanced by using a Sun HPC parallel file system (PFS). The value contributed by PFS depends on how storage and I/O are configured on your Sun HPC System.

3.3.1 PFS on SMPs and Clusters

Although a parallel file system (PFS) can be used in a Sun HPC System consisting of a single SMP, PFS provides a greater benefit to a cluster of SMPs. On a single SMP, the best I/O performance is likely to be achieved using a high-performance file system, such as VxFS.

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

3.3.2 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, individual underlying storage devices in the PFS should be storage arrays with some form of RAID support installed.

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 between multiple servers.

3.3.3 PFS and Storage Size

The amount of storage that should be configured in a parallel file system is directly related to the actual workload the cluster is expected to support.

There are two general principles to keep in mind when planning the size of a PFS:

- Files in an HPC environment are likely to be much larger than those in a typical UNIX environment.
- PFSs are typically used to store temporary data and as caches for tertiary storage.

Therefore, while the size of each file may be much larger than in a UNIX file system, those files are likely to reside in a parallel file system for a much shorter time than they would in a UNIX file system.

The amount of storage space needed will vary, according to your applications.

3.3.4 PFS and Storage Placement

Typically, administrators assign some nodes in a cluster to the role of I/O server and reserve the remainder for computational work. Often this strategy has been based on an assumption that each individual node is relatively underpowered. Given the computational power and I/O bandwidth of today's nodes, it is less likely to be beneficial to segregate I/O and computation.

Conversely, when configuring a PFS in a cluster with *N* nodes configured as I/O servers, *N*-1/*N* of the I/O will be going off-node. As a result, colocating computation with I/O server nodes is not likely to significantly increase I/O throughput.

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

- Separate compute and I/O nodes
- Mixed function nodes

3.3.4.1 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 a case, 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 (approximately) 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.

3.3.4.2 Mixed Functions

When nodes act as both compute engines and parallel I/O servers, the same considerations related to network bandwidth discussed above still apply. However, performance in this case can be increased by launching jobs so that their processes are scheduled onto the same set of nodes that host the PFS storage objects. In this case, performance may be increased because some percentage of the I/O operations performed will access local disks, thereby reducing network access to remote disks. In order to maximize efficiency in this case, applications should be examined to determine the most efficient mapping of their processes onto cluster nodes. Then, the parallel 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.

3.3.5 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 4

Running Sun HPC Software

This chapter describes

- *The run-time environment daemons — Section 4.1*
- *The procedures for stopping and restarting the run-time environment — Section 4.2*
- *The procedures for updating RTE database information — Section 4.3*
- *The procedure for redirecting error messages to a dedicated log file — Section 4.4*
- *The procedures for enabling or changing RTE methods of authentication — Section 4.5*

4.1 Run-Time Environment Daemons

The run-time environment (RTE) consists of a set of daemons, some of which run on a single node exclusively (called the *master node* since it has the master daemons: `tm.rdb`, `tm.mpmd`, `tm.watchd`), some of which run on all of the nodes (`tm.omd` and `tm.spmd`). The daemons work cooperatively to maintain the state of the system and manage program execution.

- The *resource database daemon* (`tm.rdb`) runs only on the master node and implements the resource database used by the other pieces of the run-time environment. This database represents the state of the system and the tasks running on it. To use authentication, the mode of authentication must be specified when starting the database daemon; see Section 4.2. Other pieces of the RTE add, update, and delete objects from the database and perform queries on the database to implement run-time environment policy.
- The *object-monitoring daemon* (`tm.omd`) runs on all the nodes (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 (host name, network interfaces, and so on) when it first starts up.

- The *master process-management daemon* (`tm.mpmd`) runs only on the master node and performs the following functions:
  - Acts as a server for user (client) requests via `tmrun` and `tmsub`
  - Interacts with the resource database (via calls to `tm.rdb`)
  - Coordinates nodal slave process-management daemons
- The *slave process-management daemon* (`tm.spmd`) runs on all the compute nodes of the Sun HPC System and does the following:
  - Handles spawning and termination of nodal processes (by command from `tm.mpmd`)
  - In conjunction with `tmrun`, handles multiplexing of `stdio` streams for nodal processes
  - Interacts with the resource database (via calls to `tm.rdb`)
- The *watcher daemon* (`tm.watchd`) runs only on the master node and performs the following functions:
  - Marks individual nodes as online or offline, by periodically executing remote procedure calls (RPCs) to all of the nodes.
  - Monitors and clears stale resource database (`rdb`) locks.
  - If the `–Yk` option has been enabled, aborts tasks that have processes on nodes determined to be down. This is disabled by default.

  `tm.watchd` (if the `tm.watchd` task killing option, `–Yk` is enabled) cleans up after aborted programs by killing orphaned processes when an offline node comes back online. If a node remains in the offline state indefinitely, it is the administrator's responsibility to remove the task object from the RTE database by executing `tmkill –C` (available only to the administrator, and only on the master node).

  Use `tmkill –C` with caution if the `tm.watchd –Yk` option is enabled. If an administrator removes the task object without waiting for `tm.watchd`, the system administrator becomes responsible for killing orphaned processes on nodes that have come back online. Failure to kill orphaned processes can cause resource problems.

Note – If a running `tmrun` process becomes unresponsive on a system, even where `tm.watchd –Yk` has been enabled, it may be necessary to use Ctrl-c to kill `tmrun`.

See the Sun HPC man pages for more information about each of the RTE daemons.

4.2 Stopping and Restarting the RTE

You can stop the RTE in several ways. If you want to shut down the entire system with the least risk to your file systems, use the `shutdown` command.

Stopping and/or restarting RTE, without shutting down the entire system, requires the steps listed below. By stopping the RTE daemons via the `rte.node` command (as described below), you can continue to use your nodes as individual systems without the load factor contributed by Sun HPC Software processes. If you want to exclude a node from the RTE, use `tmadmin` (See Chapter 6, "Configuring the System") to remove the node from any RTE partitions.

Note – Nodal daemons should always be brought up after the master daemons. Otherwise the nodal daemons will not be initialized properly and the run-time environment will not work. Also, nodal daemons should be shut down before the master daemons. If you shut down the master daemons first, you will be unable to use one of the Cluster Console Manager applications (`cconsole`, `ctelnet`, or `crlogin`) to issue a single command to all nodes. If you shut the master node down before shutting down the rest of the nodes, you must shut down each node individually.

▼ To Shut Down The RTE Without Shutting Down Solaris

Execute these commands from the master node.

1. **Stop the nodal daemons on the current node:**

   # **`/etc/init.d/rte.node stop`**

2. **Stop the master daemons:**

   # **`/etc/init.d/rte.master stop`**

Note – To shut down all node daemons on all nodes without repeating the `stop` command, use one of the Cluster Console Manager applications (`cconsole`, `ctelnet`, or `crlogin`). You should execute this command from the Cluster Console application's Common Window. For further information about the Cluster Console Manager, see Appendix A.

▼ To Restart the RTE Without Rebooting Solaris

Execute these commands from the master node.

1. **Start the master daemons:**

   # **/etc/init.d/rte.master start**

2. **Start the nodal daemons:**

   # **/etc/init.d/rte.node start**

Note – When the `rte.master` and `rte.node` programs are executed with `start` commands, they both initiate a `stop` command on all currently running daemons before restarting RTE daemons. `rte.node` and `rte.master` behave differently when executing a `stop` command: `rte.master` stops all processes that have been spawned by the RTE before killing the daemons; whereas `rte.node` does not kill user processes. Consequently, client daemons can be restarted without affecting running tasks.

4.3 Updating the RTE Database

4.3.1 Preserving RTE Database Information

You can safeguard the current information in your RTE database using the `tmadmin dump` command. By redirecting the output of the `dump` command to a file, you can use the contents of that file at a later time to reconstruct your database using the `tmadmin –f` option. For information about using the `tmadmin` command to store and recreate RTE database information, see Section 6.4.4.

4.3.2 Rereading the Distribution of CPUs

You can reconfigure the composition of groups of system boards (domains) on the Sun Ultra HPC 10000. This feature is called *dynamic reconfiguration*. If you reconfigure the domains of a Sun Ultra HPC 10000, you should update the information in the RTE database by executing

/etc/init.d/rte.node reread

on each domain that you have reconfigured.

Note – To prevent potential race conditions for new jobs, you should unset the enabled attribute on every partition in the affected domains before reconfiguring them and issuing the `rte.node reread` command.

4.4 Redirecting Error Messages to a Log File

You can configure the RTE to direct the error messages generated by the RTE daemons to a dedicated log file. For information about using the `tmadmin` command to set the `logfile` attribute, see Section 6.5.2.5.

4.5 To Enable or Disable Authentication

Authentication software ensures increased levels of security, guarding against access by unauthorized users or programs.

The RTE supports two forms of authentication: Data Encryption Standard (DES), and Kerberos version 4. Select the appropriate authentication method used by the `rte.master` script described in Section 4.2, “Stopping and Restarting the RTE.” You must edit this script (or a renamed copy of it), changing the current authentication method (stored in the script at installation time) to match your choice of methods. Then run the script as described. The `tm.rdb` daemon starts up with the argument `tm.rdb –a` *authentication_method.*

where the authentication method is one of:

- `none` – No authentication software used (the default)
- `des` – DES-based authentication
- `kerb4` – Kerberos version 4 authentication

When authentication option `none` is in use, any `tmadmin`, `tmrun`, or `tmsub` operation attempted by root will be allowed only if:

- The requesting host
- The master RTE host
- The hosts on which any tmrun or tmsub operation will run

either appear in the /etc/sunhpc_rhosts file, if it has been installed (the default), or in the default .rhosts file (if the sunhpc_rhosts file is not created at installation, or if it has been deleted). The sunhpc_rhosts file has read and write permissions set only for root. No other permissions are set. Therefore, the file is visible only to root.

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

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

If you change authentication methods (by stopping tm.rdb, and then restarting it with a different authentication method), you must restart all the other RTE daemons, too.

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

4.5.0.1 Using DES Authentication

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

To start RTE daemons while using DES authentication, you must issue the keylogin command on each node in the Sun HPC System. Use one of the Cluster Console Manager applications (cconsole, ctelnet, or crlogin) to issue identical commands on multiple nodes at the same time. For information about the Cluster Console Manager, see Appendix A.

Note – While DES authentication is in use, users must issue the keylogin command before issuing any commands beginning with tm, such as tmrun or tmps.

4.5.0.2 Using Kerberos Version 4 Authentication

To set up Kerberos version 4 authentication, you must ensure that all users who will issue commands beginning with `tm`, such as `tmrun`, or `tmps`, have entries in the Kerberos database.

System administrators must register a principal name of the Sun HPC Software (`sunhpc`) with an instance for each host of the Sun HPC System in the Kerberos database. Appropriate `srvtab` files must be stored on each host.

For example: For a system consisting of two nodes, `node0` and `node1`, in realm *example.com*, the database will have two principals:

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

`node0`'s `srvtab` file must have an entry for `sunhpc.node0@example.com`, and `node1`'s `srvtab` file must have an entry for `sunhpc.node1@example.com`.

Before starting the RTE daemons, you must issue the `kinit` command as superuser in order to obtain a ticket granting ticket (`tgt`). For further information, see the Solaris man page for `kinit`(1).

If you don't want `root` to be a principal in the Kerberos database, you can use either of two methods:

- Issue the `kinit` command as an ordinary user, then set the Kerberos environment variable: `KRBTKFILE` to point to the appropriate ticket file (usually `/tmp/tkt_`*uid*, where *uid* is the user's ID).
- Obtain a `tgt` for the `sunhpc` principal using the Kerberos `ksrvtgt` command.

Note – You must obtain `tgt`'s using any one of these methods on every node in your Sun HPC System, before starting the RTE daemons.

Since tgt's for the RTE daemons may expire, they have to be refreshed periodically. An example program that performs this task and a `README` file are provided in `/opt/SUNWhpc/examples/rte`.

For further information on Kerberos version 4, see your Kerberos documentation.

Note – While Kerberos version 4 authentication is in use, users must issue a `kinit` command before running any command beginning with `tm`, such as `tmrun` or `tmps`.

CHAPTER 5

System Administration Tools

If you are using the RTE, most of the administration of the Sun HPC System is performed via the `tmadmin` command. The primary exceptions are the tasks of adding or removing nodes. To perform any system administration tasks, you must be logged in to the master node or one of its client nodes as superuser.

This chapter provides an overview of `tmadmin`. In particular,

- *The syntax of the* `tmadmin` *command — Section 5.1*
- *The command-line and nested interactive interfaces — Section 5.2*
- *The commands you can use within* `tmadmin` *— Section 5.3*
- *Other useful features of* `tmadmin` *— Section 5.4*

For information about using `tmadmin` to configure your Sun HPC System, see Chapter 6. For information about configuring your Sun HPC System using LSF, see the *LSF Administrator's Guide*, and *LSF Administrator's Quick Reference*.

5.1 Invoking `tmadmin`

You can invoke `tmadmin` from a shell command line or from a file. Its syntax is:

```
% tmadmin [-c command] [-f filename] [-h] [-q] [-s systemname] [-V]
```

TABLE 5-1 lists options to the `tmadmin` command:

TABLE 5-1 `tmadmin` Options

Option	Description
`-c` *command*	Execute single specified command.
`-f` *filename*	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` *systemname*	Connect to the specified Sun HPC System.
`-v`	Display `tmadmin` version information.

