Sun™ HPC Software 2.0 User'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.: 805-1554-10
Revision A, November 1997

Contents

Preface

This manual describes how to use Sun HPC Software to develop, execute, and debug programs.

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

Sun HPC Software provides users with

- High throughput for execution of their serial programs
- Fast execution of their serial or parallel programs
- An optional suite of compilers, libraries, and programming environments that facilitates the development of application programs

1.1 Sun HPC Software 2.0

A Sun HPC System consists of Sun HPC Software running on Sun Ultra HPC Servers (either singly or in clusters), each server (called a node) containing 1 to 64 CPUs. Using the SCI interconnect, Sun Ultra HPC Clusters support up to four nodes.

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 following are valid Ultra HPC servers:

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

- Sun HPC 10000

Sun HPC Software is composed of a Foundation Package and a Parallel Development Environment. The components of the Foundation Package are bundled together, and the Foundation Package is required. The components of the Parallel Development Environment (PDE) are also bundled together, and are optional.

1.1.1 Solaris Operating Environment

Sun HPC Software uses the Solaris 2.5.1 or 2.6 operating environment, extended (for Prism, Sun MPI, Sun HPF, Parallel File System, and related products) by the Sun HPC run-time environment (RTE). All programs that execute under Solaris 2.5.1 or 2.6 execute under Sun HPC Software.

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

1.2 Sun HPC Software Foundation Package

Sun HPC Software includes a suite of resource management software, Sun MPI, PVM, and the Parallel File System.

1.2.1 Resource Management Software

Sun HPC Software provides two resource managers: the Sun HPC run-time environment (RTE), and Platform Computing's Load Sharing Facility (LSF). Both provide

- Load-balancing
- Both remote batch and interactive execution of processes
- Multiple resources for host selection
- Flexible resource requirement strings (RRS)
- Auto-restart of batch jobs if a node fails

The RTE provides access to parallel development software and supplies extensive parallel process management. LSF provides sophisticated serial and batch process management.

1.2.1.1 Sun HPC Run-Time Environment 2.0

The Sun HPC run-time environment (RTE) is the resource management system that enables you to use Sun MPI, Sun HPF, the Parallel File System (PFS), and Prism.

The RTE has three components:

- A command-line user interface, which is described in this manual. The user interface lets users run jobs on, and obtain information about, Sun Ultra HPC nodes.
- A daemon-based environment, which handles security and the allocation of resources on Sun Ultra HPC nodes. For example, when a user submits a program for execution, the run-time environment determines the best node or nodes on which to run the program, and spawns the program on the chosen node or nodes.
- System administration tools, which allow a site to configure, manage, and troubleshoot problems on Sun Ultra HPC nodes. See the *Sun HPC System Administrator's Guide* for information about these tools.

1.2.1.2 LSF 3.0

Load Sharing Facility (LSF) is a resource management system from Platform Computing Corporation that provides load sharing and distributed batch queueing on networks of UNIX systems. 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*, and the *LSF Programmer's Guide*.

For jobs that do not require Sun MPI, Sun HPF, the Parallel File System (PFS), or Prism, LSF provides a sophisticated batch processing system, including

- Start and end-time controls (time windows)
- Exclusive jobs (resource reservation)
- Rerunnable (requeueable) jobs
- Job dependency controls
- Batch file transfers between nonuniform file systems
- Job scripting
- The ability to store job specifications in files for later re-use
- Job migration (on demand or by crossing threshold values)
- Transfer of jobs between queues
- Access control (both user and host)
- Varied access control parameters (such as jobs/instance, jobs/CPU, and jobs/queue)

1.2.2 Sun MPI 3.0

Sun MPI (message-passing interface) is an implementation of the industry-standard MPI library. Sun MPI requires the Sun HPC run-time environment. Sun MPI has the following advantages over the publicly available implementations of MPI:

- Thread-safety, enabling programs to exploit the multithreading and multiprocessing features of Solaris.
- Inclusion of a subset of routines from the MPI I/O standard for coordinated, collective I/O within an MPI program
- On-node communication by shared memory where applicable
- Integration with the run-time environment, taking advantage of load balancing, batch processing, and other resource-management features.
- Prism support, that is, users can develop, run, and debug programs in the Prism programming environment

1.2.3 Parallel File System 1.0

Sun HPC Software includes a Parallel File System (PFS). PFS closely resembles UNIX-style file systems but provides significantly higher file I/O performance because it reads data from and writes data to multiple disks and multiple servers in parallel. Also, PFS supports parallel applications by allowing them to express their high-level I/O needs more clearly than the standard UNIX API allows. PFS is optimized for the complex data-access patterns that are common in parallel scientific applications. See Chapter 6, “Parallel File System.”

1.2.4 PVM 3.3.11

PVM (Parallel Virtual Machine) is a publicly available library of routines for message passing, and is used widely in academic and research computing. PVM is provided at no cost to the user; it is not supported by SunService nor covered under any maintenance agreement. For information about attaching to running PVM processes with the Prism debugger, see Chapter 5, “Debugging.”

1.3 Sun HPC Parallel Development Environment

Sun HPC Software includes a suite of software development tools, Prism, Sun HPF, S3L, and PETSc.

1.3.1 Prism 5.0

Prism is the Sun HPC programming environment, which allows you to develop, execute, debug, and visualize data in programs written in Sun's data-parallel language, Sun HPF, and in message-passing C and Fortran programs. With Prism you can

- Control program execution:
  - Start and stop execution
  - Set breakpoints and traces
  - Print values of variables and expressions
  - Display the call stack
- Visualize data in various formats
- Analyze Sun HPF program performance at several levels. You can display performance data on CPU usage, I/O time, and communication time at the level of:
  - The entire program
  - Individual procedures within the program
  - Individual source lines within procedures
- Control multiple processes, aggregating processes into meaningful groups, called process sets or *psets*.

1.3.2 Sun HPF 1.0

The Sun High Performance Fortran (Sun HPF) compiler is a data parallel program development system offering

- Subset HPF.
- Various Fortran 90 features not in Subset HPF.

- An EXTRINSIC(F77_LOCAL) interface that supports local programming with MPI.
- CM Fortran (CMF) back-compatibility via a switch-selectable CMF mode. This mode allows processing of both CMF layout directives and CMF-specific syntax.
- The HPF utility library.
- Source-level debugging and program development with Prism.

Note – Sun HPF requires the Sun Fortran 77 compiler included in the Sun Performance Workshop Fortran™ suite of tools.

1.3.3 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 includes:

- Vector and Dense Matrix operations
- LU factorization and solve
- FFTs
- Random number generators
- Sort
- Safety mechanism

The chief advantages of S3L are:

- Functions that have an array syntax interface callable from Sun HPF, as well as from C and F77 programs using Sun MPI
- Support for the multiple instance paradigm (whereby the same operation can be concurrently applied to disjoint data sets in a single call)
- Thread safety
- Use of the SunSoft Performance Library for nodal computation
- Detailed programming examples and support documentation provided online

1.3.4 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; it is not supported by SunService

nor 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.

1.3.5 Sun Performance Workshop Fortran v3.0

The Sun Performance Workshop Fortran is a collection of compilers, debuggers, libraries, and tools for both single-processor and SMP systems. Packaged with Sun HPC servers, portions of the Sun Performance Workshop are required by Sun HPC Software.

Note – Sun HPF requires the Sun Fortran 77 compiler, either version 4.0 or 4.2.

1.4 Fundamental RTE Concepts

This section introduces some important concepts that you should understand in order to use the Sun HPC RTE effectively. For information on LSF, see the *LSF User's Guide* and the *LSF Programmer's Guide*.

Note – Remember, the terms *system* and *Sun HPC System* refer to either a single Sun Ultra HPC server or cluster of servers, depending on your local configuration.

1.4.1 Partitions

The RTE organizes the nodes in your Sun HPC system in logical sets, called *partitions.* Each partition can consist of one or more nodes (or, in the HPC 10000, *domains*, which are logically equivalent to nodes). These partitions can have a variety of attributes, which control the way they operate. Your system administrator can configure them to meet the needs of your site. For example,

- In a *shared* partition, multiple programs can execute on the partition's nodes at the same time.
- In a *dedicated* partition, one program at a time has access to all the nodes of a partition. This is particularly useful for fast execution of message-passing programs, which need the coordinated use of multiple nodes.
- A *login* partition is a shared partition that accepts user logins.

Note these further points about RTE partitions:

- RTE partitions are dynamic. Your system administrator can reconfigure a partition so that it has, for example, more nodes or different attributes.
- RTE partitions can be enabled or disabled. You can run programs only in an *enabled partition*. If a partition has its enabled attribute turned off (by the system administrator), no one can run programs in it. This distinction allows the system administrator to swap partitions in and out. For example, there might be different partitions (containing overlapping sets of nodes) available on nights and weekends from those available during the day.

Note – Individual nodes can belong to *at most* one active partition at a time. They can also belong to no partition, in which case they are available to any partition.

- A Sun MPI program can run only on nodes in the same partition; the program cannot span partitions. You can, however, specify running on a node that is not in any partition.
- A user logged in to one partition can execute a program in any active partition (or on another Sun HPC System).

1.4.2 Load-Balancing

The Sun HPC run-time environment *load-balances* programs that execute in shared partitions. When you issue a command to execute a program, the RTE first determines what criteria (if any) you have specified for the node or nodes on which the program is to run. It then determines which nodes meet these criteria within the partition you specified. If more nodes meet the criteria than are required to run your program, the RTE starts the program on the node or nodes that are least loaded. (Specifically, it ranks nodes using their one-minute load averages, with an adjustment that takes into account jobs that have started too recently to affect the load averages.)

This load-balancing mechanism ensures that your program's execution will not be unnecessarily delayed because it happens to run on a heavily loaded node. It also ensures that the overall throughput of the partition is as high as possible—some nodes won't sit idle while other nodes are overburdened.

1.4.3 Tasks and Processes

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

When parallel programs, such as message-passing programs or Sun HPF data-parallel programs, execute, multiple Solaris processes are started, each with its own pid.

The RTE also assigns a *task ID*, or *tid*, to the serial program, the overall message-passing application, and the HPF program. Task IDs always begin with a `t` to distinguish them from pids. Many Sun HPC Software commands take tids as arguments. For example, you can issue a `tmkill` command with a signal number or name and a tid argument to send the specified signal to all processes that make up the task specified by the tid.

Note – A *job* is the entity submitted to a queue. Once spawned, the job becomes a *task*. Job numbers are preceded with the letter *j*, task numbers with the letter *t*. Batch tasks get their *task ID* from the *job ID*, keeping the same number. Only the initial letter of the ID changes.

1.4.4 Parallel File System

PFS combines multiple disks and multiple I/O servers into a single, unified file system that provides scalable, high-performance, parallel I/O. Under PFS, individual files are distributed across multiple disk storage units, each of which is managed by a separate I/O server.