5.2 `tmadmin` Interfaces

The `tmadmin` command has two interfaces:

- *A simple command-line format* – This interface is most useful for simple, single-step system administration tasks.

  For example, type the following command on one line, using the `-c` option. This example disables the queue `Big_Jobs` on the partition `Dedicated`:

  ```
  # tmadmin -c "partition Dedicated \
                queue Big_Jobs \
                unset enabled"
  ```

- *A nested interactive interface (NII)* – The NII is the more general method of using `tmadmin`. It places you in an interactive session, from which you can issue as many commands as you want.

  If you intend to issue more than one command, it is more convenient to use the NII. Simply issue the command `tmadmin`, and you're placed in an interactive session.

  ```
  # tmadmin
  [gauss]::
  ```

With the exception of commands that enable navigation within the hierarchy of nested levels in the NII, most of the tmadmin commands are available at both the command-line (following the `-c` option) and the NII.

5.3 tmadmin Commands

This section describes the tmadmin commands by reference to the NII. You can issue the same commands on the tmadmin command line using the –c option, with the exception of the commands that control context within the tmadmin command hierarchy.

When you invoke tmadmin using the NII, you are placed in an interactive session with a prompt for the next command.

```
# tmadmin
[gauss]N(node2) I(sc0)::
```

The line ending with two colons (::) is a prompt for a command. The prompt indicates the current *context* by incorporating the current Sun HPC System's name, the current NII level, and the current object created at that level, within the tmadmin prompt. In this case you are administering the Sun HPC System named gauss, at the Network Interface level (interface sc0), below the Node level (on node object node2). In this example, no system name has been specified on the command line. In this case, tmadmin either connects to the system specified by the environment variable SUNHPC_SYSTEM or defaults to the system you are logged in to.

Note – In Sun HPC documentation, the terms *system* and *Sun HPC System* refer to the Sun Ultra HPC Server or Servers that are running Sun HPC Software. A Sun HPC System (or system) may consist of a cluster of servers or a single server, depending on your local configuration.

The term used to describe an individual server (such as an SMP) within a cluster is *node*.

At any time, you can issue the command help to get a list of available commands.

5.3.1 Top-Level Commands

TABLE 5-2 lists the commands that are effective at the top level of the tmadmin NII hierarchy. These commands must be either passed by means of the –c argument of tmadmin on the command line or entered within the context of the NII.

TABLE 5-2 Top-Level `tmadmin` Commands.

Command	Synopsis
`connect` *system–name*	Connect to a Sun HPC System named *systemname*.
`set` *attribute*[=*value*]	Set a system attribute.
`unset` *attribute*	Delete a system attribute.
`show`	Show system attributes.
`dump`	Show all objects in the system.
`node`	Enter the `node` command mode.
`partition`	Enter the `partition` command mode.
`pfs`	Enter the `pfs` command mode.
`echo` ...	Print the rest of the line on the standard output.
`quit`	Quit `tmadmin`.
`exit`	Quit `tmadmin`.
`help` [*command*]	Show information about commands.
`?` [*command*]	Show information about commands.

At the top level, the `connect`, `set`, `unset`, and `show` commands act within the context of the entire Sun HPC System. Chapter 6 describes how to use these commands to configure the system. This chapter is concerned with the commands that control the NII. The levels of the NII hierarchy are depicted in FIGURE 5-1.

5.3.2 Subcommands

There are seven contexts within the `tmadmin` NII hierarchy, as illustrated in FIGURE 5-1.

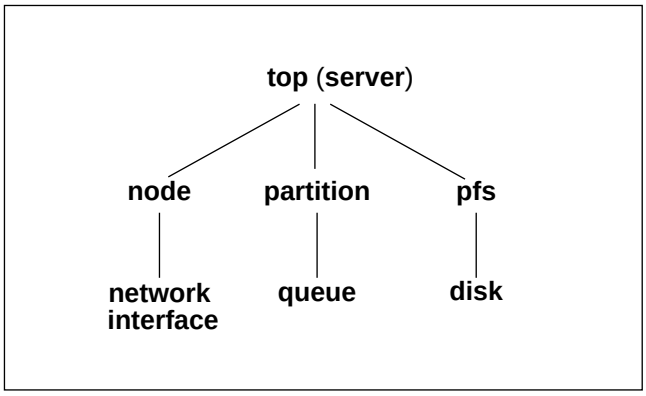

FIGURE 5-1 The `tmadmin` NII Hierarchy

From the top level, you can move to the `node`, `partition`, and `pfs` contexts by issuing the command `node`, `partition`, or `pfs`, respectively. You can move to the queue context by issuing the `queue` command from within the partition context, you

can move to the network context by issuing the `network` command from within the node context, and you can move to the disk context by issuing the `disk` command within the pfs context.

The commands at each level differ slightly. The `help` command is context-sensitive; that is, it lists only the commands specific to the current context level.

The `node`, `partition`, and `pfs` commands each drop you down a level within the command hierarchy. The new prompt then indicates the current level. For instance, if you enter the `node` command at the top level on the Sun HPC System `gauss`, the NII looks like this:

```
[gauss]:: node
[gauss] Node::
```

At this level, commands you enter are interpreted in the node context. The `partition` command works similarly.

There are three other contexts, network, queue, and disk, which are subordinate to node, partition, and pfs, respectively.

To move back up a level in the hierarchy, simply type `up`.

```
[gauss]:: node
[gauss] Node:: network
[gauss] Network:: up
[gauss] Node:: up
[gauss]::
```

Each context has its own set of commands. If you issue a command that is unavailable in the current context, `tmadmin` looks for that command at the next level up and changes context to execute it. This allows you to move from one level to another, for instance from partition to node, without first explicitly returning to the top level.

Two commands, `quit` and `help`, have synonyms. Typing `exit` is the same as typing `quit`, and `?` is equivalent to `help`.

If you type `help` alone, `tmadmin` lists the commands available at the current level. If you specify a context command (`node`, `partition`, `pfs`, `network`, `queue`, or `disk`), `tmadmin` lists the commands for that context level. Specifying another command gives you help for that particular command.

Type `quit` to end the `tmadmin` session.

TABLE 5-3 lists the commands available at the node, network, partition, queue, pfs, and disk levels of the `tmadmin` NII hierarchy. These commands are described in Chapter 6.

TABLE 5-3 Commands at the Subordinate Levels of `tmadmin`.

	Node	Network Interface	Partition	Queue	PFS	Disk
`create`	*	*	*	*	*	*
`current`	*	*	*	*	*	*
`delete`	*	*	*	*	*	*
`dump`	*		*		*	
`echo`	*	*	*	*	*	*
`help / ?`	*	*	*	*	*	*
`list`	*	*	*	*	*	*
`network`	*		*		*	
`set`	*	*	*	*	*	*
`show`	*	*	*	*	*	*
`top`	*	*	*	*	*	*
`unset`	*	*	*	*	*	*
`up`	*	*	*	*	*	*

If you set the current context to a particular object (that is, node, partition, queue, or network interface), that object's name is reflected in the prompt.

```
# tmadmin
[gauss]:: partition
[gauss] Partition:: current Big
[gauss] P(Big)::
```

At this point, any command you enter will apply to the partition `Big`.

If the current context is that of a queue or a network interface, the complete context includes that of the partition or node above it. For example:

```
# tmadmin
[gauss]:: partition
[gauss] Partition:: current Big
[gauss] P(Big):: queue
[gauss] P(Big) Queue:: current Pipeline
[gauss] P(Big) Q(Pipeline)::
```

The *network interface* context is abbreviated to `I` to avoid confusion with the `N` for node.

```
[gauss]:: node
[gauss] Node:: current Node5
[gauss] N(Node5):: network
[gauss] N(Node5) Network:: current lo0
[gauss] N(Node5) I(lo0)::
```

5.4 Additional Functionality

The `tmadmin` NII command hierarchy has the following useful features:

- Typing the name of an object automatically changes the context.

```
[gauss] Node:: Node3
[gauss] N(Node3)::
```

This works even if the object is not of the current context's type.

```
[gauss] Partition:: Node3
[gauss] N(Node3)::
```

- You can issue more than one command on a line; `tmadmin` executes them sequentially.

```
[gauss]:: node list
        Node1
        Node2
        Node3
        Node4
[gauss] Node::
```

This example sets the attribute `max_total_procs` on the partition `Dedicated`:

```
[gauss] Partition:: Dedicated set max_total_proc=1
[gauss] P(Dedicated)::
```

- You can abbreviate commands (but not the names of objects) to the shortest string of at least two letters that is still unique within the current context.

```
[gauss] Node:: pa
[gauss] Partition:: li
      Small
      Big
      Dedicated
[gauss] Partition:: Small
[gauss] P(Small):: sh
      set enabled
      set max_batch_procs = 10
      set name = Small
```

```
      set nodes = Node1 Node2
      set queues = Little_Jobs
[gauss] P(Small)::
```

is equivalent to

```
[gauss] Node:: partition
[gauss] Partition:: list
      Small
      Big
      Dedicated
[gauss] Partition:: Small
[gauss] P(Small):: show
      set enabled
      set max_batch_procs = 10
      set name = Small
      set nodes = Node1 Node2
      set queues = Little_Jobs
[gauss] P(Small)::
```

You must have superuser privileges in order to use `tmadmin` to configure a Sun HPC System. Ordinary users can use the command as an alternative to `tminfo` to display information but cannot change the configuration of the system. An ordinary user's `tmadmin` prompt ends in one colon, not two, and `help` displays only those commands that are available.

CHAPTER 6

Configuring the System

The Sun HPC System can be thought of as a collection of *objects*: nodes, partitions, parallel file systems, queues, network interfaces, disks, and the system (the top level of the NII, or *server* level) itself. The system administrator uses the `tmadmin` command to perform tasks like creating, deleting, starting, stopping, and configuring objects. Any changes to the system are automatically reflected in the resource database, described in Section 1.2.

This chapter

- *Explains about attribute types and how to obtain information about attributes — Section 6.1*
- *Defines rules for naming objects — Section 6.2*
- *Lists commands you can use to get information about the Sun HPC System — Section 6.3*
- *Describes common* `tmadmin` *commands — Section 6.4*
- *Describes how to configure the top level of the Sun HPC System — Section 6.5*
- *Describes how to configure the nodes — Section 6.6*
- *Describes how to configure network interfaces — Section 6.7*
- *Describes how to configure partitions — Section 6.8*
- *Describes how to configure queues — Section 6.9*
- *Describes how to configure parallel file systems — Section 6.10*
- *Describes how to configure PFS storage objects — Section 6.11*

6.1 Attributes

There are two kinds of attributes:

- *Boolean* – Boolean attributes can be *set* or *unset*. For instance, to enable a partition, you can set the attribute `enabled`. To disable the partition, simply unset the `enabled` attribute.
- *Value*
  - *Numeric value*. – Some attributes have numeric values. For example, you can set a limit on the number of programs that can run at once on a partition.
  - *Literal value* – The value of a literal attribute is either the name of something or a list of names. For instance, a partition's literal attributes include its own name and a list of the queues that submit jobs to it.

To get information about the current configuration of your Sun HPC System, use either the `tmadmin show` command, described in Section 6.4.6, or the `tminfo` command, described in the *Sun HPC Software User's Guide*. In general, the `tmadmin show` command displays only attributes that can be set by the system administrator. By default, `tminfo` shows a subset of attributes, but users can display the setting of any attribute by specifying it via the `–A` option to `tminfo`.

Note – To set or unset an attribute, use the `tmadmin set` or `tmadmin unset` commands, described in Sections 6.4.7 and 6.4.8.

6.2 Naming

You can assign names to nodes, partitions, parallel file systems, network interfaces, queues, and attributes. Names must start with a letter, and are case sensitive. The only limit to name length is the limit imposed by Solaris on host names, typically set at 256 characters.

The following characters are allowed:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789–_.

You cannot define an attribute starting with the characters `tm_`; these are reserved by the system.

Note – Parallel file system (PFS) storage objects, *disks*, can have only integers for names.

6.2.1 Separate Name Spaces

Nodes, partitions, PFSs, queues, and disks have separate name spaces. Thus, you can have a queue named Parallel defined on a partition named Parallel that contains a node named Parallel.

6.3 Getting Information

There are several commands you can use to get information about the Sun HPC System and processes running on it. They are:

- `tminfo` – provides information about the Sun HPC System's configuration and resource usage
- `tmps` – comparable to the `ps` command in Solaris, `tmps` tells you about currently running tasks and, optionally, processes or queued jobs

These commands and their options are described in detail in the *Sun HPC Software User's Guide*.

6.4 Common Commands

Although the assortment of `tmadmin` commands available at each level of the NII context hierarchy varies according to the objects accessed in each context, all objects have many commands in common. This section describes these commands. Most of the commands are discussed further, in the context of the various objects, in the sections that follow. See Section 5.3.1, "Top-Level Commands" and Section 5.3.2, "Subcommands."

6.4.1 current

Usage:

`:: current` *objectname*

The `current` command sets the `tmadmin` context (reflected in the `tmadmin` prompt) to the object for future commands. `tmadmin` contexts are described in Section 5.3. The `object` must be an instance of an object.

Example:

```
[gauss] Node:: current node2
[gauss] N(node2)::
```

You can also omit the `current` command. If you enter just a valid unique object name, `tmadmin` will change the context to that object.

```
[gauss] Node:: node2
[gauss] N(node2)::
```

If you omit the current command, this works even if the object is in a different context.

```
[gauss] Partition:: node2
[gauss] N(node2)::
```

Note – The previous case works only when you omit the `current` command.

You must not omit the word `current` in cases of ambiguity between a `tmadmin` command and an object name, for instance, if you have a queue named `queue`.

6.4.2 create

Usage:

`:: create` *objectname*

The `create` command creates a new object with the name *objectname*. To create an object, you must be at the context level of the type of object you want to create; that is, start at the partition level to create a new partition. The context of `tmadmin` is automatically changed to that of the new instance of the object.

Example:

```
[gauss] Partition:: create Dedicated
[gauss] P(Dedicated)::
```

6.4.3 delete

Usage:

`:: delete` [*objectname*]

The `delete` command deletes the specified object.

Example:

```
[gauss] Partition:: delete Dedicated
[gauss] Partition::
```

If you are within the context of the object you want to delete, you don't have to specify its name.

```
[gauss] P(Dedicated):: delete
[gauss] Partition::
```

The context reverts to the next higher level.

6.4.4 dump

Usage:

`:: dump` [*objectname*]

The `dump` command displays the current state of the attributes of the specified object. This object can be a node, a partition, a parallel file system, or the configuration of the entire system.

The output of a dump can be read as input by `tmadmin`. For example, you could dump the entire configuration of your system with

```
# tmadmin –c dump > sunhpc.configuration
```

Later, you could restore the state of your configuration (or update it, by editing the `sunhpc.configuration` file) with

```
# tmadmin –f sunhpc.configuration
```

When executed at the top level of the `tmadmin` hierarchy, a `dump` displays objects in a specific order, corresponding to the logical order of assignment to a configuration (essential for a restore or update operation).

For example, nodes are dumped before partitions because, when updating a system, nodes must exist before they can be assigned to a partition. For the same reason, when `dump` is executed within a node's context, the partition attribute is not included in the dump. During a restore, the partition attribute is set when the node attribute (within the partition context) is set.

Example:

```
[gauss] Partition:: dump Big
        set nodes = node1 node2 node3 node4 node5 node6
        set max_total_procs = 8
        set default_max_batch_procs = 1
        set name = Big
        set enabled
        unset default_queue
        unset no_login
[gauss] Partition::
```

(No value is listed for the attribute `enabled` because it is boolean.)

If you are within the context of the object whose attributes you want to see, you don't have to specify its name.

```
[gauss] P(Big):: dump
        set nodes = node1 node2 node3 node4 node5 node6
        set max_total_procs = 8
        set default_max_batch_procs = 1
        set enabled
        set name = Big
[gauss] P(Big)::
```

6.4.5 list

Usage:

`:: list`

The `list` command lists all of the defined objects in the current context.

Example:

```
[gauss] Partition:: list
        Dedicated
        Big
        Little
[gauss] Partition::
```

6.4.6 show

Usage:

`:: show` [*objectname*]

The `show` command displays the current state of the attributes of the specified object *objectname*.

Example:

```
[gauss] Partition:: show Big
        set nodes = node1 node2 node3 node4 node5 node6
        set max_total_procs = 8
        set max_batch_procs = 1
        set name = Big
        set enabled
        unset default_queue
        unset no_login
[gauss] Partition::
```

(No value is listed for the attribute `enabled` because it is boolean.)

If you are within the context of the object whose attributes you want to see, you don't have to specify its name.

```
[gauss] P(Big):: show
        set nodes = node1 node2 node3 node4 node5 node6
        set max_total_procs = 8
        set max_batch_procs = 1
        set enabled
        set name = Big
[gauss] P(Big)::
```

6.4.7 set

Usage:

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

The `set` command sets the specified attribute of the current object. You must be within the context of a particular object to set its attributes.

To set a literal or numeric attribute, specify its value.

```
[gauss] P(Big):: set max_batch_procs=1
[gauss] P(Big):: set default_queue=Night
[gauss] P(Big)::
```

To set a boolean attribute, specify the name of the boolean attribute to be activated.

```
[gauss] P(Big):: set enabled
[gauss] P(Big)::
```

To change the value of an attribute that has already been set, simply set it again.

```
[gauss] P(Big):: set max_batch_procs=2
[gauss] P(Big)::
```

Note – The `tmadmin` command looks only at the presence of a system-defined boolean attribute, not at any value you might erroneously set for it. (This does not apply to system administrator–defined boolean attributes. If you accidentally assign a value to an attribute you define, `tmadmin` cannot recognize it as boolean. For example, if you accidentally type `set enabled=true`, the `enabled` attribute is set. If you later type `set enabled=false`, the attribute is still set; `tmadmin` doesn't recognize the value because `enabled` isn't supposed to have one.)

6.4.8 unset

Usage:

`:: unset` *attribute*

The `unset` command deletes the specified attribute of the current object. You must be within the context of a particular object to unset one of its attributes.

Example:

```
[gauss] P(Big):: unset enabled
[gauss] P(Big):: unset max_batch_procs
[gauss] P(Big)::
```

Note – Remember that with boolean attributes, you must use `unset` to delete, not `set` the attribute to false. Using `unset` with a value attribute is equivalent to setting the attribute to its default value.

6.4.9 up

Usage:

`:: up`

The `up` command, described in Chapter 5, "System Administration Tools," moves you up one level from the current level in the `tmadmin` NII command hierarchy.

Example:

```
[gauss] Partition:: up
[gauss]::
```

6.4.10 top

Usage:

```
:: top
```

The top command, described in Chapter 5, “System Administration Tools,” moves you to the top level in the tmadmin NII command hierarchy.

Example:

```
[gauss] Partition:: top
[gauss]::
```

6.4.11 echo

Usage:

```
:: echo text–message
```

The echo command prints the specified text on the standard output. If you write a script to be run with tmadmin –f, you can use the echo command to make tmadmin print information about what the script is doing.

Example:

```
[gauss]:: echo Swapping day and night queues
Swapping day and night queues
[gauss]::
```

6.4.12 help

Usage:

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

The help command, given alone, lists the tmadmin commands that are available within the current context.

Example:

```
[gauss]:: help
connect <System–name>    connect to a Sun HPC System
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 [node]              show all objects on the System
node                     enter the node command mode
partition                enter the partition command mode
```

```
network                 enter the network interface command mode
echo ...                print the rest of the line on standard output
quit                    quit tmadmin
help [command]          show information about command command
? [command]             show information about command command
[gauss]::
```

You can specify a command to get help for that command. If you specify a context command (`node`, `partition`, `queue`, or `network`), `tmadmin` lists the commands for that context level. Note that you can specify `network` only at the node level, `disk` only at the pfs level, and `queue` only at the partition level.

```
[gauss]:: 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 top level command prompt
top                     go up to the top 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
[gauss]::
```

As a shortcut, you can substitute `?` for `help`.

6.5 The Top-Level of the NII Hierarchy

You can configure a Sun HPC System in two ways: by using the `tmadmin` command or by setting environment variables. Sections 6.5.1 and 6.5.2 describe how to configure the System using `tmadmin`. Section 6.5.3 explains the environment variables.

6.5.1 Top-Level (Server) Commands

TABLE 6-1 lists the `tmadmin` commands which apply to the Sun HPC System at the top-level (also called *server level*) of the NII hierarchy.

TABLE 6-1 Top-Level `tmadmin` Commands

Command	Synopsis
`connect` *systemname*	Connect to a Sun HPC System named *systemname.*
`show`	Show system attributes.
`dump`	Show all objects in the Sun HPC System.
`set` *attribute[=value]*	Set a server-level attribute.
`unset` *attribute*	Delete a server-level attribute.
`node`	Enter the node command mode.
`partition`	Enter the partition command mode.
`pfs`	Enter the pfs command mode.
`echo` ...	Print the rest of the line on the standard output.
`quit / exit`	Quit `tmadmin`.
`help` [*command*] `/ ?`	Show information about commands.

The `node`, `partition`, `pfs`, `echo`, `quit`/`exit`, and `help`/`?` commands are described in Chapter 5, "System Administration Tools." The `show`, `set`, and `unset` commands are described in Section 6.4. The `connect` command is described below.

6.5.1.1 `connect`

In order to configure a Sun HPC System, you must be connected to it. The name of a system is the same as the host name of its master node. There are three ways to connect to a system:

- If you are logged in to a system when you issue the `tmadmin` command, you are automatically connected to that system.
- Specify the system name after the `tmadmin –s` option. See Section 5.2, "tmadmin Interfaces."
- Use the `tmadmin connect` command.

Usage:

`:: connect` *systemname*

The `connect` command connects you to the specified Sun HPC System. An explicit connection overrides the value of the environment variable `SUNHPC_SYSTEM` described in Section 6.5.3.

6.5.2 Configuring a Sun HPC System at the Server Level

Once you are connected to a Sun HPC System at the server level, you can configure it by setting values for its attributes.

6.5.2.1 Server Level Attributes

TABLE 6-2 lists the system-defined server-level attributes. To see their current values, use either the `tmadmin show` command or the `tminfo` command. (See Section 6.1.)

TABLE 6-2 Server Level Attributes

Attribute	Kind	Description
`default_interactive_partition`	Value	The default interactive partition
`default_batch_partition`	Value	The default batch partition
`default_pfs`	Value	The default parallel file system
`logfile`	Value	An optional destination for daemon error messages
`administrator`	Value	An email address for the system administrator(s)
`lock_max_age`	Value	The maximum amount of time a lock can remain set (in seconds)

These six attributes are described in the following sections..

6.5.2.2 `default_interactive_partition`

This attribute specifies the default partition for running interactive jobs. Its value is used by the commands `tmrun`, `tmlogin`, and `tmtelnet`, which are described in the *Sun HPC Software User's Guide.*

When a user executes a program via `tmrun`, the run-time environment (RTE) decides where to run the program. To choose a partition, the RTE follows this set of steps until it finds a partition:

1. Check for a command-line `-p` flag. If a partition is specified, execute the program in that partition. If the specified partition is invalid, the command will fail.
2. Check for a setting for the environment variable `TMRUN_FLAGS`. If a partition is specified, execute the program in that partition. If the specified partition is invalid, the command will fail.
3. Check for a setting for the environment variable `SUNHPC_PART`. If a partition is specified, execute the program in that partition. If the specified partition is invalid, then check whether the user is logged into a partition, and then execute the program in that partition.
4. Check whether the user is logged into a partition. Execute the program in that partition.
5. If none of the above yield a partition name, check for the existence of the `default_interactive_partition` attribute. If a partition is specified, execute the program in that partition.

The `SUNHPC_PART` environment variable is described in Section 6.5.3. The `TMRUN_FLAGS` environment variable is described in the *Sun HPC Software User's Guide*.

6.5.2.3 `default_batch_partition`

This attribute specifies the default partition for running batch jobs. Its value is used by the `tmsub` command when no partition has been specified, which is described in the *Sun HPC Software User's Guide*.

When a user executes a program via `tmsub`, the run-time environment (RTE) chooses the queue to which the program is submitted, using these criteria:

- If the `-p` flag is used (either on the command line or in the environment variable `TMRUN_FLAGS`), only queues in the given partition are considered.
- If the `-q` flag is used, only queues with the given name are considered.
- If only one queue is eligible after applying one or both of the two preceding rules, that queue is used.
- If there is only one queue defined over all the enabled partitions in the system, it is used even if neither `-p` nor `-q` is specified.
- If no partition is specified using the `-p` flag and more than one partition has an eligible queue, the RTE takes into consideration, in order of precedence:
  - The server's `default_batch_partition` attribute
  - The environment variable `SUNHPC_PART`
  - The partition of the node on which `tmsub` was invoked

- Only partitions with queues (whose names match the `-q` argument if `-q` was used) are considered. For example, if the command was `tmsub -q Blue`, and the System has `default_batch_partition=Sapphire`, but the Sapphire partition has no queue named Blue, `tmsub` does not use the Sapphire partition.
- If a partition has been chosen, but `-q` was not used and the partition has multiple queues, look for the partition's default queue as specified by its `default_queue` attribute.

The `SUNHPC_PART` environment variable is described in Section 6.5.3. The `TMRUN_FLAGS` environment variable is described in the *Sun HPC Software User's Guide*.

6.5.2.4 `default_pfs`

This attribute specifies the default parallel file system (PFS). By specifying a parallel file system as the default, you shorten the command needed to access the default parallel file system.

For example, assume you have two parallel file systems defined: `kant` and `hegel`. You can define `hegel` to be the default filesystem using the `default_pfs` attribute. Users can access files on `kant` by specifying:

`pfs:kant:`*/full_pathname*

They can access files on `hegel` by specifying:

`pfs:hegel:`*/full_pathname*

However, since the `default_pfs` has been set to `hegel`, users can also specify:

`pfs:`*/full_pathname*

In other words, they can just drop the file system name.