Distributing file systems across multiple disks with independent I/O channels allows HPC processes to read and write PFS files in independent parallel streams, one per server-disk channel. The result is higher aggregate bandwidth as well as opportunities for masking disk access latencies.

1.5 Using the Sun HPC RTE

This section gives a brief overview of how to use the Sun HPC run-time environment (RTE).

1.5.1 Logging In

Use `tmlogin` or `tmtelnet` to log in to a Sun HPC System. The Sun HPC RTE places your login session on the least-loaded node of a login partition.

Logging in is described in more detail in Chapter 2.

Once you are logged in to the system, you can do anything you normally do in a Solaris environment—such as write and compile programs, execute and debug them, send email, and so forth.

All file systems in the system are visible from every node in the system.

1.5.2 Executing Programs

There are two methods of executing a program using the Sun HPC RTE: *interactive* and *batch*.

- Use the `tmrun` command to execute a program in interactive mode. If resources are available, the program executes immediately.
- Use the `tmsub` command to submit a program to a batch queue for execution. The program executes when it reaches the front of the queue, if resources are available.

You can include arguments to both commands to specify the resources you want to use for executing your program—for example, the number of nodes you require, the amount of memory, or the partition in which to execute.

These commands are described in more detail in Chapter 3.

1.5.3 Obtaining Information

You can use RTE commands to obtain various kinds of information:

- Use the `tmps` command to obtain information about currently running tasks and processes.
- Use the `tminfo` command to obtain information about the configuration of the system's partitions, nodes, and batch queues.

These commands are discussed in more detail in Chapter 4.

1.5.4 Operating on PFS Files

PFS includes a collection of utilities that allow you to perform many of the same file operations on PFS files that are commonly performed in UNIX file systems. TABLE 1-1 summarizes these utilities; see their respective man pages for detailed descriptions.

Note – All the PFS utilities that have Solaris counterparts are functionally equivalent to those counterparts, with minor exceptions. The differences in functionality are noted in the table. A few PFS utilities have no Solaris equivalents; these are also identified in the table.

TABLE 1-1 PFS General Utilities

`tmcd`	Set the PFS current working directory.
`tmchgrp`	Change the group ownership of a PFS file or directory. Note: `tmchgrp` does not support the `–f` and `–h` options provided by `chgrp`.
`tmchmod`	Change the access protection bits of a file. Note: `tmchmod` does not support the `–f` option provided by `chmod`.
`tmchown`	Change the owner of a PFS file or directory. Note: `tmchown` does not support the `–f` and `–h` options provided by `chown`.
`tmcmp`	Compare the contents of two PFS files.
`tmcp`	Copy a PFS file or directory within the PFS system. Note: `tmcp` does not support the `–R` option provided by `cp`.
`tmimport/` `tmexport`	Move data between PFS and Solaris file systems. Note: These options do not have Solaris equivalents.
`tmln`	Create a hard link to a PFS file. Note: PFS does not support symbolic links; thus, `tmln` does not support the `–s` option provided by `ln`.
`tmls`	List the contents of PFS directories. Note: `tmls` does not support the `–AbCdfFLmpqsx1` options provided by `ls`.
`tmmkdir`	Create a PFS directory. Note: `tmmkdir` does not support the `–p` option provided by `mkdir`.
`tmmv`	Move a PFS file or directory to another location in the PFS system. Note: `tmmv` does not support the `–f` option provided by `mv`.
`tmpwd`	Print the current working directory.
`tmrm`	Remove a PFS file. Note: `tmrm` does not support the `–R` option provided by `rm`.
`tmrmdir`	Remove a PFS directory. Note: `tmrmdir` does not support the `–R` option provided by `rmdir`.

CHAPTER 2

Logging In and Issuing Commands

This chapter describes how to log in to and issue commands on a Sun HPC System. You must be logged in to the System to issue Sun HPC Software commands.

2.1 Logging In

To log in to a Sun HPC System, use either the `tmlogin` or `tmtelnet` commands from another Solaris system on the network.

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*

You do not log in to a specific node in the system. The run-time environment (RTE) starts your login session on the least-loaded node in a partition that is configured to accept logins (referred to as a *login partition*). All file systems on the system are visible from each node in the system, so the actual node on which you have your session doesn't matter.

To log in via either method, you need to know the name of the Sun HPC System. The name of the Sun HPC System is the same as the name of the node running the RTE master daemons. You can obtain this name from your system administrator.

The two login commands have the same format. The only difference is that `tmlogin` uses the `rlogin` protocol to log in, and `tmtelnet` uses the `telnet` protocol.

To issue the command, type it at your Solaris prompt, specifying the name of the system to which you want to log in:

```
% tmlogin Mars
```

You can omit the system name if you have set the SUNHPC_SYSTEM environment variable to the name of the system. In a C shell, for example,

```
% setenv SUNHPC_SYSTEM Mars
% tmtelnet
```

If there is more than one login partition in the system, you can use the –p option to specify the partition in which you want to have your login session. For example,

```
% tmtelnet –p Deimos
```

or you can set the SUNHPC_PART environment variable to the name of the partition you want to use. If you don't specify a partition in either of these ways, the RTE uses a default partition.

As described above, the RTE then logs you in to a node in the login partition. (Depending on the command you use and the way your site's system is configured, you may have to enter a user name and password first.) You receive the standard Solaris login information, followed by a Solaris prompt:

```
Sun Microsystems Inc.   SunOS 5.51       Generic November 1997
Users: georgek  bowker
Phobos%
```

You are now logged in to one of the nodes in the Mars System. You can issue any Solaris commands and execute programs. See Chapter 3 for more information about executing programs.

2.2 After Logging In

Once you are logged in to a Sun HPC System, you can issue any Solaris or Sun HPC commands, and you can execute any programs that will execute under Solaris 2.5.1 or 2.6. See Chapter 3 for more information about executing programs.

2.2.1 Writing Programs

You can log in to the Sun HPC System to do your program development, or you can do it on any computer that contains the appropriate software.

2.2.2 Compiling and Linking Programs

If you are using a Sun HPC compiler or library, you must compile and link your program on a Sun HPC System on which the required software is loaded.

To compile your Sun HPF, Fortran 77, or C program for use with Prism, use the `-g` option. If you are using a SPARCompiler (from Sun Performance Workshop Fortran, for example), you must also include the `-xs` option. If you want to collect performance data on your Sun HPF program with Prism, compile with the `-tmprofile` option.

See the S3L Guide and Sun MPI Guide for information on linking in S3L or the Sun MPI library, respectively.

2.2.3 Issuing RTE Commands

The Sun HPC RTE provides commands that allow you to execute programs and obtain information. This section provides general information about issuing these commands. The commands are discussed in detail in the next two chapters; in particular, see Section 3.3 for further information about executing programs.

Sun HPC RTE programs are typically in the directory `/opt/SUNWhpc/bin`. If you are unable to execute them, you may need to add this directory to your path; check with your system administrator. The man pages for Sun HPC commands are in `/opt/SUNWhpc/man`. If you cannot display these man pages, you may need to add this directory to your manpath.

You issue Sun HPC RTE commands (which all begin with `tm`) just as you would any Solaris command.

RTE options consist of a dash, typically followed by one or two letters—for example, `-h`, `-np`. You can combine single-letter options that don't take arguments so long as they don't create ambiguity with multiletter options. For example, the command

```
% tmrun -B -T
```

can also be written as

```
% tmrun -BT
```

2.3 Logging Out

To log out of the Sun HPC System and return to your local computer, issue the command

```
% logout
```

CHAPTER 3

Executing Programs

This chapter describes how to issue commands to execute programs on a Sun HPC System. You can execute programs on any node or nodes in any partitions to which you have access. A major difference between the Sun HPC System and a collection of workstations is that the Sun HPC run-time environment (RTE) provides you with extensive and flexible tools for specifying where and how your program should run.

All programs written for Solaris 2.5.1 or 2.6 can run without recompilation on a Sun HPC System.

Note – Remember, the terms *system* and *Sun HPC System* refer to either a single Sun Ultra HPC server or cluster of servers, depending on your local configuration.

3.1 Introduction

3.1.1 Execution Methods

There are two basic methods of execution: *interactive* and *batch.* The system administrator configures each partition to accept one method or both of these methods.

3.1.1.1 Interactive Execution

With interactive access, your program runs right away if the resources you specify are available. This is particularly useful, for example, if you are using Prism to debug the program. The drawback is that, if the resources aren't available, you have to keep resubmitting the program until they become available; your program isn't queued for execution later.

Use the `tmrun` command to execute a program immediately, if resources are available; see Section 3.4.

3.1.1.2 Batch Execution

With batch access, you submit a job to a queue, and it is run when it reaches the front of the queue if the resources you request are available. This is usually the preferable access method for a dedicated partition, where only one job can run at a time, and you are therefore less likely to be able to gain access to it immediately.

Use the `tmsub` command to submit a job to a batch queue; see Section 3.5.

3.1.2 Choosing Where to Execute

The Sun HPC RTE provides you with considerable flexibility in choosing where you want your program to execute. For example, you can specify

- The partition in which you want to execute your program
- The number of processes you want to start, and how you want them to map to nodes
- The characteristics of the node or nodes on which you want to run—for example, the minimum amount of memory required or the maximum acceptable load

See Section 3.7 for further information on specifying where a program is to run.

You can specify default execution criteria via the `TMRUN_FLAGS` environment variable; see Section 3.2. You can also override these criteria via options to the `tmsub` or `tmrun` command.

3.1.3 Authentication Methods

Sun HPC Software includes two optional forms of user authentication that require the execution of user-level commands. The two methods are Kerberos Version 4 authentication and DES authentication. If one of these authentication methods is enforced on your Sun HPC System (see your system administrator for details), use the commands listed in TABLE 3-1 .

TABLE 3-1 User Commands Required by Authentication Methods

Authentication Method	Required Command
DES	While DES authentication is in use, you must issue the `keylogin` command before issuing any commands beginning with `tm`, such as `tmrun` or `tmps`.
Kerberos 4	While Kerberos version 4 authentication is in use, you must issue a `kinit` command before running any command beginning with `tm`, such as `tmrun` or `tmps`.

3.2 Specifying Default Execution Options

Use the `TMRUN_FLAGS` environment variable to specify the default options for `tmsub` or `tmrun`. The system interprets the contents of `TMRUN_FLAGS` as if they follow the `tmrun` or `tmsub` command but precede all options on the command line. You can override a specific default option by including a new value for the option on the `tmrun` or `tmsub` command line. (In the case of the `-R` and `-t` options, the interaction between `TMRUN_FLAGS` and `tmrun` or `tmsub` is slightly more complicated; see Section 3.7.4 and Section 3.7.5.)

The setting of the environment variable can be any number of valid `tmrun` or `tmsub` options. (If you use more than one word, enclose the list in quotation marks.) These options are described in more detail in the remainder of the chapter and are listed in Section 3.5.1, TABLE 3-2.