6.5.2.5 `logfile`

By setting the `logfile` attribute to a specified file (using an absolute path name), you can separate RTE daemons' error messages from all other system error messages. Otherwise, by default, the RTE directs all error messages to the `syslog` facility, typically writing error messages to the file `/var/adm/messages`).

Note – The error log file is a great source of information when troubleshooting program failures, such as daemon failure.

If an RTE daemon fails to open the specified log file, all errors will be written to the `syslog` facility.

After changing the log file attribute, you must restart the `tm.spmd` and `tm.omd`, daemons. To restart `tm.spmd` and `tm.omd`, issue the `rte.client start` command on all nodes in the Sun HPC System. The `rte.client` program is described in Section 4.2.

6.5.2.6 `administrator`

Set the `administrator` attribute to specify the email address of the system administrator. (Note the use of double quotes.)

For example:

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

6.5.2.7 `lock_max_age`

The RTE uses locks for internal purposes. You can raise or lower the interval during which locks remain in force. However, changing the default value can cause system unresponsiveness if the interval is too long. If you shorten the interval from the default, you risk unreliable behavior from the system.

This attribute specifies the length of time that the run-time system waits before removing a lock. (Note the use of double quotes.) For example,

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

6.5.3 Environment Variables

The following environment variables can be used to create default settings:

TABLE 6-3 Environment Variables

Variable	Definition
`SUNHPC_CONFIG_DIR`	Directory in which `tm.rdb` keeps resource database files. The default is `/var/hpc`.
`SUNHPC_SYSTEM`	Name of the default Sun HPC System.
`SUNHPC_PART`	Name of the default partition.

6.6 Nodes

The following rules govern nodes in a Sun HPC System:

- A node must be added to the Sun HPC System before it can be configured into a partition or enabled for use. For information about creating nodes, see Section 6.6.2.
- A node can be a member of no more than one enabled partition.
- A node that is not a member of any partition can be used by any task, provided the node is enabled and that it is specifically requested, either directly (by name) or indirectly (by requesting a resource that it contains that cannot be satisfied by any nodes in the partition under consideration). For example, a single node could have a unique resource, such as a disk drive array, that it could share with nodes in partitions.

  Such nodes, called *shared* or *partitionless* nodes, have none of the attributes of nodes in partitions (See TABLE 6-13 in Section 6.8.3 for partition attributes). Users cannot log in to nodes that do not belong to partitions.

6.6.1 Node Commands

TABLE 6-4 lists the commands at the node level of the `tmadmin` command hierarchy. You must be at the node level in order to execute these commands. To get to the node level, type `node` at the `tmadmin` prompt.

TABLE 6-4 Node-Level `tmadmin` Commands.

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

With the exception of `network`, these commands are described in Section 6.4. Issuing the `network` command at the node level drops you to the network level of the `tmadmin` hierarchy. For information on the network level of the `tmadmin` command hierarchy, see Section 6.7.

6.6.2 Creating Nodes

The object-monitoring daemon (`tm.omd`, described in Section 1.2.2) creates nodes automatically when it is started during the initialization process. That is, it adds node objects to the resource database. (For further information on initialization, see the *Sun HPC Software Installation Guide*.)

Note – Since the run-time environment (RTE) automatically creates node objects in its database during RTE initialization, creating node objects while the RTE is running is strongly discouraged. You need to create a node object only if you want to set up attributes for that node in the RTE database before the new node goes through RTE initialization. However, if you set the node's name to something other than the name that `tm.omd` acquires from each node during initialization (the name returned by Solaris's `gethostname`, minus any domain name extensions), the daemons will create an additional node object (using the name returned by `gethostname`) when they initialize, and you'll wind up with two objects for one node.

To create a node, you must be at the node context level of the `tmadmin` command hierarchy. If you are not already there, type `node` at the `tmadmin` prompt.

6.6.2.1 Prerequisite

A node must be loaded with Sun HPC software before it can be added to a Sun HPC System.

6.6.2.2 Viewing Existing Nodes

Before creating a new node, you might want to look at what nodes have already been created. To do this, use the `list` command.

```
[gauss] Node:: list
        node1
        node2
        node3
        node4
[gauss] Node::
```

6.6.2.3 Creating Node Objects

To create a node, use the `create` command followed by the name of the new node. The name of a node is that node's host name, excluding the domain name. (Creating node objects while the RTE is running is strongly discouraged — see Section 6.6.2)

For example:

```
[gauss] Node:: create node5
[gauss] N(node5)::
```

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

6.6.3 Configuring Nodes

You can configure the nodes of your Sun HPC System by setting and deleting attributes using the `set` and `unset` commands.

6.6.3.1 Node Attributes

TABLE 6-5 and TABLE 6-6 list the system-defined and administrator-defined node attributes. To see their current values, use `tmadmin`'s `show` command or the `tminfo` command. (See Section 6.1.)

TABLE 6-5 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).

TABLE 6-5 Node Attributes That Cannot Be Set by the System Administrator *(Continued)*

Attribute	Kind	Description
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 tasks 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.
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 6-6 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 tasks.
master	Boolean	Specify node on which the master daemons are running as an argument to tmrun.
max_batch_procs	Value	Maximum number of batch processes per node.
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 (see Note below).

TABLE 6-6 Node Attributes That Can Be Set by the System Administrator *(Continued)*

Attribute	Kind	Description
`partition`	Value	Partition of which node is a member.
`pfs_buffers`	Value	Number of buffers (32 Kbyte size) used by PFS I/O server.
`pfs_network`	Value	Network interface for client access to PFS I/O server.
`shmem_minfree`	Value	Fraction of swap space kept free for non-MPI use.

Note – The `name` attribute is a special case. The system sets `name` automatically, but `tmadmin` also allows the system administrator to set it. However, setting `name` does not change the host name on that node, so the next time the node is booted, the resource database will look for a node object with the old name. For this reason, we recommend that you not set a node's `name` attribute.

6.6.3.2 `enabled`

The attribute `enabled` is set by default when the daemons are first started on a node. Unsetting it prevents new tasks from being spawned on the node. A node that isn't enabled can still be part of an enabled partition. In essence, the partition size is shrunk by one.

6.6.3.3 `max_batch_procs`

To assign a limit to the number of batch processes to run per node, set the node's `max_batch_procs` attribute. If `max_batch_procs` is unset, batch processing is not allowed.

```
[gauss] N(node1):: set max_batch_procs=1
[gauss] N(node1)::
```

6.6.3.4 `max_locked_mem` and `min_unlocked_mem`

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 unpleasant side effects because it prevents that memory from being used by other processes on the node.

You can set these attributes to ensure that, on each node, a controlled amount of memory will be available for locking by parallel jobs. The RTE will then allow Sun MPI processes to lock a memory segment unless:

- That would cause the total amount of memory locked down by all processes on the system to exceed `max_locked_memory`.
- That would cause the amount of memory not locked down to dip below `min_unlocked_mem`.

If memory-locking attributes are set at both the partition and node levels, the node-level settings override the partition-level settings.

Note – Unless `max_locked_mem` or `min_unlocked_mem` (or both) are set, memory locking is disabled.

Solaris has two related tunable kernel parameters: `tune_t_minasmem` is similar to `min_unlocked_mem`, and `pages_pp_maximum` is is similar to `max_locked_mem`. The RTE parameters limit only parallel jobs (Sun MPI programs), while the kernel parameters limit all processes. Also, the kernel parameters' units are pages rather than Kbytes. For more information, see your Solaris documentation.

6.6.3.5 `partition`

To assign a node to a partition, set the node's partition attribute.

```
[gauss] N(node1):: set partition=Saturn
[gauss] N(node1)::
```

The partition must already exist. If it doesn't, `tmadmin` responds with an error message.

Note – Alternatively, you can assign a node to a partition by setting the partition's `nodes` attribute. See Section 6.8.3.

Setting a node's `partition` attribute always has the side effect of adding the node to that partition's `nodes` attribute.

```
[gauss] N(node1):: set partition=Saturn
[gauss] N(node1):: partition
[gauss] Partition:: current Saturn
[gauss] P(Saturn):: show
        set nodes = node1
        set name = Saturn
[gauss] P(Saturn)::
```

In other words, if you want to add a node to a partition, you can do it either from the node context (by setting the `partition` attribute) or from the partition context (by setting the `nodes` attribute). To see which nodes are members of a partition, you have to use the `show` command within the partition context.

Unsetting a node's `partition` attribute removes the node from that partition's `nodes` attribute, whether or not the partition is enabled. Note that which partition the node is removed from depends on the last value of the node's `partition` attribute.

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, and that name is always the name of the enabled partition, if one exists for that node.

In general, a node's `partition` attribute is meaningful only when the node is part of an enabled partition. At other times, it has only its most recent value, or no value at all if the node has not been assigned to a partition or if the most recent value is no longer allowed (which happens if an enabled partition is deleted, for example).

If a node is a member of an enabled partition, `tmadmin` prevents you from setting the `partition` attribute. If a node is a member of one or more disabled partitions only, setting the `partition` attribute adds the node to a new partition, while leaving it in the old one(s).

Compare the effects of setting a partition's `nodes` attribute. See Section 6.8.3.11.

6.6.3.6 shmem_minfree

To reserve a fraction of the space in `/tmp` for non-MPI use, set the `shmem_minfree` attribute.

For example, if the size of the `/tmp` file system is 1 Gbyte and `shmem_minfree` is set to the value of 0.2, when the free space on the `/tmp` file system drops below 200 Mbytes (1 Gbyte * 0.2), then Sun MPI programs using the MPI shared memory protocol will not be allowed to run.

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

Valid values are between 0.0 and 1.0. When this value is unset, it defaults to 0.1.

This attribute can be set on both nodes and partitions. The node attribute overrides the partition attribute if they are both set.

6.6.3.7 Setting Additional Attributes

Sun HPC Software does not limit you to the attributes listed. You can define new attributes as necessary.

If any node has special hardware, you will probably want to set an attribute for it. For instance, if node3 has a frame buffer attached, you may want to indicate that by creating an attribute.

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

Users can then use the attribute `has_frame_buffer` to request the frame buffer when they execute programs.

See Section 6.2 for restrictions on attribute names.

6.6.4 Configuring Nodes as PFS I/O Servers

For a node to function as a parallel file system (PFS) I/O server, you must add the following information to its node object definition in the RTE database:

- The amount of memory to reserve for I/O cache use
- Which network interface to use for all I/O communication

The amount of memory to set aside is specified in units of 32-Kbyte buffers. The number of such buffers you specify will depend, of course, on the amount of I/O traffic each server is likely to experience at any given time. As a rule of thumb, you should allocate at least five buffers for each file or directory that you expect will be in use concurrently. For example, if you expect an average of 10 users, each accessing an average of three files concurrently, you should allocate 150 buffers on each I/O node.

The network interface you specify must be common to all storage objects in the PFS. That is, its network must be accessible to all PFS I/O servers configured into the PFS. In Sun Ultra HPC Systems, this will ordinarily be the SCI network.

Note – This configuration step assumes that the nodes have already been created as described in Section 6.6.2. They may, but need not, be members of a partition.

The following example illustrates three nodes being configured for use as PFS I/O servers. For each, 128 buffers are allocated to I/O cache use and the SCI interface is specified as the communication network. Their names, `io-node8, io-node9`, and so on, were given to them when they were created.

```
[gauss]:: node current io-node8
[gauss] N(io-node8):: set pfs_buffers = 128
[gauss] N(io-node8):: set pfs_network = fa0
[gauss] N(io-node8):: current io-node9
[gauss] N(io-node9):: set pfs_buffers = 128
[gauss] N(io-node9):: set pfs_network = fa0
[gauss] N(io-node9):: current io-node10
[gauss] N(io-node10):: set pfs_buffers = 128
[gauss] N(io-node10):: set pfs_network = fa0
```

First, make an I/O node your current context and set the `pfs_buffers` and `pfs_network` attributes for that node. Then go to the next I/O node to be configured.

6.6.5 Deleting Nodes

If you physically remove a node from the Sun HPC System, you must delete the corresponding node object from the RTE resource database; otherwise the software will try to use it.

6.6.5.1 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.
- Wait for any jobs running on it to terminate, or stop them using the `tmkill` command, which is described in the *Sun HPC Software User's Guide*.

6.6.5.2 Using the `delete` Command

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

```
[gauss] N(node4):: delete
[gauss] Node::
```

6.7 Network Interfaces

The high performance, Sun-supported network interfaces are listed in TABLE 6-7.

TABLE 6-7 Network Interfaces

Symbol	Interface
idc*n*	HPC 10000 Inter-Domain Network
sci*n*	Dolphin Systems SCI
ba*n*	Sun ATM
hme*n*	SPARC Ethernet 100 Mbit/sec

Note – The *n* at the end of each interface symbol indicates an integer.

All nodes have an `lo0` interface. Its address is traditionally 127.0.0.1.

The trailing `0` in a network interface name can be any digit, depending on how many of the given network interfaces are installed on a node. Thus, you could have `le0` and `le1` on a node.

6.7.1 Network Interface Commands

TABLE 6-8 lists the commands at the network-interface level of the `tmadmin` NII command hierarchy. You must be at the network-level in order to execute these commands. To get to the network level, type `network` within the node context of `tmadmin`.

TABLE 6-8 Network Interface–Level `tmadmin` Commands

Command	Synopsis
`current` *network*	Set the context to the specified network for future commands.
`list`	List all the defined network interfaces.
`show` [*network*]	Show a network interface's attributes.
`set` *attribute*[=*value*]	Set the current network's attribute.
`unset` *attribute*	Delete the current network's attribute.
`up`	Move to the node level in the command hierarchy.
`top`	Move to the top level in the command hierarchy.
`echo` ...	Print the rest of the line on the standard output.
`help / ?`[*command*]	Show information about commands.

These commands are described in Section 6.4.

6.7.2 Configuring Network Interfaces

You can configure the network interfaces of your Sun HPC System by setting and deleting attributes using the `set` and `unset` commands.

Whenever you set or unset network-interface attributes with `tmadmin`, the RTE will cause the `tm.spmd` daemons to reread certain database information. Only the `tm.spmd` daemons of online nodes will reread the database. System administrators must issue the `rte.node reread` command on any nodes marked offline by `tm.watchd` to ensure that all `tm.spmd` daemons have been updated.

6.7.2.1 Network Interface Attributes

TABLE 6-9 lists the system-defined network interface attributes. To see their current values, use the `tmadmin show` command or the `tminfo` command. (See Section 6.1.)

TABLE 6-9 Network Interface Attributes.

Attribute	Kind	Description
`bcast_addr`	Value	Address to use to represent broadcasts to the network.
`ip_addr`	Value	IP address of the interface.
`mtu`	Value	Size of maximum transmission unit (MTU).
`name`	Value	Name of the network interface.
`netmask`	Value	Mask to specify how much of the address to reserve for subdividing networks.
`ranking`	Value	Ranking of the network interface compared with the others on the node.

For more information about a network interface's broadcast address or netmask, see the man page for the Solaris command `ifconfig`.

The system administrator can set the attributes `ranking` and `mtu`. The rest are set automatically by the system.

6.7.2.2 `ranking` and `mtu` attributes

You can tell the system which network interface to use by configuring the order of their rankings. Lower numbers correspond to higher ranking. The system uses network interface rankings to determine the order in which it tries to use network(s) for communication.

TABLE 6-10 Default `ranking` and `mtu` Sizes of Supported Network Interfaces

Symbol	Default Ranking	Default MTU	Interface
`idcn`	1	16384	HPC 10000 Inter-Domain Network
`scin`	2	16384	Dolphin Systems SCI
`ban`	3	8196	Sun ATM
`hmen`	10	4096	SPARC Ethernet 100 Mbit/sec

Default values are set for `ranking` and `mtu` (maximum transmission unit) in an initialization file supplied at installation time. The default ordering of network interface rankings is by order of expected use. See the *Sun HPC Software Installation Guide*.

You can adjust the efficiency of communications with the `mtu` attribute. Setting the `mtu` value too low will result in excessive communications overhead, wasting communications bandwidth. Setting the `mtu` value too high can cause inefficiencies, for example, because the timing of data transmission may interact poorly with the availability of communications buffers.

The best way to pick a good value for the `mtu` is to experiment: Run a Sun MPI program that sends large messages in a variety of patterns (point-to-point, one-to-many, many-to-one, many-to-many) and measure the time it takes for each communication pattern with different `mtu` sizes. The `mtu` values that were set during installation were determined through just such benchmarking. You may want to adjust the `mtu` for an interface if your users' Sun MPI programs have a particular predominant communications pattern.

To change the value of a network interface's `mtu`, set the `mtu` attribute (measured in bytes).

```
[gauss] N(node2) I(fa0):: set mtu=4096
[gauss] N(node2) I(fa0)::
```

The value of the `ranking` attribute is an integer. It must be consistent across all the nodes of a partition.

When you change `ranking` of a network interface on one node in a Sun HPC Cluster, `tmadmin` changes it automatically on all the other nodes in the cluster. If the new ranking conflicts with that of another interface, `tmadmin` automatically changes the other interface's ranking.

To change a network interface's ranking, set the `ranking` attribute.

```
[gauss] N(node2) I(fa0):: set ranking=1
[gauss] N(node2) I(fa0)::
```

6.7.2.3 Setting Additional Attributes

Sun HPC Software does not limit you to these predefined attributes. You can define new attributes as necessary.

You might, for example, want to define descriptive attributes, such as `no_trailers`, `multicast_capable`, or `externally_connected`, to provide users with more detailed information about the network interfaces.

See Section 6.2 for restrictions on attribute names.

6.8 Partitions

Partitions are logical collections of nodes that work cooperatively to run programs on the Sun HPC System. Tasks cannot run in multiple partitions; they can run in a single partition or a combination of a single partition and one or more nodes that are not assigned to any partition. Serial programs run on a single node of a partition. Parallel programs run on any number of nodes of a partition simultaneously. You must create a partition before you can run programs on your Sun HPC System.

You can create and configure partitions to meet the specific needs of your site.

There are three kinds of partitions:

- *Shared* – A shared partition allows simultaneous execution of multiple programs. This maximizes throughput of users' programs.
- *Dedicated* – A dedicated partition allows only one program to execute on it at a time. That program has unlimited access to all the resources of the partition, which minimizes execution time.
- *Login* – When users log in to the Sun HPC System, they are placed in a login partition, that is, a partition specifically configured to accept user logins. If you do not create a login partition, users will be unable to log in to the Sun HPC System.

Login and shared partitions perform load balancing. When you log in to a login partition, your login session is placed on the least-loaded node of the partition. When you execute a program on a shared partition using `tmrun` or `tmsub`, the RTE automatically runs it on the least-loaded nodes that satisfy any specified criteria.

Partitions are dynamic; that is, you can change them as needed after you create them. You can add nodes to a partition or remove them. You can change a partition's attributes. You can enable and disable partitions, so 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, as long as no node is a member of more than one enabled partition.

6.8.1 Partition Commands

TABLE 6-11 lists the commands at the partition level of the `tmadmin` NII command hierarchy. You must be at the partition level in order to execute these commands. To get to the partition level, type `partition` at the `tmadmin` prompt.

TABLE 6-11 Partition-Level `tmadmin` Commands

Command	Synopsis
`current` *partition*	Set the context to the specified partition for future commands.
`create` *partition*	Create a new partition with the given name.
`delete` [*partition*]	Delete a partition.
`list`	List all the defined partitions.
`show` [*partition*]	Show a partition's attributes.
`dump` [*partition*]	Show the attributes of a partition and its queues.
`set` *attribute*[=*value*]	Set the current partition's attribute.
`unset` *attribute*	Delete the current partition's attribute.
`queue`	Enter the queue command mode.
`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*.

With the exception of `queue`, these commands are described in Section 6.4. Issuing the `queue` command at the partition level drops you to the queue level of the `tmadmin` NII command hierarchy.

6.8.2 Creating Partitions

To create a partition, you must be at the partition level of the `tmadmin` NII command hierarchy. If you are not already there, type `partition` at the `tmadmin` prompt.

6.8.2.1 Prerequisites

- Nodes have to be added to the RTE database before you can add them to partitions.
- All nodes in a partition must be running the same version of Solaris.

6.8.2.2 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.

```
[gauss] Partition:: list
        Nile
        Volga
        Ganges
        Orinoco
[gauss] Partition::
```

6.8.2.3 Creating a Partition

To create a partition, use the `create` command, followed by the name of the new partition. See Section 6.2 for rules for naming a partition.

For example:

```
[gauss] Partition:: create Mississippi
[gauss] P(Mississippi)::
```

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, by setting the partition's `nodes` attribute. See Section 6.8.3. Alternatively, you can assign nodes to a partition by setting the `partition` attribute for each of the nodes that you want to assign to the partition.

6.8.3 Configuring Partitions

You can configure partitions by setting and deleting their attributes using the `set` and `unset` commands.

You can configure partitions along multiple dimensions, including shared or dedicated, interactive or batch, and login. TABLE 6-12 shows the attributes associated with some common partitions.

TABLE 6-12 Common Partitions and Their Attributes

Partition Type	Attributes	Default
Login	no_logins	not set
Shared	max_total_procs	not set or set greater than 1
Dedicated	max_total_procs no_logins	=1 set
Serial	no_mp_tasks	set
Parallel	no_mp_tasks	not set
Batch only	no_interactive_tasks	set
Interactive only		no queues defined on the partition

You can combine the attributes listed in TABLE 6-13 in any way that makes sense for your site. See Section 1.2.3 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 interactive use during the day, and dedicated partitions for batch use at night and on weekends. Perhaps you will choose to configure all nodes into one large partition for running huge parallel jobs at night.

Note – Some combinations of attributes don't make sense. For instance, if you allow both batch and interactive access on a partition and limit the number of total tasks allowed, you put interactive users at a disadvantage. If the maximum number of tasks is running on the partition and one task finishes, the queue will almost invariably beat an interactive user to the freed resource. Sun HPC Software is designed to allow you maximum flexibility, so it doesn't prohibit such configurations. You must use your own judgment.

6.8.3.1 Partition Attributes

TABLE 6-13 lists the predefined partition attributes. To see their current values, use the `tmadmin show` command or the `tminfo` command. (See Section 6.1.)

TABLE 6-13 Predefined Partition Attributes

Attribute	Kind	Description
default_queue	Value	Default queue.
enabled	Boolean	Set if the partition is enabled, that is, if it is ready to accept logins, jobs, or tasks.
max_batch_procs	Value	Maximum number of simultaneously running batch processes allowed on each node in the partition.
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 simultaneously running processes allowed on each node in the partition.
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 partition.
no_interactive_tasks	Boolean	Disallow interactive tasks.
no_logins	Boolean	Disallow interactive logins.
no_mp_tasks	Boolean	Disallow multiprocess parallel tasks.
nodes	Value	List of nodes in the partition.
queues	Value	List of queues feeding the partition.
shmem_minfree	Value	Fraction of swap space kept free for non-MPI use

The queues attribute is set automatically by the system when you create a queue on a partition. You can set the rest of the attributes.

6.8.3.2 default_queue

The tmsub command looks for a default queue to determine where to execute jobs. To set a default queue, set the default_queue attribute.

```
[gauss] P(Mars):: set default_queue=Deimos
[gauss] P(Mars)::
```

6.8.3.3 enabled

Set the enabled attribute to activate a partition. See Section 6.8.4. By default, the enabled attribute is not set when a partition is created.

6.8.3.4 max_batch_procs

To limit the number of simultaneously running batch processes allowed on a partition, set the max_batch_procs attribute.

```
[gauss] P(Earth):: set max_batch_procs=1
[gauss] P(Earth)::
```

Normal usage of batch systems is to run only one batch process (per node) at a time. The default value for `max_batch_procs` is zero. When you create a queue, you must assign a value greater than zero to `max_batch_procs` before the queue can submit jobs to the partition. If you `unset max_batch_procs`, the result is equivalent to setting `max_batch_procs` to zero — batch processing is disabled.

6.8.3.5 `max_locked_mem` and `min_unlocked_mem`

`max_locked_mem` and `min_unlocked_mem` limit the amount of shared memory available for being 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 unpleasant side effects because it prevents that memory from being used by other processes on the node.

You can set these attributes to ensure that, on each node, a controlled amount of memory will be available for locking by parallel jobs. The RTE will then allow you to lock a memory segment unless:

- That would cause the total amount of memory locked down by all processes on the system to exceed `max_locked_memory`.
- That would cause the amount of memory not locked down to dip below `min_unlocked_mem`.

If memory locking attributes are set at both the partition and node levels, the node level settings override the partition level settings.

Note – Unless `max_locked_mem` or `min_unlocked_mem` (or both) are set, memory locking is disabled.

Solaris has two related tunable kernel parameters: `tune_t_minasmem` is similar to `min_unlocked_mem`, and `pages_pp_maximum` is is similar to `max_locked_mem`. The RTE parameters limit only parallel jobs (Sun MPI and Sun HPF), while the kernel parameters limit all processes. Also, the kernel parameters' units are pages rather than kbytes. For more information, see your Solaris documentation.

6.8.3.6 `max_total_procs`

To limit the number of simultaneously running tmrun processes (and tmsub processes) allowed on a node, set the `max_total_procs` attribute.

```
[gauss] P(Venus):: set max_total_procs=10
[gauss] P(Venus)::
```

You can set `max_total_procs` if you want to limit the load on a partition. By default, max_total_procs is `unset` — there is no limit imposed by the RTE on the number of processes allowed on a node.

6.8.3.7 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.

```
[gauss] P(Venus):: set name=Mercury
[gauss] P(Mercury)::
```

See Section 6.2 for rules for naming a partition.