For example, this C-shell command makes Deimos the default partition to be used for `tmrun` and `tmsub`:

```
% setenv TMRUN_FLAGS "-p Deimos"
```

If you use the Bourne shell (`sh`), the comparable commands would be

```
# TMRUN_FLAGS = "-p Deimos"; export TMRUN_FLAGS
```

Unless you specify otherwise (via the `-p` option), the Sun HPC RTE will execute your programs in this partition.

You can set both `tmrun` and `tmsub` options via `TMRUN_FLAGS`. Note, however, that if you include an option that is unique to one command, issuing the other command will fail, because it won't recognize the option.

You can check the current setting of `TMRUN_FLAGS` by issuing the command

```
% printenv TMRUN_FLAGS
```

In addition, you can use the environment variable `SUNHPC_PART` to specify the default partition in which you want to execute your programs. Once again, you can override this by specifying the appropriate option to `tmrun` or `tmsub` (or by specifying the option in the `TMRUN_FLAGS` environment variable). By default, your programs execute in the partition where you are logged in.

Finally, your system administrator may have established a default partition in configuring the System. You can find this out by issuing the `tminfo` command; see Section 4.2. This default is used only if you are not currently in a partition on that Sun HPC System—for example, if you are logged in to one Sun HPC System and executing a program on another Sun HPC System.

3.3 Executing Programs via `tmrun` and `tmsub`

This section provides general information about executing programs via `tmrun` and `tmsub`. See also Section 2.2.3 for general information on issuing Sun HPC commands.

Execution via `tmrun` and `tmsub` is similar to standard Solaris program execution. For example,

- Your environment is used as if you executed the program from a traditional shell, although you can specify options to manipulate your environment; see Section 3.8.
- Signals are treated as they are in standard Solaris; for multiprocess programs, if one process is killed via a signal, all processes are killed.
- You can run a program in the background:

  ```
  % tmrun a.out &
  ```

There are some differences from standard Solaris execution; they are discussed below.

3.3.1 Moving tmrun Processes to the Background

When you move to the background either a process started with tmrun or a script that issues tmrun commands, you must do one of the following:

- Redirect stdin away from the terminal
- Specify the –n option to tmrun (See Section 3.9.1, “Introducing tmrun I/O”)

If you do not take one of these steps, the tmrun process will contend with your shell for characters typed at the shell, causing unexpected effects.

3.3.2 Shell-Specific Actions

If you want to perform actions that are shell specific, such as executing compound commands, you must first invoke the appropriate shell as part of the tmrun or tmsub command. For example,

```
% tmrun csh –c ‘echo $USER‘
```

or

```
% tmrun csh –c ‘cd /foo ; bar‘
```

3.3.3 Core Files

Core files are produced as they normally are in Solaris. However, if more than one process dumps core in a multiprocess program, the resulting core file may be invalid.

3.3.4 Standard Output and Standard Error

By default, tmrun handles standard output and standard error the way rsh does: The output and error streams are merged and are displayed on your terminal screen. Note that this is slightly different from the standard Solaris behavior when you are not executing remotely; in that case, the stdout and stderr streams are separate. You can obtain this behavior with tmrun via the –D option, and you can specify other methods of handling the standard I/O streams; see Section 3.9.

By default, tmsub puts standard output and standard error into a file; again, for more information, see Section 3.9.

3.3.5 File Descriptors

If your task consists of a large number of processes, you may need to consider the number of file descriptors the task is using and, if necessary, increase the default number available to you.

For merged standard I/O, each process in a task requires two descriptors. For separate `stderr`/`stdout` streams, each process requires three descriptors. (See Section 3.3.4 for more information on merged and separate standard I/O streams.) You also need three file descriptors for interacting with your terminal.

You can find out the default number of file descriptors available in your shell by issuing this command in the C shell:

```
% limit descriptors
```

The corresponding command in the Bourne shell is

```
# ulimit –n
```

The default for most shells is 64. This limits you to about 30 processes for merged standard I/O and about 20 processes for separate standard I/O. If this isn't sufficient, you can increase your limit by issuing a C-shell command, such as

```
% limit descriptors 128
```

or a Bourne-shell command, such as

```
# ulimit –n 128
```

Or you can set it to the maximum value via the C-shell command

```
% unlimit descriptors
```

or the Bourne-shell command

```
# ulimit –n ‘ulimit –Hn‘
```

The file descriptor maximum in Solaris 2.*x* is 1024.

3.3.6 SMP Characteristics of Sun HPC Systems

Since your Sun HPC System consists of symmetric multiprocessors (SMPs), by default Sun HPC Software takes into consideration the number of CPUs per node. In general, `tmrun` will assign more processes to larger SMPs. For information about how the RTE allocates processes to CPUs, see Section 3.7.3.1, "Specifying How –np Relates to the Number of Available CPUs," and Section 3.7.6, "Running on SMPs."

3.4 Executing Programs Interactively

Use the `tmrun` command to execute a program in a partition that has been configured for interactive access. If the resources you request are available, the program runs. If they aren't available, you receive an error message and should try again later.

Recall that, in addition to specifying execution criteria via `tmrun`, you can also set up default execution criteria via the `TMRUN_FLAGS` environment variable; see Section 3.2. If you don't have default criteria and don't request specific resources, the system executes the program on the least loaded-node or nodes in the partition it chooses; once again, see Section 3.2 to learn how the system chooses the partition. If you do request specific resources, the system executes the program on the least-loaded node or nodes that meet the criteria you specify.

The basic format of the `tmrun` command is

```
% tmrun [options] [-] executable [args ... ]
```

Note – The dash between the `tmrun` options and the name of the executable program is optional. You must include it when the name of the program conflicts with that of a `tmrun` option. TABLE 3-2 lists all `tmrun` and `tmsub` options.

3.5 Submitting Batch Jobs

Use the `tmsub` command to submit a job to a batch queue for execution in a partition that has been configured for batch access. Use the `-q` option to specify the queue. For example,

```
% tmsub -q Demeter a.out
```

submits the program `a.out` to the queue Demeter.

When you submit a program to a queue, it is executed when it reaches the front of the queue if the requested resources are available. If they aren't, the system will act based on the value of the queue policy attribute. The two possible values of this attribute are

- *block* – retries executing the job periodically.
- *requeue* – tries to run another job before retrying the job.

You can remove a job from a queue via the `tmkill` command; see Section 3.18. The queue must be enabled for you to be able to submit to it; it must be running for the job to run once submitted.

Once a queue accepts a job, it returns a job ID that identifies that job. You can use this job ID to find out about the job's status via the `tmps` command.

To obtain information about queues, issue the command

```
% tminfo -Q
```

See Section 4.2 for more information about `tminfo`.

TABLE 3-2 lists all `tmsub` options. Section 3.6 through Section 3.15 describe `tmsub` options in detail. You can include `tmsub` options in your `TMRUN_FLAGS` environment variable; see Section 3.2.

3.5.1 Choosing the Queue: More Detail

As discussed above, you can use `tmsub -q` to specify the queue to which to submit your job. Queue names are unique within a partition. However, multiple queues in a Sun HPC System can have the same name if they are not in the same partition.

If you specify `tmsub -p` *a_partition* `-q` *a_queue*, then the System will use *a_queue* in *a_partition*. If there's no such queue in the partition, the System will generate an error.

Note – If the `TMRUN_FLAGS` environment variable includes a `-p` or `-q` flag, that's equivalent to having the flag(s) on the command line.

If you specify `tmsub -p` *a_partition*, then the system will use the default queue in *a_partition*. If there's only one queue in *a_partition*, that's the default queue.

If you specify `tmsub -q` *a_queue*, Sun HPC Software uses the following criteria, in this order of precedence:

1. If there's only one queue in the entire system called *a_queue*, the system will use that queue.
2. If the default batch partition has a queue called *a_queue*, the system will use that queue.
3. If the `SUNHPC_PART` environment variable is set to the name of a partition that has a queue called *a_queue*, the system will use that queue.
4. If the node on which `tmsub` was invoked is part of a partition that has a queue called *a_queue*, the system will use that queue.

If you specify `tmsub` without naming a partition or queue, Sun HPC Software uses the following criteria, in this order of precedence:

1. If there's only one queue in the entire system, the System will use that queue.
2. If there's a default batch partition with a default queue, the System will use that queue.

Note – If a partition has only one queue, that queue is treated as the default queue for that partition.

---

3. If the `SUNHPC_PART` environment variable is set to the name of a partition with a default queue, the system will use that queue.
4. If the node on which `tmsub` was invoked is part of a partition that has a default queue, the system will use that queue.

To determine whether a partition has been assigned a default queue, use the `tminfo` command; see Section 4.2

3.6 `tmrun` and `tmsub` Options

The `tmrun` and `tmsub` commands share most options in common. Options that are unique to one or the other are identified as such when we discuss them. See Section 2.2.3 for general information about options for Sun HPC commands.

Recall that the setting of the `TMRUN_FLAGS` environment variable may provide default settings for `tmrun` and `tmsub` options; see Section 3.2. Specifying the option on the command line generally overrides the setting of that option in the environment variable..

TABLE 3-2 Options for `tmrun` and `tmsub`.

Option	Meaning
`–A` *aout*	Execute `aout`, and use a different argument as the `argv[0]` argument to the program. See Section 3.14.
`–B`	Send `stderr` / `stdout` output streams to files (Specified for `tmrun` only. This is the default for `tmsub`). See Section 3.9.
`–C` *path*	Use the specified directory as the current working directory for the task. See Section 3.11.

TABLE 3-2 Options for `tmrun` and `tmsub`.

Option	Meaning
`-D`	Provide separate `stdout` and `stderr` streams (`tmrun` only). See Section 3.9.
`-Ea` *envar*	Add the specified entry to the inherited environment. See Section 3.8.
`-Ec`	Remove the inherited environment. See Section 3.8.
`-Ed` *envar*	Delete the specified entry from the inherited environment. See Section 3.8.
`-Em`	Set up a minimal environment. See Section 3.8.
`-G` *group*	Execute with the specified group ID or group name. See Section 3.12.
`-h`	Display help. See Section 3.13.
`-i`	Standard input to tmrun is sent only to rank 0, and not to all other ranks (tmrun only).
`-I` *iofds*	Use the specified I/O file descriptor string to control I/O stream handling. See Section 3.9.
`-M` *user*	Send mail upon job completion to the specified user, rather than the user who ran `tmsub` (`tmsub` only). See Section 3.15
`-n`	Read `stdin` from `/dev/null`. See Section 3.9.
`-N`	Do not open any standard I/O connections (`tmrun` only). See Section 3.9.
`-np` *number*	Request the specified number of processes. See Section 3.7.3.
`-Nr`	Don't restart the task if a node fails (`tmsub` only). See Section 3.10.
`-Ns`	Disable spawning of multiple processes from a task on SMPs; see Section 3.7.6.
`-o`	Use line buffering on standard output, prefixing each line with the rank of the process that wrote it (tmrun only)
`-p` *partition*	Run in the specified partition. See Section 3.7.1.
`-P` *priority*	Run with the specified priority (`tmsub` only). See Section 3.9.2.
`-q` *queue*	Submit to the specified queue (`tmsub` only). See Section 3.5.
`-R "`*specifier*`"`	Specify conditions for choosing nodes. See Section 3.7.4.
`-S`	Settle for the available number of nodes (used with –np). See Section 3.7.3.
`-s` *system*	Run on the specified system. See Section 3.7.2.