6.8.3.8 no_interactive_tasks

To prohibit interactive tasks from running on a partition, that is, to limit it to access through batch queues, set the `no_interactive_tasks` attribute.

```
[gauss] P(Saturn):: set no_interactive_tasks
[gauss] P(Saturn)::
```

6.8.3.9 no_logins

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

```
[gauss] P(Jupiter):: set no_logins
[gauss] P(Jupiter)::
```

6.8.3.10 no_mp_tasks

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

```
[gauss] P(Uranus):: set no_mp_tasks
[gauss] P(Uranus)::
```

6.8.3.11 nodes

To specify the nodes in a partition, set the partition's `nodes` attribute.

```
[gauss] P(Neptune):: set nodes=node1
[gauss] P(Neptune):: show
        set nodes = node1
        set enabled
[gauss] P(Neptune)::
```

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.

```
[gauss] P(Neptune):: set nodes=+node2 node3 node4
[gauss] P(Neptune):: show
        set nodes = node1 node2 node3 node4
        set enabled
[gauss] P(Neptune)::
```

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 Neptune all nodes whose names are alphabetically greater than or equal to node1 and less than or equal to node6:

```
[gauss] P(Neptune):: set nodes = node1:node6
[gauss] P(Neptune)::
```

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 Neptune affects the partition attribute of node2:

```
[gauss] P(Neptune):: node node2
[gauss] N(node2):: show
        set partition = Neptune
[gauss] 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, tmadmin returns an error message.

```
[gauss] P(Neptune):: show
        set nodes = node1 node2 node3 node4 node5 node6
        set enabled
[gauss] P(Neptune):: current Venus
[gauss] P(Venus):: set enabled
[gauss] P(Venus):: set nodes=node1
tmadmin: node1 must be removed from Neptune before it can be added to
Venus
```

Unsetting the nodes attribute of an enabled partition has the side effect of unsetting the nodes' partition attributes. Unsetting the nodes attribute of a disabled partition removes the nodes from the partition but does not change the nodes' partition attributes.

Compare the effects of setting a node's partition attribute. See Section 6.6.3.

6.8.3.12 shmem_minfree

To reserve a fraction of the space in `/tmp` for non-MPI use, set the shmem_minfree attribute.

For example, if the size of the /tmp file system is 1 Gbyte and `shmem_minfree` is set to the value of 0.2, when the free space on the `/tmp` file system drops below 200 Mbytes (1 Gbyte * 0.2), then Sun MPI programs using the MPI shared memory protocol will not be allowed to run.

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

Valid values are between 0.0 and 1.0. When this value is unset, it defaults to 0.1.

This attribute can be set on both nodes and partitions. The node attribute overrides the partition attribute if they are both set.

6.8.3.13 Setting Additional Attributes

The RTE does not limit you to these predefined attributes. You can define new attributes as necessary.

For example, you might want to make it easy for users to recognize a dedicated partition without having to remember which combination of attributes defines a dedicated partition. You can do that by creating an attribute called `dedicated`.

```
[gauss] P(Pluto):: set max_total_procs=1
[gauss] P(Pluto):: set no_logins
[gauss] P(Pluto):: set no_interactive_tasks
[gauss] P(Pluto):: set dedicated
[gauss] P(Pluto)::
```

Then users can use the `tminfo` or `tmadmin show` command to see whether a partition is dedicated before they submit programs for execution.

See Section 6.2 for restrictions on attribute names.

6.8.4 Enabling Partitions

A partition must be enabled before users can log in to it or run programs on it.

6.8.4.1 Prerequisite

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

6.8.4.2 Setting `enabled`

To enable a partition, set its `enabled` attribute.

```
[gauss] P(Colorado):: set enabled
[gauss] P(Colorado)::
```

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, `tmadmin` prints an error message.

```
[gauss] P(Big):: show
        set nodes = node1 node2 node3
        set enabled
[gauss] P(Big):: current Small
[gauss] P(Small):: show
        set nodes = node1
[gauss] P(Small):: set enabled
tmadmin: Big/node1: partition resource conflict
```

6.8.5 Disabling Partitions

Disable a partition when you don't want programs to run on it.

6.8.5.1 Recommendations

After disabling a partition, you should either wait for any jobs running on the partition to terminate or stop them using the `tmkill` command, which is described in the *Sun HPC Software User's Guide*.

6.8.5.2 Unsetting `enabled`

To disable a partition, unset its `enabled` attribute.

```
[gauss] P(Colorado):: unset enabled
[gauss] P(Colorado)::
```

Now the partition can no longer be used.

If any jobs are running on a partition when it is disabled, they continue to run.

6.8.6 Deleting Partitions

Delete a partition when you don't plan to use it anymore.

When you delete a partition, any queues belonging to that partition are deleted automatically.

6.8.6.1 Recommendation

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.

6.8.6.2 Deleting a Partition

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

```
[gauss] P(Orinoco):: delete
[gauss] Partition::
```

6.9 Queues

A queue manages a list of jobs that are running or waiting to run on a partition. Batch jobs are submitted to queues via the `tmsub` command, described in the *Sun HPC Software User's Guide*. Jobs submitted to a batch queue are executed in order of priority, with jobs of equal priority executed on a first-come, first-served basis.

6.9.1 Queue Policy

If a job arrives at the head of a queue and the resources it needs are not available, the behavior depends on the queueing policy in place for that queue. The `policy` attribute can have one of two values:

- `block` – The job waits at the head of the queue until the resources become available or it is removed from the queue. If `max_batch_procs` is equal to 1, the job waiting at the head of the queue blocks the others. If the partition's `max_batch_procs` attribute is set to a value greater than 1, other jobs queued behind this job may be scheduled to run, while this job continues to wait for sufficient resources.
- `requeue` – The job waits in place, letting other jobs run while waiting for resources.

To remove a job from a queue, use the `tmkill` command, which is described in the *Sun HPC Software User's Guide*.

The following rules govern queues in a Sun HPC System:

- A queue is associated with a single partition.
- A queue cannot be moved from one partition to another.
- A partition can be fed from multiple enabled queues.
- Programs are executed in order of priority and submission, regardless of queue, subject to the availability of resources and the value assigned to `max_batch_procs`.

Queues on different partitions can have the same name. When a user gives a non-unique queue name to `tmsub`, `tmsub` uses the rules listed in Section 6.5.2 to determine which queue to submit the job to.

Normally, you shouldn't have to define more than one queue per partition. However, the RTE does not prevent you from defining more. You might, for instance, choose to define two queues with different queueing policies on a partition and enable and disable them separately.

6.9.2 Queue Commands

TABLE 6-14 lists the commands at the queue level of `tmadmin`. You must be at the queue level in order to execute these commands. To get to the queue level, type `queue` within the `partition` context of `tmadmin`.

TABLE 6-14 Queue-Level `tmadmin` Commands.

Command	Synopsis
`current` *queuename*	Set the context to the specified queue for future commands.
`create` *queuename*	Create a new queue with the given name.
`delete` [*queuename*]	Delete a queue.
`list`	List all the defined queues on the current partition.
`show` [*queuename*]	Show a queue's attributes.
`set` *attribute*[=*value*]	Set the current queue's attribute.
`unset` *attribute*	Delete the current queue's attribute.
`up`	Move to the partition level in the command hierarchy.
`top`	Move to the top level in the command 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*.

These commands are described in Section 6.4.

6.9.3 Creating Queues

To create a queue, you must be at the queue context level of the `tmadmin` NII command hierarchy, within the context of a specific partition. If you are not already there, type `queue` at the `tmadmin` prompt within the context of a partition.

6.9.3.1 Prerequisite

You cannot create a queue until you have created the partition to which it will belong.

6.9.3.2 Viewing Existing Queues

Before creating a new queue, you might want to see whether any queues have already been created for a particular partition. To do this, use the `list` command.

```
[gauss] P(Mississippi) Queue:: list
        Missouri
[gauss] P(Mississippi) Queue::
```

6.9.3.3 Creating a Queue

To create a queue, use the `create` command, followed by the name of the new queue. See Section 6.2 for rules for naming a queue.

For example:

```
[gauss] P(Mississippi) Queue:: create Ohio
[gauss] P(Mississippi) Q(Ohio)::
```

The `create` command automatically changes the context to that of the new queue and sets the `queues` attribute of the associated partition.

6.9.4 Configuring Queues

You can configure queues on your Sun HPC System by setting and deleting attributes using the `set` and `unset` commands.

6.9.4.1 Queue Attributes

TABLE 6-15 lists the predefined queue attributes. To see their current values, use the `tmadmin show` command or the `tminfo` command. (See Section 6.1.)

TABLE 6-15 Queue Attributes

Attribute	Kind	Description
`default_priority`	Value	Priority value to be used when no `-P` option given to `tmsub`.
`enabled`	Boolean	Set if the queue is enabled, that is, if it is ready to accept tasks.
`maxpri`	Value	The highest priority allowable for the queue.
`minpri`	Value	The lowest priority allowable for the queue.
`name`	Value	Name of the queue.
`partition`	Value	Partition to which the queue is allowed to submit tasks.
`policy`	Value	Policy determining disposition of jobs exceeding current resources.
`running`	Boolean	Set if the queue is running, that is, if it is allowed to submit tasks to the partition.

The `partition` attribute is set automatically because a queue is created only within the context of a specific partition. You can set the rest of the attributes.

6.9.4.2 enabled

Set the enabled attribute to allow a queue to start accepting jobs. See Section 6.9.5.

6.9.4.3 name

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

```
[gauss] P(Colorado) Q(Batch):: set name=Night
[gauss] P(Colorado) Q(Night)::
```

See Section 6.2 for rules for naming a queue.

6.9.4.4 policy

Set the policy attribute to determine how the RTE treats batch jobs whose requirements exceed currently available resources. This attribute cannot be unset. The default policy is block. See Section 6.9.1.

To see the current policy, type

```
% tminfo -Q
```

6.9.4.5 running

Set the running attribute to allow a queue to start submitting jobs to a partition. See Section 6.9.5.

6.9.4.6 Setting Additional Attributes

The RTE does not limit you to the attributes listed. You can define new attributes as necessary.

For example, if you have different sets of queues for use during the day, at night, and on weekends, you might want to add a descriptive attribute that holds information about when a queue is enabled.

```
[gauss] Q(Missouri):: set when_used=night
[gauss] Q(Missouri):: current Ohio
[gauss] Q(Ohio):: set when_used=weekdays_9-5
[gauss] Q(Ohio)::
```

Users can then use tminfo or show to get schedule information.

Note – The example above does not cause the queues to be enabled and disabled at certain times. The attributes simply indicate to users when you intend to enable and disable the queues.

See Section 6.2 for restrictions on attribute names.

6.9.5 Enabling Queues

There are two parts to enabling a queue. Its `enabled` attribute controls whether users are allowed to submit jobs to the queue. Its `running` attribute controls whether the queue can submit jobs to a partition.

6.9.5.1 Setting `enabled`

To make a queue start accepting submitted jobs, set its `enabled` attribute.

```
[gauss] Q(Ohio):: set enabled
[gauss] Q(Ohio)::
```

6.9.5.2 Setting `running`

To make a queue start submitting jobs to a partition for execution, set its `running` attribute.

```
[gauss] Q(Ohio):: set running
[gauss] Q(Ohio)::
```

Note – A queue can submit jobs to a partition only when the partition's `max_batch_procs` attribute is greater than zero. Since the attribute's default is zero, you must set its value before the queue can submit jobs.

6.9.6 Disabling Queues

There are two parts to disabling a queue.

6.9.6.1 Unsetting `enabled`

To make a queue stop accepting jobs submitted to it, unset the `enabled` attribute.

```
[gauss] Q(Ohio):: unset enabled
[gauss] Q(Ohio)::
```

Any jobs that are waiting in the queue when the `enabled` attribute is unset remain in the queue. If the `running` attribute is still set, the queue can still submit them to a partition for execution. If the `running` attribute is unset, the jobs wait in the queue until it is reactivated.

6.9.6.2 Unsetting `running`

To make a queue stop submitting jobs to a partition for execution, unset the `running` attribute.

```
[gauss] Q(Ohio):: unset running
[gauss] Q(Ohio)::
```

Any jobs in the queue that are running when the `running` attribute is unset continue running, but no subsequent jobs are started.

6.9.7 Deleting Queues

Delete a queue when you don't plan to use it anymore.

6.9.7.1 Recommendations

Before deleting a queue, you should:

1. Disable it by unsetting its `enabled` and `running` attributes.

   It is possible to delete a queue without first disabling it, but it is not advisable.

2. Make sure it is empty. You can wait for running or waiting jobs to finish, or you can remove them. To remove a waiting job or to stop a running job, use the `tmkill` command. See the *Sun HPC Software User's Guide* for information about `tmkill`.

   It is possible to delete a queue that has jobs running or waiting in it, though it is preferable to remove these jobs first. Running jobs continue to execute. Waiting jobs are deleted.

6.9.7.2 The `delete` Command

To delete a queue, use the `delete` command in the context of the queue you plan to delete.

```
[gauss] Q(Night):: delete
[gauss] Queue::
```

Jobs that are running in a queue when it is deleted continue running. Jobs that are waiting in the queue are deleted.

6.10 Parallel File Systems

Sun HPC parallel file systems (PFSs) are based on three abstract entities: PFS I/O servers, storage objects, and the parallel file systems themselves.

A PFS I/O server is simply a Sun Ultra HPC node that has been configured in the run-time environment (RTE) with additional attributes that enable it to participate with other PFS I/O servers in the distributed storage and retrieval of parallel file systems. A PFS I/O server also differs from other nodes in that it runs a PFS I/O daemon, which manages the storage and retrieval of data to and from its portion of the parallel file systems. Certain other attributes are also defined for the file system, such as the amount of memory on each PFS I/O server that is reserved for I/O buffering and the network interface assigned to each PFS I/O server. See Section 6.6.4, “Configuring Nodes as PFS I/O Servers.”

A storage object is an RTE abstraction that represents one portion of a parallel file system—that is, the portion of a file system that resides on a particular storage device and is managed by the PFS I/O daemon associated with that device. To create a specific parallel file system, the RTE is given a name and a set of storage objects to be associated with that file system.

Note – If you want to use only part of a disk partition, you must create a standard file system on that partition and identify a file in that partition to use as the storage object using `tmadmin`’s disk context. (See Section 6.11.3).

File system naming, storage object associations, and the other configuration tasks involved in creating and defining parallel file systems are all done using `tmadmin` commands within the PFS context. These procedures are described in this section.

For information about starting and stopping a PFS, see Section 7.1, “Starting and Stopping PFS.”

6.10.1 PFS Commands

TABLE 6-16 lists the commands at the PFS level of `tmadmin`. You must be at the PFS level to execute these commands. To get to the PFS level, type `pfs` at the `tmadmin` prompt.

TABLE 6-16 PFS-Level `tmadmin` Commands

Command	Synopsis
`current` *filesystem*	Set the current context to the specified parallel file system for future commands.
`create` *filesystem*	Create a new parallel file system with a given name.
`delete` [*filesystem*]	Delete a parallel file system.
`list`	List all defined parallel file systems.
`show` [*filesystem*]	Show a parallel file system's attributes.
`dump` [*filesystem*]	Show the attributes of a parallel file system and its disks.
`set` *attribute*[=*value*]	Set the current parallel file system's attributes.
`unset` *attribute*	Delete the specified attribute of the current parallel file system.
`disk`	Enter the disk command mode.
`up`	Move to the next-higher level in the command hierarchy.
`top`	Move to the top level in the command hierarchy.
`echo` ...	Print the rest of the line on the standard output.
`help` [*command*]	Show information about command(s). Synonym for `?`.

6.10.2 Parallel File System Attributes

TABLE 6-17 Parallel File System Attributes

Attribute	Kind	Description
`disks`	Value	List of storage objects in PFS.
`name`	Value	Name of PFS.

6.10.3 Viewing Existing Parallel File Systems

Use the `list` command to see what parallel file systems already exist.

```
[gauss]:: pfs
[gauss] F:: list
         teams
         presidents
         flavors
```

In this example, three parallel file systems are already defined in the RTE database.

6.10.4 Creating New Parallel File Systems

Creating a parallel file system is done in three steps.

- Identify which nodes will function as PFS I/O servers for the new PFS, and update their node objects in the RTE database with two pieces of additional information. See Section 6.6.4 for details.
- Use the `tmadmin create` command to name a new file system and to add that name to the RTE database. See Section 6.10.5 for details.
- Describe the new PFS's storage objects in the RTE database. See Section 6.11 for details.

Note – To create a parallel file system, you must be at the PFS context level of the `tmadmin` command hierarchy. If you are not already there, type `pfs` at the `tmadmin` prompt.

6.10.5 Naming the New Parallel File System

Use the `create` command to assign a name to a new parallel file system. The name is added to the RTE database. For example, to add the PFS `cities` to the database:

```
[gauss] N(io-node10):: top
[gauss]:: pfs
[gauss] F():: create cities
[gauss] F(cities)::
```

The `create` command automatically changes your context to that of the new parallel file system named by the command.

The name assigned to the file system must also be included in the path name specification of any constituent directory or file. PFS path names take the form `pfs:`*filesystem*`:`*pathname*. For example, the file `/users/clovis/paris` in the PFS `cities` would be named `pfs:cities:/users/clovis/paris`.

The new parallel file system `cities` now exists in the database, but is not yet ready for use. To complete the process, you must describe a set of set of storage objects that belong to the parallel file system `cities`.

6.11 Disks

The final step in configuring a parallel file system is to describe its storage objects to the RTE. The information required by the RTE for each storage object consists of:

- the rank of the storage object within the PFS
- the name of the I/O server
- the name of the storage device

These steps must be carried out within the context of each storage object being described.

Use the `disk create` command within the context of each PFS to specify a storage object, identified by its rank in the PFS. This *rank* determines the storage object's place in the ordering of PFS file data blocks. That is, when a PFS file is distributed across a set of storage objects, the blocks are sequenced to the storage objects in increasing rank order. Likewise, when reading a PFS file, the blocks are accessed from storage objects in order of increasing rank. The rank of the storage object is synonymous with the rank of the paired I/O server node and storage device.

Note – The RTE regards rank values as relative, not absolute. For example, if you have three storage objects, you can assign them ranks of 1, 15, and 30 or 2, 4, and 7. RTE behavior is the same for either set of values. Note also that the `tmadmin show` command displays the rank of a storage object as the value of the *name* attribute.

6.11.1 Disk Commands

TABLE 6-18 lists the commands at the disk level of the `tmadmin` command hierarchy.You must be at the disk level in order to execute these commands. To get to the disk level, type `disk` within the parallel file system context of `tmadmin`.

TABLE 6-18 Disk-Level Commands

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

6.11.2 Configuring Disk Systems

You can configure the parallel file system of your Sun HPC System by setting and deleting attributes using the `set` and `unset` commands.

6.11.3 Creating Disks

When specifying the storage object, use the full path name. For example, if the storage object is a device, use the the form `/dev/rdsk/c0t1d0s2` rather than `/dev/dsk/c0t1d0s2`. If the storage object is a file, use the form `/usr/local/pfs_storage`.

The following example uses the same three I/O servers that were configured in Section 6.6.4. Note that it begins in the context of the new PFS, `cities`.

```
[gauss] F(cities):: disk create 1
[gauss] F(cities) D(001):: set node = io-node8
[gauss] F(cities) D(001):: set device = "/dev/rdsk/c0t1d0s2"
[gauss] F(cities) D(001):: disk create 15
[gauss] F(cities) D(015):: set node = io-node9
[gauss] F(cities) D(015):: set device = "/dev/rdsk/c0t1d0s2"
[gauss] F(cities) D(015):: disk create 30
[gauss] F(cities) D(030):: set node = io-node10
[gauss] F(cities) D(030):: set device = "/dev/rdsk/c0t1d0s2"
```

TABLE 6-19 lists the predefined disk attributes. To see their current values, use the `tmadmin show` command or the `tminfo` command. (See Section 6.1.)

TABLE 6-19 Disk Attributes

Attribute	Kind	Description
`name`	Value	Rank of storage object (integer)
`device`	Value	Path to device (or file) holding a storage object of the PFS.
`node`	Value	Host for this storage object.

6.11.4 Deleting Disks

6.11.4.1 Recommendations

Stop all I/O daemons (IOD) before deleting a `Disk`. To shut down the IODs, issue the `tmfsstop` command. For further information on `tmfsstop`, see Section 7.1.4.1.

6.11.4.2 Using the `delete` Command

To delete a disk, use the `delete` command within the context of the disk you plan to delete.

```
[gauss] F(cities) D(015):: delete
[gauss] F(cities) Disk::
```

CHAPTER 7

PFS Operations and Utilities

7.1 Starting and Stopping PFS

7.1.1 Starting PFS

PFS requires an I/O daemon to be running on each I/O server. At boot time, these daemons will start automatically on those nodes configured as I/O servers. 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 start
```

Once the daemons are running, execute the `tmmkfs` and `tmmount` commands in each parallel file system. For example, if `cities` is a 512-Mbyte file system:

```
# /opt/SUNWhpc/bin/tmmkfs cities 512M
# /opt/SUNWhpc/bin/tmmount cities
```

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

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

7.1.2 Stopping PFS

Stopping PFS is a two-step process.

To unmount all PFSs, use `tmumount`:

```
# /opt/SUNWhpc/bin/tmumount -a
```

Or, to unmount a single file system, such as `cities`:

```
# /opt/SUNWhpc/bin/tmumount cities
```

Once you've stopped the PFS(s), you must stop the I/O daemons on each I/O server by running the `sunhpc.pfs stop` command on each node. For example:

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

Note – Unmounting PFSs and stopping I/O daemons happens automatically when a node is shut down or rebooted.

If it is not possible to log on to a node to run the shutdown script (that is, the command `sunhpc.pfs stop`), the utility `tmfsstop` can be used to remotely shut down all I/O daemons running on the node(s). Use the option `-f` to forcibly shut down an I/O daemon even if it still has file systems mounted. See the `tmfsstop` man page for additional information.

7.1.3 PFS or I/O Daemon Failures

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

7.1.4 PFS Utilities

PFS provides a number of utilities to assist the system administrator and users in maintaining and manipulating their parallel file systems. This section describes these utilities.

7.1.4.1 Administration Utilities

The following utilities closely resemble their Solaris counterparts, except they are able to operate on file systems that are distributed across multiple storage devices. These utilities require that all I/O daemons associated with the affected PFS to be running. If any such I/O daemon is not running, the utilities will fail, reporting the reason for the failure.

See the applicable man pages for detailed descriptions of these utilities.

TABLE 7-1 PFS Administration Utilities.

Utility	Description
`tmmkfs`	Make a PFS file system.
`tmfsck`	Check and repair a PFS file system.
`tmmount/tmumount`	Mount and unmount a PFS file system.
`tmfsstop`	Shut down all PFS I/O daemons.

7.1.4.2 General Utilities

With two exceptions, the following utilities are the PFS equivalents of Solaris utilities with similar names. The exceptions are `tmimport` and `tmexport`, which do not have Solaris equivalents. All these utilities facilitate various file handling tasks that users and system administrators commonly perform.

TABLE 7-2 PFS General Utilities.

Utility	Description
`tmcd`	Set the PFS current working directory.
`tmchgrp`	Change the group of a PFS file or directory.
`tmchmod`	Change the access protection bits of a file.
`tmchown`	Change the owner of a PFS file or directory.
`tmcmp`	Compare the contents of two PFS files.
`tmcp`	Copy a PFS file or directory within the PFS system.
`tmimport/tmexport`	Move data between PFS and Solaris file systems. (Use for backup and restore of PFS data)
`tmln`	Create a hard link to a PFS file (PFS does not support symbolic links).
`tmls`	List the contents of PFS directories.
`tmmkdir`	Create a PFS directory.
`tmmv`	Move a PFS file or directory to another location in the PFS system.
`tmpwd`	Print the current working directory.
`tmrm`	Remove a PFS file.
`tmrmdir`	Remove a PFS directory.

See the applicable man pages for detailed descriptions of these utilities.

CHAPTER 8

Accounting

At some sites a system administrator is responsible for billing users for their resource usage. Sun HPC Software includes an accounting facility that measures the resources used per job.

8.1 The Accounting Record

Whenever a task exits, an entry is added to the accounting record, `/var/hpc/tmacct.txt`, on the master node. The accounting record consists of several lines, as in this example:

```
Mon Jan 22 13:29:03 1996 t8 rivera staff /usr/rivera/a.out 1162 10
    ultra-node2 20513 1 0.855692979 0.23264999 0.27277044 0 7313 E 0
    ultra-node3 14635 0 0.809145888 0.27309328 0.32709915 0 8339 E 0
    Totals: 1.664838867 0.50574327 0.59986959 0 15652
```

The first line is a task summary. It contains the date, the time of day, the task ID (tid), the user name and group of the invoker, and the program name and argument list.

Following the task summary line are summary lines for each process in that task. These lines contain the host name, process ID (pid), rank, real (elapsed) time, user time, system time, I/O blocks, I/O characters, process state, and exit status (all process time values are measured in seconds).

The last line lists the total amounts for various accounting records. This line contains the word *Totals*: followed by the total real, user, and system times; total I/O blocks; and total I/O characters used by all the processes in the task.

The process state is a one-character field before the exit status. It determines the meaning of the exit status field, as indicated in TABLE 8-1.

TABLE 8-1 Process States and Exit Status Field Meanings

Process State	Meaning	Status Meaning
E	Exited	Exit status
S	Signaled	Signal number
F	Failed	Undefined
O	Orphan	Undefined
U	Unknown	Undefined
R	Running	Undefined

8.2 Managing Accounting

You can write your own routine to periodically harvest the `tmacct.txt` files. A sample script, `tmreport.pl`, in Perl5, is provided with the Sun HPC Software Foundation package. This script writes a summary by user and by node, of the `tmacct.txt` files presented to it. The `tmacct` utility synchronizes accounting records, telling the RTE master daemon (`tm.mpmd`) to append accounting information for all running tasks to the `tmacct.txt` file. The `tm.mpmd` daemon then renames `tmacct.txt` to `tmacct.txt.1` and creates a new (empty) `tmacct.txt` file. If a file named `tmacct.txt.1` already exists, it will be renamed to `tmacct.txt.2`, as well as incrementing the numerical parts of the file names of older `tmacct.txt` files.

Note – The totals produced by `tmreport.pl` are inexact. This is due to rounding error.

The sample script, `tmreport.pl`, is written in Perl5. If you want a different report, you can replace `tmreport.pl` with your own routine. The Standard Version of the Perl sources can be acquired at the Perl URL:

`http://www.perl.com.`

Given the sample script, the system administrator can write a script to handle accounting, pruning `tmacct.txt` files, and so forth.