TABLE 3-2 Options for `tmrun` and `tmsub`.

Option	Meaning
`-t` *tid*	Run on the same node(s) as the task with task ID `tid` (`tmrun` only). See Section 3.7.5.
`-T`	Show the tid, system name, and number of processes after executing (`tmrun` only). See Section 3.13.
`-U` *user*	Execute with the specified user ID or user name. See Section 3.12.
`-V`	Display version information. See Section 3.13.
`-W`	Wrap the requested processes on the available CPUs (used with `-np`). See Section 3.7.3.
`-Yr`	Restart the task if a node fails (`tmsub` only). See Section 3.10.
`-Ys`	Allow spawning on SMPs. See Section 3.7.6.
`-Z` *rank*	Run processes, by groups of size *rank*, together on the same node. (incompatible with –S and –W) See Section 3.7.6.1.

3.7 Specifying Where a Program Is to Run

The `tmrun` and `tmsub` commands provide you with considerable flexibility in specifying where you want your job to run. Section 3.7.1 describes how to choose the partition in which a program is to run; Section 3.7.2 describes how to choose the System on which you want your program to run. Section 3.7.3 describes how to specify how many processes are to be started and how they should be mapped to nodes. See Section 3.7.4 for a discussion of resource requirement specifiers, which provide a syntax for specifying complex requirements that can't be encapsulated in these simple command-line options.

In cases where your requirements can be met by more than one node, the system chooses the least-loaded node, unless you have specified other sorting criteria.

3.7.1 Specifying the Partition

When executing programs interactively, use `tmrun -p` to specify the partition in which you want your program to run. (To find out the names of enabled partitions in your System, use the `tminfo` command; see Section 4.2.) For example,

```
% tmrun -p Deimos a.out
```

specifies that `a.out` is to be run in the partition Deimos.

When submitting batch jobs, use `tmsub -p` to specify the partition associated with the queue to which you want to submit your job. You can omit `-q` if this partition has a default queue or if it has only a single queue enabled.

3.7.2 Specifying the System

For interactive program execution, use `tmrun -s` *systemname* to specify the Sun HPC System on which you want your program to run (for RTE purposes, the system name of a Sun HPC System is the same as the name of the node running the RTE master daemons). By default, it runs on the System to which you are logged in.

When submitting batch jobs, use `tmsub -s` *systemname* to specify the system that contains the queue where you want to submit your job. By default, the RTE looks on the System where you are logged in for this queue.

3.7.3 Specifying the Number of Processes

Use the `-np` option to specify the number of processes you want to start; the default is 1. This option is typically used with a Sun MPI program or a Sun HPF program. It determines how many copies of the executable will run.

For example,

```
% tmrun -p Deimos -np 4 myprog
```

specifies that you want four copies of `myprog` to start on the nodes of the partition Deimos.

You can also specify 0 as the setting of `-np`. This specifies one process per CPU on each available CPU. Thus, if the partition Deimos has six available CPUs, the command

```
% tmrun -p Deimos -np 0 myprog
```

starts six copies of `myprog`.

Note – If you specify the argument `-Ns`, `tmrun` starts one process per node.

3.7.3.1 Specifying How –np Relates to the Number of Available CPUs

The Sun HPC run-time environment (RTE) attempts to start one process per CPU when you request multiple processes via the `–np` option. If it cannot do this, the command fails, unless you use the `–W` or `–S` option to specify the behavior you want in this situation.

Use the `–W` option if you want the processes to *wrap*, so that more than one process will run on a CPU. For example, if you specify

```
% tmrun –p Deimos –np 6 –W myprog
```

and there are only four CPUs available in the partition Deimos, the RTE starts six processes on the four CPUs. All the processes are started in a load-balanced fashion, so you can't determine ahead of time where they will run.

Use the `–S` option to specify that you will *settle* for the available number of CPUs. The RTE will start one process on each available CPU. Thus, if you issue the same command as above, but substitute `–S` for `–W`:

```
% tmrun –p Deimos –np 6 –S myprog
```

and four CPUs are available on Deimos, then four copies of `myprog` will start, one per CPU.

Note – If a `tmrun` resource request does not provide you with enough nodes from inside the partition, (controlled by the `–np`, `–S`, and `–W` options), then the search expands to look for enabled nodes not contained within any partition. If you specify `–np 0`, `–S`, or `–W`, the search will never go outside the partition.

3.7.4 Expressing More Complex Requirements

Use the `–R` option to express complex node requirements that aren't accessible via the general options discussed above.

The argument to the `–R` option is a *resource requirement specifier* (RRS). The RRS is enclosed in quotation marks and provides the settings for any number of attributes that you want to use to control the selection of nodes. You combine these attribute settings using the logical & (AND) and | (OR) operators. The system parses the attribute settings in the order they are listed in the RRS, along with other options you specify, then merges the results with the results of an internal RRS that provides the load balancing (with one exception discussed below). The result is an ordered list of CPUs that meet your criteria. If you are starting a single process, the RTE

starts the process on the CPU that's first in the list. If you are starting *n* processes, the run-time environment starts them on the first *n* CPUs, wrapping if necessary, or settling for the number of CPUs in the list.

Note – The RRS specifies node resources but generates a list of CPUs unless `–Ns` is specified. Also, your RRS is interpreted only with regard to the nodes of a single partition—either the partition you specify on the command line or a default partition; see Section 4.2.

3.7.4.1 Attributes

TABLE 3-3 lists predefined attributes you can include in an RRS. Your system administrator may also have defined attributes specific to your Sun HPC System. You can find out about these attributes via the `tminfo` command; see Section 3.2.

There are two types of attributes, value and boolean:

- *Value* attributes can have either a literal value or a numeric value.
  - Attributes with a literal value take a name as a setting. Use an equal sign and the name after the attribute to show the setting. For example,

```
% tmrun –R "name = telemachus" a.out
```

  - Attributes with a numeric value take as a setting either an operator and a value or, for << and >>, an operator without a value. For example,

```
% tmrun -R "os_arch_kernal > sun4c" a.out
```

    specifies that you want to use any kernal architecture lexically greater than sun4c. Thus, sun4m and sun4u are acceptable, but not sun3c.

```
% tmrun -R "mem_total>>" a.out
```

    specifies that you prefer nodes with more physical memory.

Note – You can specify a combination of attributes — for example, –R "name < telemachus". Also, literal and numeric attributes can be used together.

- *Boolean* attributes are either true or false. If you want the attribute to be true, simply list the attribute in the RRS. For example, if your system administrator has defined an attribute called `ionode`, you can request a node with that attribute:

```
% tmrun –R "ionode" a.out
```

  If you want the attribute to be false (that is, you do not want a resource with that attribute), precede the attribute's name with `!`. (Precede this with a backslash in the C shell; the backslash is an escape character to prevent the shell from interpreting the exclamation point as a "history" escape.) For example,

```
% tmrun –R "\!ionode" a.out
```

- The accepted operations are

Operator	Meaning
<	Attribute must be less than the specified value.
<=	Attribute must be less than or equal to the specified value.
=	Attribute must be equal to the specified value.
>=	Attribute must be greater than or equal to the specified value.
>	Attribute must be greater than the specified value.
!=	Attribute must not be equal to the specified value. (Precede with a backslash in the C shell.)
<<	Attribute must be as low as possible.
>>	Attribute must be as high as possible.

For example,

```
% tmrun –R "mem_free > 256" a.out
```

specifies that the node must have over 256 megabytes of available RAM.

```
% tmrun –R "swap_free >>" a.out
```

specifies that the node picked must have the highest available swap space.

Note – If you use the << or >> operator, the run-time environment does not provide load balancing. In the above example, the run-time environment would choose the node with the most free swap space, no matter what its load was. If you use << or >> more than once, only the last use has any effect – it overrides the previous uses. For example,

```
% tmrun –R "load1<2 & mem_free>>" a.out
```

first selects the nodes that have a one-minute load average less than 2; of these nodes, it selects the one with the most free memory.

You can also use arithmetic expressions involving numeric attributes anywhere you can use a numeric attribute. For example,

```
% tmrun –R "load1 / load5 < 2" a.out
```

specifies that the ratio between the one-minute load average and the five-minute load average must be less than 2. In other words, the load average on the node must not be growing too fast. You can use standard arithmetic operators, as well as the C `?:` conditional operator.

Note – Some shell programs interpret characters used in an RRS (for example, the characters: >, <, and !). By using the proper syntax, you can protect your RRS from undesired interpretation by your shell. For example, if you use `csh`, write "`-R \!private`" instead of "`-R !private`".

The operators have the following precedence, from strongest to weakest:

```
unary –
*, /
+, binary –
=, !=, >=, <=, >, <, <<, >>
!
&, |
?:
```

TABLE 3-3 Standard node attributes that can be included in an RRS.

Attribute	Meaning
`cpu_idle`	Percent of time that the CPU is idle.
`cpu_iowait`	Percent of time that the CPU spends waiting for I/O.
`cpu_kernel`	Percent of time that the CPU spends in the kernel.
`cpu_scale`	Performance rating of the CPU.
`cpu_swap`	Percent of time that the CPU spends waiting for swap.
`cpu_type`	CPU architecture.
`cpu_user`	Percent of time that the CPU spends running user's program.
`load1`	Node's load average for the past minute.
`load5`	Node's load average for the past five minutes.
`load15`	Node's load average for the past 15 minutes.
`manufacturer`	Hardware manufacturer.
`mem_free`	Nodes's available RAM, in Mbytes.
`mem_total`	Node's total physical memory, in Mbytes.
`name`	Node's hostname.
`os_max_proc`	Maximum number of processes allowed on the node, including system daemons.
`os_arch_kernel`	Node's kernel architecture.
`os_name`	Operating system's name.

TABLE 3-3 Standard node attributes that can be included in an RRS.

Attribute	Meaning
os_release	Operating system's release number.
os_release_maj	The major number of the operating system's release number.
os_release_min	The minor number of the operating system's release number.
os_version	Operating system's version.
serial_number	Node's serial number.
swap_free	Node's available swap space, in Mbytes.
swap_total	Node's total swap space, in Mbytes.

3.7.4.2 Examples

Here are some examples of the use of the –R option.

This example specifies that the program must run on a node in the partition Deimos with 512 Mbytes of memory:

```
% tmrun –R –p Deimos "mem_total=512" a.out
```

This example specifies that you want to run on any of the three nodes listed:

```
% tmrun –R "name=node1 | name=node2 | name=node3" a.out
```

This example chooses nodes with over 300 Mbytes of free swap space. Of these nodes, it then chooses the one with the most total physical memory:

```
% tmrun –R "swap_free > 300 | mem_total>>" a.out
```

The following example assumes that your system administrator has defined an attribute called framebuffer, which is true if a node has a frame buffer attached to it. You could then request such a node via the command

```
% tmrun –R "framebuffer" a.out
```

3.7.4.3 The –R option and TMRUN_FLAGS

If you have included a –R option in your TMRUN_FLAGS environment variable (see Section 3.2) and also have a –R option on your tmrun or tmsub command line, the two RRSs are combined; the command-line –R option does not override the one specified via the environment variable. The same behavior occurs if you specify multiple –R options on the same command line. For example, if you have

```
% setenv TMRUN_FLAGS '-R "load1 < 1"'
```

and issue the command

```
% tmrun –R "load5 < 1" –R "load15 < 1" a.out
```

this is the same as issuing the command

```
% tmrun –R "(load1<1) & (load5<1) & (load15<1)" a.out
```

If you use a –t option in TMRUN_FLAGS and a –R option in tmrun/tmsub, the –R option overrides the –t option; see Section 3.7.5 for information on –t.

3.7.5 Running on the Same Node(s) as a Currently Running Task (tmrun Only)

Use the –t option to tmrun to specify that the program you want to execute should run on the same node or nodes as the task with the task ID (tid) you specify. (Use the tmps command to obtain a task's tid.) For example, to run a.out on the same node(s) as t85, issue the command

```
% tmrun –t t85 a.out
```

If –t follows –np or –R on the command line, it overrides these options. If –np (along with –W or –S) follows –t, –t determines which nodes to run on, and the other options determine the number of processes to map onto these nodes.

3.7.6 Running on SMPs

By default, tmrun and tmsub spawn multiple processes on SMPs (symmetric multiprocessors). For example, if you have a two-node partition in which one node has two CPUs and the other has four CPUs, then the command

```
% tmrun –np 6 a.out
```

runs six copies of a.out, two on the two-headed node and four on the four-headed node. The –t option (see Section 3.7.5) and –R option (see Section 3.7.4) override this behavior. In addition, you can use the –Ns option to disable spawning of processes on individual heads of an SMP node. By default, tmrun spawns multiple processes on an SMP. If you specify –Ns, only one process is spawned on each SMP. For example, in the two-node partition described above,

```
% tmrun –np 0 –Ns a.out
```

would cause two processes to run, one on each node.

Use the –Ys option to force spawning on SMPs when used with –R. –Ys does not override –t.

3.7.6.1 Rank-to-Node Mapping

To specify that a subgroup of a task's processes run grouped on the same SMP node, one process per CPU, specify the Z option with the group size. For example,

```
% tmrun –Z 3 –np 8 a.out
```

groups the task's processes by threes.

Note – The –Z option is incompatible with the –S and –W options.

3.8 Specifying the Execution Environment

Use the options discussed in this section to determine the environment to be used for the execution of your program.

Use the –Ec option to "clobber" the inherited environment—that is, to erase all existing settings of environment variables.

Note – The –Ec option should be used with care, since some settings (for example, HOME, USER, SHELL) are required to execute most programs properly from your computer.

Use the –Ea option to add the environment variable you specify to your inherited environment. For example,

```
% tmrun -Ea "LD_BIND_NOW = 1" -np 2 a.out
```

Use the –Ed option to delete the environment variable you specify from your inherited environment.

Use the –Em option to set up a minimal environment. This is comparable to –Ec, but preserves the settings of these environment variables:

- DISPLAY
- USER
- HOME
- TERM
- PATH

Note – The -EC and -EM options remove user-specified environment variables, such as those set for MPI.

3.9 Specifying What to Do with Standard Input, Output, and Error

3.9.1 Introducing `tmrun` I/O

By default, `tmrun` arranges for all standard output and standard error (Solaris file descriptors 1 and 2) from all the processes in a task to be sent to `tmrun`'s standard output (Solaris file descriptor 1). Typically, this is the user's terminal. Likewise, `tmrun`'s standard input (file descriptor 0) is sent to the standard input of all the processes. You can redirect `tmrun`'s standard input, output, and error using the standard shell syntax. For example,

```
% tmrun -np 4 echo hello > hellos
```

You can also change what happens to the standard input, output, and error of each process in the task. For example,

```
% tmrun echo hello > message
```

sends `hello` across the network from the echo process to the `tmrun` process, which writes it to a file called `message`.

The `tmrun` command's own options allow you to control I/O in several other ways. For example, rather than making remote processes communicate with `tmrun` (when it may not be necessary), you can make each process write to or read from a file on the node on which it is running. For example, you can make each process write `hello` directly to a file on its own node called `message`:

```
% tmrun -I "1w=message" echo hello
```

Using `tmrun` options, you can also

- Make the standard error from each process go to the standard error of `tmrun`, instead of its standard output. To divide standard error from standard output, use the `tmrun -D` option. For example,

  ```
  % tmrun -D a.out
  ```

  sends standard output from `a.out` to the standard output of `tmrun`, and sends the standard error of `a.out` to the standard error of `tmrun`.

- Make output from each process go to a file whose name includes the rank and tid of the process. This corresponds to the `tmrun -B` option. Using the `-B` option specifies that the I/O streams are to operate as in a batch environment. Specifically,
  - There is no standard input stream.

- The standard error and standard output streams for a task are merged and written to files named `out.`*tid*`.`*rank*, where *tid* is the task ID of the task and *rank* is the rank of this process within the task. The files are located in the task's working directory.

- Shut off all standard I/O to all the processes. By using the `tmrun –N` option, you specify that there are to be no `stdin`, `stdout`, and `stderr` connections. Use the `–N` option for situations in which standard I/O is not necessary; you can reduce the overhead incurred by establishing standard I/O connections for each remote process and then closing those connections as each process ends.
- Use the `–n` option to cause `stdin` to be read from `/dev/null`. This is important if you are running `tmrun` in the background, either directly or through a script; without `–n`, `tmrun` will block in this situation, even if no reads are posted by the remote task. When `–n` is specified, the user process encounters an `EOF` if it attempts to read from `stdin`. This is comparable to the behavior of the `–n` option to `rsh`.

Note – The set of `tmrun` (and `tmsub`) options that control `stdio` handling cannot be combined. These options override one another. If more than one is given on a command line, the last one overrides all of the rest. The affected options are: `-D`, `-N`, `-B`, `-n`, `-i`, `-o`, and `-I`.

3.9.1.1 Creating a Custom Configuration

Use the `–I` option to specify a custom configuration for standard input, output, and error. The `–I` option takes as an argument a comma-separated series of *file descriptor strings*. These strings specify what is to happen with each of these streams.

In Solaris, each process has a numbered set of *file descriptors* associated with it. The standard I/O streams have these file descriptors assigned:

- `0` – standard input (`stdio`)
- `1` – standard output (`stdout`)
- `2` – standard error (`stderr`)

You can include a string for each of these file descriptors; if you omit one, that file descriptor won't be connected to any device.

The file descriptor string assigns one or more of the following attributes to a file descriptor:

- `r` – File descriptor is to be read from.
- `w` – File descriptor is to be written to.
- `p` – File descriptor is to be attached to a pseudo-terminal (pty).

You must specify either `r` or `w` for each file descriptor—that is, whether the file descriptor is to be written to or read from.

Thus, the string

```
2w
```

means that the standard error is to be written. And

```
0rp
```

means that the standard input is to be read from the pseudo-terminal.

If you use the `p` (pty) attribute, you must have one `rp` and one `wp` in the complete series of file descriptor strings. In other words, you must specify both reading from and writing to the pty. No other attributes can be associated with `rp` and `wp`.

In addition, these attributes can only be used in conjunction with `w`:

- `l` – Line-buffered output.
- `t` – Tag the line-buffered output with process rank information.
- `a` – Stream is to be appended to the specified file.

Note – NFS does not support append operations.

Use the `l` attribute in combination with the `w` attribute to line-buffer the output of multiple processes. This takes care of the situation in which output from one process arrives in the middle of output from another process. For example,

```
% tmrun –np 2 echo "Hello"
HelHello
lo
```

With the `l` attribute, you ensure that processes don't step on each other's output in this way:

```
% tmrun –np 2 –I "0r, 1wl" echo "Hello"
Hello
Hello
```

Use the `t` attribute to force line-buffering and, in addition, prefix each line with the rank of the process producing the output. For example,

```
% tmrun –np 2 –I "0r, 1wt" echo "Hello"
r0:Hello
r1:Hello
```

Use the `b` attribute in combination with the `r` option in multiprocess tasks to specify that input is to go only to the first process, rather than to all processes (the default). This attribute provides compatibility with Release 1 of Sun HPC, in which input to the first process only was the default.

Example – In this example, the standard input is read from the pty, the standard output is written to the pty, and the standard error is sent to the file named `errors`:

```
% tmrun –I "0rp,1wp,2w=errors" a.out
```

(The quotation marks are optional.)

This attribute can be used only with `r`:

- `b` – Input only goes to the first process, rather than to all processes.

This attribute can be used only with `rp`:

- `m` – Keypresses are to be echoed multiple times when multiple processes are running. (The default is to display them only once.)

You can direct one file descriptor's output to the same location as that specified by another file descriptor by using the syntax

fdattr=@other_fd

For example,

```
2w=@1
```

means that the standard error is to be sent wherever the standard output is going. You cannot do this for a file descriptor string that uses the `p` attribute.

Finally, you can tie a file descriptor's output to a file by using the syntax

fdattr=filename

For example,

```
1w=output
```

says that the standard output is to be written to the file `output`. Once again, however, you cannot use this feature for a file descriptor defined with the `p` attribute.

If you use the `w` attribute without specifying a file, the file descriptor's output is written to the corresponding output stream of the parent process; the parent process is typically a shell, so the output is typically written to the user's terminal.

For multiprocess tasks, each process creates its own file; the file is opened on the node on which the process runs.

Note – If multiple processes open the same file over NFS, the processes will overwrite each other's output.

In specifying the individual file names for processes, you can use these symbols, which are replaced by the indicated values:

- `&T` – The task ID of the task
- `&R` – The rank of the process within the task

In the next example, there is no standard input (it comes from `/dev/null`), and the standard output and standard error are written to the file `out.`*task*`.`*rank*:

```
% tmrun –I "0r=/dev/null,1w=out.&T.&R,2w=@1" a.out
```

This is the behavior of the `–B` option. See Section 3.9.1.