8.2.1 Suspending Accounting

You can call the `tmacct` utility to temporarily suspend accounting so that it can obtain a snapshot of system usage, since the previous `tmacct` invocation (or since startup).

When you issue the `tmacct` command, it:

- Suspends accounting, so that no other process is allowed to exit and write an accounting record.
- Finds the resource usage for each running process and writes it to the resource database, subtracting that usage from any subsequent accounting for the process.
- Writes information from the resource database into the `tmacct.txt` log file.
- Rotates the numbering of `tmacct.txt` files, as described in Section 8.2.
- Ends the suspension of accounting.

CHAPTER 9

Troubleshooting

9.1 Scheduled Maintenance

System administrators can control error-causing conditions by performing periodic maintenance. Examples of useful maintenance are cleaning up defunct tasks and removing excess files. Periodic use of diagnostic software is also recommended.

9.1.1 Cleaning Up Defunct RTE Tasks

One form of administrative precaution involves cleaning up defunct tasks. There are several types of such tasks:

- Tasks that have exited, but still appear in `tmps` output
- Tasks that have not terminated, but need to be removed
- Tasks that have orphan processes

Note – To remove jobs from LSF, see the *LSF Administrator's Guide* for information about `bjob`, and `bkill`.

9.1.1.1 Removing RTE Tasks that have Exited

If a task exits, some (if not all) of its processes must be in transit from a RUNNING state to a final state. While a task is exiting, a noticeable interval of time must elapse, as all of the processes of the task make the transition to a final state. Situations can arise in which all of a task's processes have reached a final state, but the task object has not been removed from the RTE database.

Examples of exited tasks are:

- A process (identified by `tmps`) in the EXIT, SEXIT, FAIL, or CORE states
- A Prism *main* window that won't close or exit

If you see a task in one of these defunct states:

1. **Execute `tmps` again in case the RTE has had time to update the database (and remove the task).**
2. **Kill the task (if it is still running), using its task ID (tid):**

   % **`tmkill`** *tid*
3. **If necessary, remove the task object manually from the RTE database:**

   If `tmps` continues to report the killed task, remove the task object from the RTE database using (as root, from the master node):

   # **`tmkill -C`**

9.1.1.2 RTE Tasks that Have Not Terminated

One example of the second type of defunct task is the class of tasks that are waiting for signals from processes on nodes that have gone off line. The `tmps` utility displays such tasks in states such as RUNNING, EXITING, SEXITING, or COREING. If the task-killing option of `tm.watchd` (–Yk) is enabled, the RTE will take care of such situations automatically. This section assumes that the task-killing option of `tm.watchd` is not enabled.

Kill the task using:

% **`tmkill`** *tid*

There are several variants of the `tmkill` command (documented on the `tmkill` man page), similar to the variants of the Solaris `kill` command. You may also use:

% **`tmkill -9`** *tid*

or

% **`tmkill -I`** *tid*

If these do not succeed, execute `tmps -pe` to display the unresponsive processes. Then, execute the Solaris `ps` command on the each of the nodes that are listed. If those processes still exist on any of those nodes (they may have died on some nodes) you can remove them using `kill -9` *pid*. (Note that `kill` is a Solaris command).

Once you have eliminated all the defunct tasks, data about the tasks may remain in the RTE database. Use `tmkill -C` (as root, from the master node) to clean this up.

9.1.1.3 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. Such processes can still tie up resources and use CPU cycles. Unless the node resumes communication with the RTE daemons (in which case the watch daemon will kill the `ORPHAN` processes), you will have to kill the processes manually using the Solaris `kill` command.

Symptoms of orphaned processes can be detected by examining error log files (or `stdout`, if you're running from a terminal) and by searching for errors such as `RPC: cannot connect`, or `RPC: timout`. These errors will appear under `user.err` priority in `syslog`.

9.1.2 Removing Excess Files

Excess files can accumulate in `/var/tmp` directories. These are files generated by the C and F77 compilers. These directories can grow unnecessarily large, wasting storage space.

9.1.3 Diagnostics

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

9.1.3.1 Network Diagnostics

You can use `/usr/sbin/ping` to check whether you can connect to the network interface on another node. For example:

```
% ping dev-node11
```

will test (over the default network) the connection to dev-node11.

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 dev–node12
```

sends 100 small packets to dev–node12.

You can use tminfo –N, or if the RTE 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.

9.1.3.2 Interval Diagnostics

These diagnostics all accept a numerical option, specifying an interval. Without an interval number, the diagnostics output an average 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 is useful to run these diagnostics periodically, monitoring their output for multiple intervals.

9.2 Error Conditions and Troubleshooting Tips

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

9.2.1 Error Messages

- The error message

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

  usually indicates that all the nodes in an RTE partition are marked down (that is, their node daemons are not running). This could happen, for example, if RTE was unable to check out its licenses. The error may also indicate an error in the construction of an RRS.

- Under certain circumstances, when a user attempts to kill a task, it can cause the RTE to log error messages of the following form on the master node:

  ```
  Aug 27 11:02:30 ops2a tm.rdb[462]: Cond_set: unable to
  connect to ops2a/45126: connect: Connection refused
  ```

  If these can be correlated to tasks being killed (e.g. by looking at the accounting logs for tasks that were signaled during this time), then these errors can be safely ignored.

- The error message

  ```
  tmrun: unique partition: No such object
  ```

  indicates that no partitions have been set up.

9.2.2 Troubleshooting Tips

- To ensure that you don't damage an NFS file system (at the current working directory level) when running tasks with a number of processes (`np`) greater than one, see the manpage for your user shell or for the Solaris `limit` command for information about limiting core dumps to zero. For example, for `csh`-based shells the command is:

  ```
  limit coredumpsize 0
  ```

- After client restart, the RTE resource database (`rdb`) does not remove missing interfaces, but they will be listed as *down* with `tminfo –Nv`.

- The content of the `/var/adm/messages` file is local to each node. If there is a problem with a daemon, error messages regarding this daemon will appear only in the `/var/adm/messages` file on the node where the daemon runs. Note that you may have configured `syslog` to operate differently.

- Use shell I/O redirection instead of `tmrun –I` options whenever possible. Using shell redirection will reduce the likelihood of problems involving standard I/O.

- If `tmrun` is signaled prematurely, it exits without signalling the task's processes. Use `tmkill –9` *tid* to kill such a task.

- The RTE does not pass supplemental group ID information to remote processes — you must use the `-G` *gid* option with `tmrun` or `tmsub` to run with the group permissions of that group. You must be a member of that group.
- *RPC timeouts* – RTE 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. This is different from RPC timeouts occurring in the RTE daemons themselves. These are (by default) logged to `/var/adm/messages`.

Note – If you have set the `logfile` RTE attribute (using `tmadmin` at the top- or server-level), errors are not logged using `syslog`, and all error messages generated by user code will be logged and not dropped.

  RTE RPC timeouts in user code are generally not recoverable. The task might continue to run, but processes probably won't be able to communicate with other processes. There are two ways to deal with this:

  - Enable the `tm.watchd` task killing option (`-Yk`) which will automatically kill tasks 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 `tmkill` to kill the associated task 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 adjust the limit on the RTE's `shmem_minfree` attribute, increasing the value.
- If you execute `tmkill` 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. These files are of the form:

  `/tmp/.tmtl_segbasename.`*tid*

  The Sun MPI shared memory protocol module uses these files for inter-process communication on the same node. These files consume swap space.

9.3 Procedures for Recovery

9.3.1 Re-creating the RTE 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 `tmrun`, tminfo, or tmps hang). Follow these steps to reboot:

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

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

2. **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 `tmadmin dump` command), kill all the running tasks, 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.

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`.

APPENDIX **A**

Cluster Console Tools

This appendix describes a set of cluster administration tools that are installed with the Sun HPC Software 2.0 release.This toolset, called the Cluster Console Manager, allows you to issue commands to all nodes in a cluster simultaneously through a graphical user interface. The CCM offers three modes of operation:

- `cconsole` – This interface provides access to each node's console port through terminal concentrator links. To use this tool, the cluster nodes must be connected to terminal concentrator ports and those node/port connections must be defined in the `hpc_config` file. See the *Sun HPC 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 Ultra HPC 10000 has no provision for a terminal concentrator), only `ctelnet` and `crlogin` can be used, not `cconsole`.

A.1 Launching Cluster Console Tools

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

```
% tool_name cluster_name
```

where *tool_name* is `cconsole`, `ctelnet`, or `crlogin`, and *cluster_name* is a name given to the cluster. The default cluster name is `hpc_cluster`, which is established automatically when `telnet_setup` is executed. For example,

- Launch `ctelnet` by entering:

```
% ctelnet hpc_cluster
```

- Launch `crlogin` by entering:

```
% crlogin hpc_cluster
```

- Launch `cconsole` by entering:

```
% cconsole hpc_cluster
```

If you want to use `cconsole` to monitor messages generated while rebooting the cluster nodes, you will need to launch it from a machine outside the cluster. If you launch 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 Cluster Console commands take the standard X/Motif command-line arguments.

A.2 Common Window

The Common Window is the primary window used by the system administrator to send input to all the nodes. This window has a menu bar with three menus and a text field for command entry. The Common Window is always displayed when the Cluster Console is launched.

A.2.1 Menu Bar

The menu bar has three menus:

- Hosts
- Options
- Help

Note that in this manual, the Cluster Console term *Hosts* refers to Sun HPC Cluster nodes.

A.2.2 Hosts Menu

The Hosts menu displays a list of the nodes contained in the cluster, plus two other entries, `Select` Hosts and Exit. TABLE A-1 describes these menu choices.

TABLE A-1 Cluster Console Menu Entries

Entry	Function
Host toggle buttons	Selects whether or not the host gets input from the Common Window text field. There is a separate toggle button for each node currently connected to the Cluster Console. ON — Enables input from the Common Window text field to the node. OFF — Disables input from the Cluster Console.
Select Hosts	Displays the Select Hosts dialog window. See “Select Hosts Dialog” for details.
Exit	Quits the Cluster Console program.

A.2.3 Select Hosts Dialog

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

There are three Select Hosts dialog buttons, which are described in TABLE A-2.

TABLE A-2 Select Hosts Dialog Buttons

Entry	Function
Insert	Opens a Term Window and establishes a connection to the specified hosts(s). Adds the host(s) specified in the Hostname text field to the list of accessible hosts. The inserted host name(s) are displayed in the hosts list in the scrolled text window and in the Common Window.
Remove	Deletes the host selected in the Hosts list in the scrolled text window.
Dismiss	Closes the Select Hosts dialog.

▼ 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.**

   The Cluster Console automatically expands the cluster name into its constituent host names and then opens one Term Window for each node. A connection is established for each of the constituent host names. The Cluster Console automatically displays the names of the hosts in the cluster in the scrolled text window and in the hosts list on the Hosts menu in the Common Window.

▼ 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.

A.2.4 Options Menu

The Options menu has one entry, Group Term Window; see TABLE A-3 for a description.

TABLE A-3 Group Term Window Entry

Entry	Function
Group Term Windows	This is a toggle button that groups and ungroups the Common Window and the Term Window. ON — Group; the Term Windows follow the Common Window when the Common Window is moved. OFF — Ungroup: the Term Windows and the Common Window move independently.

A.2.5 Help Menu

The Help Menu has three entries; see TABLE A-4 for a description.

TABLE A-4 Help Menu

Entry	Function
Help	Displays a Help window—the interface to the Sun online help system.
About	Displays the About box, which contains information on the Cluster Console application, such as version number.
Comments	Displays the Comments box, which allows you to enter comments about the software and send them to the development team.

A.3 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.

A.4 Term Windows

The Term Window is just like a normal terminal window. To type on only one host, move the cursor to the Term Window of the desired host and type directly into it.

The Cluster Console Term Windows are like other terminal programs, such as `xterm`, `cmdtool`, and `shelltool`, except that they can also receive input from the Common Window. The Term Windows use VT220 terminal emulation.

The environment variable `TERM` informs your editor of your terminal type. If you are having display problems from `vi` or any other tools, set the environment variable using the appropriate commands for your shell.

The Term Window contains additional functionality, which you can access by positioning the pointer over the Term Window and pressing the right mouse button. This displays the menu described in TABLE A-5.

TABLE A-5 Term Window Menu Entries

Entry	Function
Disable / Enable Scroll bar	Toggles the scroll bar display on and off in the Term Window.
Exit This Window	Closes the current Term Window.

A.5 Using the Cluster Console

To issue commands to multiple nodes simultaneously:

- **Position the cursor in the text field of the Common Window and enter your command.**

  Every keystroke entered in this field is sent to all hosts that are currently selected for input.

  To issue commands to a single node:

- **Position the cursor in the corresponding Term Window and enter your command.**

  Alternatively, you can turn off all hosts in the Hosts menu, except the one you want to access. Then issue your commands from the Common Window.

A.6 Administering Configuration Files

Two configuration files are used by Cluster Console: `clusters` and `serialports`. These files are created automatically by `telnet_setup`, which places them in `/etc`.

A.6.1 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.

A.6.2 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.

Index

B

C

D

E

F

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W