Here is the default behavior (merged standard error and standard output):

```
% tmrun –I "0rp,1wp,2w=@1" a.out
```

The `–D` option (providing separate standard output and standard error streams) is equivalent to:

```
% tmrun –I "0rp,1wp,2w" a.out
```

You can force each line of output (`tmrun` only) to be prepended with the rank of the process writing it, using the `–o` option (`See Section 3.9.1`), which is equivalent to:

```
% tmrun -I "0rp,1wt,2w=@1" a.out
```

If you redirect output to a shared file, you must use standard shell redirection rather than the equivalent `–I` formulation (`–I "lwt=outfile"`). This restriction also applies to the linebuffer formulation (`–I "lwl=outfile"`).

Note – Specifying `-o` (forcing processes to prepend rank on output lines), or the equivalent `-I` syntax (such as `-I1wt`) will not work if redirection is also specified with `-I` (such as with `-I1w=outfile`). Use the standard shell redirection operator instead.

Shortcuts for common `tmrun –I` commands are listed in TABLE 3-4

TABLE 3-4 `tmrun` Shortcut Summary

Command	Description
`tmrun –i`	Standard input to `tmrun` is sent only to rank 0, and not to all other ranks (`tmrun` only). Equivalent to `tmrun –I "0rpb,1wp,2w=@1" a.out`
`tmrun –B`	Standard output and standard error are written to the file `out.`*task*`.`*rank*. Equivalent to `tmrun –I "0r=/dev/null,1w=out.&T.&R,2w=@1" a.out`
`tmrun –o`	Use line buffering on standard output, prefixing each line with the rank of the process that wrote it (`tmrun` only). Equivalent to `tmrun –I "0rp,1wt,2w=@1" a.out`

These shortcuts are not direct substitutions. The RTE uses ptys correctly, whether the –I option is present or absent. Also, the RTE merges standard error with standard output when it is appropriate. If either stderr or stdout is redirected (and not both), then ptys are not used, and stderr and stdout are separated. If both stderr and stdout are redirected, then again ptys are not used, but stderr and stdout are combined.

Note – Use the -i option to tmrun with caution, since the -i option provides only one stdin connection (to rank 0). If that connection is closed, keyboard signals are no longer forwarded to those remote processes. To signal the task, you must go to another window and issue the tmkill command. For example, if you issue the command tmrun –np 2 –i cat then type the Ctrl-d character (which causes cat to close its stdin and exit), rank 0 will exit. However, rank 1 is still running, and can no longer be signaled from the keyboard.

3.9.2 tmsub

By default, tmsub's behavior is like that of the tmrun –B option. Namely,

- There is no standard input stream.
- The standard error and standard output streams for a task are merged and written to files named out.*tid*.*rank*, where *tid* is the task ID of the task and *rank* is the rank of this process within the task. These files are located in the task's working directory.

You can use the –I option to specify a different behavior for standard output and standard error, with the exception that you cannot use the p attribute to attach a file descriptor to a pty. See Section 3.9.1 for more information on –I.

3.9.2.1 Setting Queue Priorities

Two factors control job priorities:

- The queue configuration, set for each queue by the system administrator
- The tmsub –P option, set for each job by the user

The system administrator can set a default priority for each queue, which is used by the system for all jobs submitted without the tmsub –P option (the –P option takes an integer argument, the magnitude of the integer corresponding to magnitude of the job's priority). The system administrator can also set a minimum and a maximum allowed priority for each queue. If you submit a job with a priority lower than the minimum, the job's priority will be adjusted upwards to the minimum. Likewise, a job submitted with a priority higher than the maximum has its priority adjusted down to match the maximum.

When the system schedules jobs to run, it will pick the highest-priority job across all the queues in a partition. If multiple jobs have the same priority, it will pick the one that was submitted first (regardless of which queue the job is in).

3.10 Restarting a Task if a Node Goes Down (`tmsub` Only)

Use the `-Yr` option to `tmsub` to specify that you want your task to be restarted if a node goes down while the task is being executed. If the task is parallel, the entire task is aborted if it has any processes on a node that fails.

After the node goes down, the RTE immediately tries to restart the task; if it fails, it continues periodically to attempt the restart. It chooses the node(s) on which the task is to run in the normal way; therefore, the task won't necessarily run on the same node(s) on which it was running before the failure, and you don't necessarily have to wait for the node to come back up for your task to run.

By default, tasks do not restart. If you have specified that a job is to restart (for example, via the setting of the `TMRUN_FLAGS` environment variable), you can override this setting by issuing the `-Nr` option.

3.11 Changing the Working Directory

Use the `-C` option to specify the path of an alternative working directory to be used by the program; if you don't specify `-C`, the default is the current working directory. For example,

```
% tmrun -C /home/collins/bin a.out
```

changes the working directory for `a.out` to `/home/collins/bin`.

3.12 Executing with a Different User or Group Name

Use the `-U` option to execute with the specified user ID or user name. Use the `-G` option to execute with the specified group ID or group name. For example,

```
% tmrun –U traveler a.out
```

executes `a.out` as the user `traveler`.

You must have the appropriate level of permissions to use these options (for example, you must belong to the group you specify, or be the superuser).

3.13 Getting Information

Use the `–h` option to display a list of `tmrun` or `tmsub` options and their meanings.

Use the `–V` option to display the command's version number.

If you specify either `–h` or `–V`, it must be the only option on the command line.

Use the `–T` option to display the program's tid, along with the name of the System and the number of processes, after executing `tmrun`. This option is not available for `tmsub`.

3.14 Specifying a Different Argument Vector

The `tmrun` and `tmsub` commands by default pass the vector of a program's command-line arguments to the program in the standard way. For example, if you issue the command

```
% tmrun a.out arg1 arg2
```

`tmrun` passes an array in which the name of the program, `a.out`, is the first element (`argv[0]`), and `arg1` and `arg2` are the second and third elements.

In system-level programming it is sometimes useful to specify an `argv[0]` that is not the name of the program. You can use the `–A` option to do this. The argument to `–A` is the name of the program to execute. You can then follow this with the `argv[0]` argument, and any other arguments that you want to pass to the program. For example, if you want to pass `newarg` as the `argv[0]` to the program `a.out`, along with `arg1` and `arg2`, you could issue the command

```
% tmrun –A newarg a.out arg1 arg2
```

3.15 Sending Mail About Job Status (`tmsub` Only)

By default, `tmsub` sends mail to the user who submitted a job when the job is done. You can specify that mail is to be sent to a different valid email address by including the address as the argument to the `-M` option.

3.16 Exit Status

The exit status of `tmrun` specifies the number of processes that exited with nonzero exit status.

3.17 Omitting `tmrun` or `tmsub`

You can execute a serial program without using `tmrun` or `tmsub`. For example, you could simply type

```
% a.out
```

In that case, the program executes locally, on the node where you happen to be logged in. Doing this loses the benefits of load-balancing provided by the Sun HPC System, since the local node may be heavily loaded, and execution will therefore probably take longer than on the node chosen by the system. However, this might not matter to you if it is a brief program.

Note – You cannot run Sun MPI or Sun HPF programs in this way; you must use `tmrun` or `tmsub`.

3.18 Sending a Signal to a Process

The `tmkill` command is comparable to the Solaris `kill` command. You use it to terminate all processes of the tasks with the specified task IDs running on the Sun HPC System, or to send a signal to it; see Section 1.4.3 for a discussion of tasks. You can send any standard Solaris signal; use the `-l` option to obtain a list of the supported signals, or the `-d` option to list them along with brief descriptions.

Specify the signal's name or number, followed by the task ID, to send that signal to the task. For example,

```
% tmkill -CONT t59
```

sends a `SIGCONT` to the processes that constitute task t59.

Issuing `tmkill` without specifying a signal sends a `SIGTERM` to the task.

Use the command `tmps` to obtain a task ID (see Section 4.1) or use the `-T` option to `tmrun` (see Section 3.13).

3.18.1 `tmkill` Status

`tmkill` returns these status values:

- `0` – The command executed successfully.
- `1` – An error was encountered during execution. For example, the task was not known.
- `2` – The command was partially successful. This typically occurs when you send a signal to a task in which one or more of the processes had already exited and therefore couldn't receive the signal. This is usually not an error, since it is probably the reason you're using `tmkill` to eliminate the task in the first place.

CHAPTER 4

Getting Information

The Sun HPC RTE contains several commands for obtaining information about the Sun HPC System, its components, and tasks running on it.

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*

See Section 2.2.3 for general information on issuing Sun HPC commands.

4.1 Finding Out Task Status: The `tmps` Command

The `tmps` command is comparable to the Solaris `ps` command. It gives information about tasks, batch jobs, and processes currently running on the Sun HPC System.

By default `tmps` shows basic information about the user's tasks currently running in the default partition. For example,

```
% tmps
    TID   NPROC  UID    STATE  AOUT
     t41     3   slu    RUN    AAA
     t46     4   slu    EXNG   tmp
     t49     1   slu    EXIT   tmp
     t99     9   slu    EXNG   uname
    t100     9   slu    EXNG   uname
```

In the response,

- TID is the executing program's task ID.
- NPROC is the number of processes in the task.
- UID is the user ID of the person who executed the program.
- STATE is the execution status of the program. (See TABLE 4-1 for a list of possible states.)
- AOUT is the name of the executable program.

TABLE 4-1 Task states.

State	tmps Display	Meaning
CORE	CORE	The task exited due to a signal, and core was dumped.
COREING	CRNG	The task is exiting due to a signal. The first process to die also dumped core.
EXIT	EXIT	The task exited normally.
EXITING	EXNG	The task is exiting. At least one process exited normally.
FAIL	FAIL	The task failed on startup or was aborted.
FAILING	FLNG	The task's initialization failed, or a task abort has been signaled.
RUNNING	RUN	The task is running.
SEXIT	SEXIT	The task exited due to a signal.
SEXITING	SEXNG	The task is exiting due to a signal. The first process to die died due to a signal. At least one of its processes is still in the RUN state.
SPAWNING	SPAWN	The task is being spawned.

Use the -f option to display, in addition, the start time for each task and the task's arguments.

Use the -e option to display information on all tasks, not just your tasks.

4.1.1 Specifying the Partition

To show information about tasks running in all partitions, use the -A option.

To show information about tasks running in a specific partition, use the -a option, followed by the name of the partition.

4.1.2 Displaying Process Information

Use the –p option to also view information about the processes that make up the tasks. The process information is listed below each task. For example,

```
% tmps -p
  TID   NPROC    UID     STATE   AOUT
          RANK    PID   STATE   NODE
   t1   4       crawford  RUN     tmp
            3   17691    RUN     ultra-node0
            2   23449    RUN     ultra-node1
            1   17688    RUN     ultra-node0
            0   23446    RUN     ultra-node1
```

In this example,

- RANK is the process's rank within the task.
- PID is the process's process ID.
- STATE is the process's execution status.
- NODE is the node on which the process is running.

4.1.3 Displaying Specific Process, Task, and Job Information

In addition, you can use the –T option to display one or more specific task values, the –P option to display one or more specific process values, the –J option to display one or more job values. Separate multiple values either with spaces or with commas and no spaces.

Arguments to –T are

- tid – a specific task ID
- nproc – the number of processes in the task
- uid – the user ID of the owner of the task
- gid – the group ID of the owner of the task
- state – the current state of the task
- wkdir – the working directory of the owner of the task
- aout – the name of the executable program
- paout – the path of the executable program
- running – the number of running processes in the task

You can list these via the –lt option.

Arguments to –P are

- rank – the rank of the process within the task
- pid – the process's process ID
- state – the current execution state of the process

- `iod` – the process ID of the I/O daemon for this process.
- `load` – the load on the node on which the process is executing.
- `node` – the name of the node on which the process is executing.

You can list these via the `-lp` option.

Arguments to `-J` are

- `part` – the name of the partition in which the job will run.
- `queue` – the name of the queue to which the job was submitted.
- `jid` – the job's unique ID, which can be used as an argument to `tmkill`.
- `nproc`– the number of processes requested (the actual number of processes started may differ if the `–W` or `–S` flags were used with `tmsub`).
- `uid` – the user on whose behalf the job will be run (normally the user who submitted the job, but see the `–U` flag to `tmsub`).
- `gid` – the group on whose behalf the job will be run (normally the group of the user who submitted the job, but see the `–G` flag to `tmsub`).
- `state` – there are six states:
  - `BUILD` – The job is being submitted.
  - `WAIT` – The job is waiting to run.
  - `SPAWN` – The job is preparing to run.
  - `RUN` – The job is running.
  - `RSTRT` – The job has been killed because one of the nodes on which it was running went down; the job will be restarted.
  - `SIDE` – The job is at the head of its queue, but unable to run because the resources it requires are not available; the system is looking for another job to run while this job is waiting for its resources.
- `running` – the number of processes actually running for this job. This is not always equal to the number of processes started for this job, since processes that have exited are not counted.
- `wkdir` – the directory in which the job's processes will be (or were) started.
- `aout` – the name of the program to be run.
- `paout` – the full path of the program to be run.
- `ctime` – the job creation time (when `tmsub` was run).
- `args` – the command-line arguments for the program to be run.
- `stime` – the time the job was started.
- `prio` – the job priority (higher numbers run first).

4.1.4 Displaying Batch and Queue Information

Use the `tmps –B` (Batch option) command to obtain information about the current status of jobs in batch queues on a Sun HPC System. You can also use `tmkill` to remove one or more jobs from a batch queue.

Issuing the `tmps –B` command displays a list of all of your jobs in all queues, arranged by job ID. For example,

```
% tmps –B
  JID   QUEUE    UID     STATE  AOUT
  j10   q1       igb     WAIT   hostname
  j11   q1       igb     WAIT   vi
  j12   q1       igb     WAIT   cat /var/adm/mess
```

To remove a job from a queue, issue `tmkill`, followed by the job ID or task ID. For example,

```
% tmkill j10
```

You can remove only your own jobs (unless you are superuser). For further information on `tmkill`, see Section 3.18.

You can use the `–q` argument to specify the queue that you want `tmps` to display

```
% tmps –q queuename
```

4.2 Configuration and Status Information

Use the `tminfo` command to display information about the configuration of partitions, queues, and nodes, and current status information about nodes (for example, load and free memory).

4.2.1 Overview

You can display information on all partitions, queues, or nodes, or on any subset of them. You can either list the partitions, queues, or nodes, or you can use the `–R` option, along with a resource requirement specifier (RRS), to have the RTE determine which objects should be displayed. See Section 3.7.4 for information on RRSs. If you specify a partition, you must include only partition attributes in the RRS; if you specify a queue, you must use only queue attributes, and so on.

Use the –A option to specify an attribute whose value you want to display. If you want to display more than one attribute, separate them by commas with no spaces; or you can issue multiple –A options on the same command line. If you omit –A, `tminfo` displays values for a default set of attributes.

Use the –v option to display information about all attributes (including system administrator–defined attributes) for one or more partitions, queues, or nodes.

In displaying the value of a Boolean attribute, `yes` indicates that the attribute is set, and `no` indicates that the attribute is not set.

4.2.2 Partitions

To display information for all partitions, use the –P option.

To display information about an individual partition, use the –p option, followed by the name of the partition. To display information about multiple partitions, separate their names with commas and no spaces or enclose the list in quotation marks.

Partition attributes whose settings you can view via `tminfo` are shown in TABLE 4-2; the heading displayed for each attribute is shown in parentheses after its description.

Note these points:

- You can specify one or more of these attributes via the –A option, or as part of an RRS as an argument to the –R option. You can use either the attribute's real name or, in some cases, a shorter version.
- For attributes that are defined as negatives (for example, `no_logins`), you can specify a positive version (for example, `logins`) for –A. Note that the tabular display, shown in the examples below, is in terms of the positive version. For example, the column is labeled `LOG`, rather than `NO_LOG`.
- You can list the settings of all attributes (including any system administrator–defined attributes) on a per-partition basis via the –v option.
- You can list the names and brief descriptions of these attributes via the –lp option.

TABLE 4-2 Partition attributes available via `tminfo`.

Attribute (`tmadmin` form)	Short Form	Description (`tminfo` output heading)
`default_queue`	`defq`	Default queue (DEFAULT QUEUE).
`enabled`		Set if the partition is enabled, that is, if it is ready to accept logins, jobs, or tasks (ENA).
`max_batch_procs`	`maxb`	Maximum number of simultaneously running batch processes allowed on each node of the partition (MAXB).
`max_total_procs`	`maxt`	Maximum number of simultaneously running processes allowed on each node of the partition (MAXT).
`name`		Name of the partition (NAME).
`no_interactive_tasks`		Allow interactive tasks (INT).
`no_logins`		Logins allowed (LOG).
`no_mp_tasks`		Allow multinode tasks (MP).
`nodes`		Number of nodes in the partition (NODES).
`queues`		Number of queues in the partition (QCNT).

The values of three `tmadmin` attributes are inverses of the corresponding `tminfo` attributes:

- When `no_interactive_tasks` is unset, `INT` is set.
- When `no_logins` is unset, `LOG` is set.
- When `no_mp_tasks` is unset, `MP` is set.

Here is an example of the default `tminfo` output for partitions:

```
% tminfo -P
NAME         NODES ENA LOG INT MP  DEFAULT QUEUE
cigs             4 yes yes yes yes lonely
empty            - no  yes yes yes -
```

This example displays the names, numbers of nodes, and enabled status for all partitions:

```
% tminfo -A name,enabled,nodes -P
NAME         ENA NODES
cigs         yes     4
empty        no      -
```

4.2.3 Queues

To display information about all queues, use the `-Q` option.

To display information about an individual queue, use the `-q` option, followed by the name of the queue. To display information about multiple queues, separate their names with commas and no spaces.

Queue attributes whose settings you can see via `tminfo` are shown in TABLE 4-3; the heading displayed for each attribute is shown in parentheses after its description.

Note these points:

- You can specify one or more of these attributes via the `-A` option, or as part of an RRS as an argument to the `-R` option. You can abbreviate the `running` attribute to `run` (only with the `-A` option).
- You can list the settings of all attributes (including any system administrator–defined attributes) on a per-queue basis via the `-v` option.
- You can list the names and brief descriptions of these attributes via the `-lq` option.

TABLE 4-3 Queue attributes available via `tminfo`.

Attribute	Description (`tminfo` output heading)
`enabled`	Set if the queue is enabled, that is, if it is ready to accept tasks (`ENA`).
`name`	Name of the queue (`NAME`).
`partition`	Partition to which the queue is allowed to submit tasks (`PARTITION`).
`policy`	Determines how the run-time system is to treat jobs whose requirements exceed currently available resources (`POLICY`).
`running`	Set if the queue is running, that is, if it is submitting tasks to the partition (`RUN`).

Here is a sample of `tminfo` output for queues:

```
% tminfo -Q
NAME      PARTITION ENA RUN
free      cigs      yes yes
another   cigs      no  no
```

This example displays all queues in partition cigs that are running:

```
% tminfo -Q -p cigs -R "running"
NAME      PARTITION ENA RUN
free      cigs      yes yes
```

4.2.4 Nodes

To display information about all nodes, use the `-N` option.

To display information about an individual node, use the `-n` option, followed by the name of the node. To display information about multiple nodes, separate their names with commas and no spaces.

Node attributes whose settings you can see via `tminfo` are shown in TABLE 4-4; the heading displayed for each attribute is shown in parentheses after its description.

Note these points:

- You can specify one or more of these attributes via the `-A` option, or as part of an RRS as an argument to the `-R` option. You can use either the attribute's real name or, in some cases, a shorter version.
- You can list the settings of all attributes (including any system administrator–defined attributes) on a per-node basis via the `-v` option.
- You can list the names and brief descriptions of these attributes via the `-ln` option.

TABLE 4-4 Node attributes available via tminfo.

Attribute	Short Form	Description (`tminfo` output heading)
`cpu_idle`	`idle`	Percent of time CPU is idle (`IDLE`).
`cpu_iowait`	`iowait`	Percent of time CPU spends waiting for I/O (`IWAIT`).
`cpu_kernel`	`kernel`	Percent of time CPU spends in kernel (`KERNL`).
`cpu_swap`	`swap`	Percent of time CPU spends waiting for swap (`SWAP`).
`cpu_type`	`cpu`	CPU architecture (`CPU`).
`cpu_user`	`user`	Percent of time CPU spends running user's program (`USER`).
`domain`		DNS domain.
`enabled`		If set, node is available for spawning tasks on it.
`load1`		Load average for the past minute (`LOAD1`).
`load5`		Load average for the past five minutes (`LOAD5`).

Attribute	Short Form	Description (`tminfo` output heading)
load15		Load average for the past 15 minutes (LOAD15).
manufacturer	manuf	Hardware manufacturer (MANUFACTURER).
mem_free	memf	Node's available RAM (in Mbytes) (FMEM).
mem_total	memr	Node's total physical memory (in Mbytes) (MEM).
name		Name of the node (NAME).
ncpus	ncpu	Number of CPU modules in the node (NCPU).
os_arch_kernel	mach	Node's kernel architecture (MACH).
os_max_proc	maxproc	Maximum number of processes allowed on the node (note that this is *all* processes, including system daemons) (MPROC).
os_name	os	Name of the operating system running on the node (OS).
os_release	osrel	Operating system's release number (OSREL).
os_release_maj	osmaj	The major number of the operating system release number (MAJ).
os_release_min	osmin	The minor number of the operating system release number (MIN).
os_version	osver	Operating system's version (OSVER).
partition		The partition of which the node is a member (PARTITION).
serial_number	serno	Hardware serial number (SERIAL).
swap_free	swapf	Node's available swap space (in Mbytes) (FSWP).
swap_total	swapr	Node's total swap space (in Mbytes) (SWAP).

Here is a sample of the `tminfo` output for nodes:

```
% tminfo –N
 patton–tm 87 =>tminfo –N
NAME    UP PARTITION  OS    OSREL NCPU   FMEM   FSWP LOAD1 LOAD5 LOAD15
lucky    y p1        SunOS  5.5.1    1   0.89 158.34  0.09  0.11  0.13
camel    y p1        SunOS  5.5.1    1  31.41 276.12  0.00  0.01  0.01
vantage  y p1        SunOS  5.5.1    1  25.59 279.77  0.00  0.00  0.01
winston  y p1        SunOS  5.5.1    1  25.40 279.88  0.00  0.00  0.01
```

This example shows only the names of nodes and the partition they're in:

```
% tminfo –N –A name,partition
NAME         PARTITION
lucky        cigs
camel        cigs
vantage      cigs
winston      cigs
```

4.2.5 System

To display information about the entire Sun HPC System, use the `tminfo` command with the `–S` option. For example,

```
% tminfo –S
NAME ADMINISTRATOR DEF_BATCH_PART DEF_INTER_PART DEF_PFS
rog  simons        batch          general        pfs2
```

where:

- `NAME` – The name of the System
- `ADMINISTRATOR` – The name of its administrator
- `DEF_BATCH_PART` – The default batch partition
- `DEF_INTER_PART` – The default interactive partition
- `DEF_PFS` – The default parallel file system

4.3 Using the `tmadmin` Command

The `tmadmin` command is primarily used by the system administrator in configuring a Sun HPC System. Users can also issue the command to view, but not change, configurations. Typically you will not need to do this, since the same information is available more easily via `tminfo`. If you are interested, however, you can issue the command. It is in the directory `/opt/SUNWhpc/etc`, which is typically not in users' paths, so you would have to issue the command as follows:

```
% /opt/SUNWhpc/etc/tmadmin
```

This starts up an interactive interface from which you can issue `tmadmin` commands such as `show` and `list`. You can also change levels within this interface to obtain information about partitions, nodes, queues, and networks. Type `help` to display the commands available to you at any level of the interface. For more information, see the *Sun HPC Software System Administrator's Guide*.

4.4 Getting Help

4.4.1 Sun Online Documentation

To bring up online documentation for Solaris and the Sun compilers, use Sun's AnswerBook facility. To do this, you must be running OpenWindows or the Common Desktop Environment (CDE).

Before starting, you must set your `DISPLAY` environment variable to specify the terminal or workstation from which you are running OpenWindows or CDE. For example, if your workstation is named Valhalla, issue this command (if you are running the C shell):

```
% setenv DISPLAY valhalla:0
```

To start AnswerBook, issue this command:

```
% answerbook
```

4.4.2 Man Pages

Use the Solaris `man` command to view online manual pages for Solaris and Sun HPC Software commands and routines. For example,

```
% man tmps
```

displays information about the `tmps` command.

CHAPTER 5

Debugging Programs

Prism is a component in the Sun HPC Parallel Development Environment. You can use it to debug and visualize data in C, HPF, or Fortran serial or message-passing programs on a Sun HPC System. For complete information on Prism, see the *Prism User's Guide* and *Prism Reference Manual*. This chapter gives a brief overview of how to start up Prism.

To use Prism, you must first log in to the Sun HPC System, as described in Chapter 2. If you are using the graphical version of Prism, you must be running the Solaris 2.5.1 or 2.6 with either OpenWindows or CDE, and your `DISPLAY` environment variable must be set to the name of your terminal or workstation. For example (in the C shell),

```
% setenv DISPLAY mysun:0
```

You then execute Prism as you would any other program. For example,

```
% prism
```

You can then load your executable program within Prism. Or, you can specify the program's name on Prism's command line, and Prism comes up with the program already loaded. For example,

```
% prism a.out
```

Once the program is loaded in Prism, you can execute it, debug it, and visualize data in it. The program executes on the same node as Prism.

Note – Prism does not debug programs at the thread level.

5.1 Debugging Sun MPI and Sun HPF Programs

Specify the `-np` option if you are going to use Prism to debug a Sun MPI program or a Sun HPF program; its argument indicates how many processes are to be started. For example,

```
% prism -np 4 mpi_program
```

or

```
% prism -np 4 hpf_program
```

The behavior here is slightly different from that of Prism when debugging a serial program. When you use the `-np` option, you can also use other Prism options, such as `-p`, to determine where the Sun MPI, or Sun HPF processes are to run and how they are mapped onto nodes. For example,

```
% prism -p Deimos -np 4 mpi_program
```

or

```
% prism -p Deimos -np 4 hpf_program
```

starts Prism, and starts the Sun MPI, or Sun HPF processes in the partition Deimos. (Actually, client Prisms are also started with each of the parallel processes; they receive instructions from and send back information to the master Prism that is started by `tmrun`)

You can attach to a running Sun MPI, or Sun HPF program by specifying its task ID after the name of the executable program. For example,

```
% prism mpi_program t462
```

or

```
% prism hpf_program t462
```

You can obtain the task ID of a program by issuing the `tmps` command or by using the `-T` option to `tmrun`.

The setting of the `TMRUN_FLAGS` environment variable applies to both `tmrun` starting Prism and to Prism starting the parallel processes. This means that the default options are likely to be incorrect for one or the other, since you would typically want to start Prism on one node in a shared partition, and the Sun MPI or Sun HPF processes on multiple nodes, possibly in a dedicated partition.

5.1.1 Setting MPI_INIT_TIMEOUT

Sun MPI has timeouts built into the software to help detect when there are problems starting an MPI task. However, these timeouts can be triggered erroneously when you are debugging programs, such as when using Prism, and should therefore be disabled prior to using a debugger on a Sun MPI program. The environment variable `MPI_INIT_TIMEOUT` can be used to lengthen or disable the timeout time. When `MPI_INIT_TIMEOUT` is set to a positive integer, the timeout value is set to that time in seconds. When it is set to 0 or a negative integer, the timeout is disabled. The default value is 600 seconds (10 minutes).

For example, to disable timeouts (in a C shell):

```
% setenv MPI_INIT_TIMEOUT -1
```

Again in a C shell, to set timeouts to 5 minutes:

```
% setenv MPI_INIT_TIMEOUT 300
```

5.2 Debugging PVM Programs

You can attach to a running PVM task. Specifying the pid of one of the processes causes Prism to attach to all of those processes in the PVM task that are running the same executables as the specified process.

A PVM program must already be running before you can begin working with it; once it is running, if you specify one process's pid, Prism will attach to all of the processes belonging to the same task as the specified process. You must run Prism on a node that is running a PVM daemon; this typically means you would start Prism without `tmrun`, since using these commands couldn't necessarily guarantee the node on which Prism would run. Use the `-pvm` option to indicate that you are going to be working on a PVM program. Follow this with the name of the PVM executable program, and the pid of any of the program's processes. You can obtain the pid via the `ps` command.

For example,

```
% prism -pvm pvm_program 652
```

starts Prism on your login node and attaches it to the program `pvm_program`.

CHAPTER 6

Parallel File System

From the user's perspective, PFS closely resembles UNIX file systems. It uses a conventional inverted-tree hierarchy, with a `root` directory at the top and subdirectories and files branching down from there. The fact that individual PFS files are distributed across multiple disks managed by multiple I/O servers is transparent to the programmer. The way that PFS files are actually mapped to the physical storage facilities is implementation dependent and is based on file system configuration entries in the run-time environment (RTE) database.

The most obvious difference between PFS and Solaris file systems is in file pathname construction. Where a full Solaris pathname begins with `/` (`root`), a PFS pathname takes the form `pfs:`*filesystem*`:/`*pathname*.

- The first term, `pfs:`, is required, except when specifying a file name in your current working directory.
- *filesystem:* is the name of the file system—an arbitrary ASCII string assigned by the system administrator when configuring the file system. It is terminated by a *:* (colon). *filesystem:* can be omitted if there is only one PFS.
- The third term (*/pathname*) is the file's UNIX-style pathname.

For example, the PFS file `/users/clovis/paris` in the file system `cities` would be named `pfs:cities:/users/clovis/paris`.

PFS supports the current working directory (`.`) concept. That is, if the PFS file is below the current working directory, you need only to specify that portion of the path between `.` and the file. For example, if you are in `/users/clovis` and you want to specify the file `pfs:cities:/users/clovis/paris`, simply use the file name `paris`. The following example uses the `tmcp` command to copy a file within the PFS system:

```
% tmcp paris detroit
```

User programs access PFS via calls to Sun MPI I/O library routines. See the *Sun MPI User's Guide* for information about the Sun MPI I/O facility. See the *Sun HPF Guide* for information about the Sun HPF interface to the Sun MPI I/O library.

PFS provides various utilities for operating on PFS files. They will all be familiar to users of comparable Solaris file management programs. These utilities, which can be invoked from the command line, are summarized in TABLE 6-1. See the applicable man pages for details on their behavior.

TABLE 6-1 General PFS Utilities.

`tmcd`	Set the PFS current-working-directory.
`tmchgrp`	Change the group of a PFS file or directory. Note: `tmchgrp` does not support the `-f` and `-h` options provided by `chgrp`.
`tmchmod`	Change the access protection bits of a file. Note: `tmchmod` does not support the `-f` option provided by `chmod`.
`tmchown`	Change the owner of a PFS file or directory. Note: `tmchown` does not support the `-f` and `-h` options provided by `chown`.
`tmcmp`	Compare the contents of two PFS files.
`tmcp`	Copy a PFS file or directory within the PFS system. Note: `tmcp` does not support the `-R` option provided by `cp`.
`tmimport/` `tmexport`	Move data between PFS and Solaris file systems. Note: These options do not have Solaris equivalents.
`tmln`	Create a hard link to a PFS file. Note: PFS does not support symbolic links; `tmln` does not support the `-s` option provided by `ln`.
`tmls`	List the contents of PFS directories. Note: `tmls` does not support the `-AbCdfFLmpqsx1` options provided by `ls`.
`tmmkdir`	Create a PFS directory. Note: `tmmkdir` does not support the `-p` option provided by `mkdir`.
`tmmv`	Move a PFS file or directory to another location in the PFS system. Note: `tmmv` does not support the `-f` option provided by `mv`.
`tmpwd`	Print the current working directory.
`tmrm`	Remove a PFS file. Note: `tmrm` does not support the `-R` option provided by `rm`.
`tmrmdir`	Remove a PFS directory. Note: `tmrmdir` does not support the `-R` option provided by `rmdir`.

Note – `tmexport` and `tmimport` are functionally equivalent to `cp`, except they copy files or directories between PFS and Solaris file systems. Use `tmimport` and `tmexport` to backup and restore PFS data.

PFS also provides a set of PFS administration utilities. See the *Sun HPC System Administrator's Guide* of a discussion of these utilities.

Index

T

U

V

W