
LSF Parallel User's Guide

LSF Version 3.2
First Edition, August 1998

Platform Computing Corporation

LSF Parallel User's Guide

Copyright © 1998 Platform Computing Corporation
All rights reserved.

This document is copyrighted. This document may not, in whole or part, be copied, duplicated, reproduced, translated, electronically stored, or reduced to machine readable form without prior written consent from Platform Computing Corporation.

Although the material contained herein has been carefully reviewed, Platform Computing Corporation does not warrant it to be free of errors or omissions. Platform Computing Corporation reserves the right to make corrections, updates, revisions or changes to the information contained herein.

UNLESS PROVIDED OTHERWISE IN WRITING BY PLATFORM COMPUTING CORPORATION, THE PROGRAM DESCRIBED HEREIN IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL PLATFORM BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

LSF, LSF Base, LSF Batch, LSF JobScheduler, LSF MultiCluster, LSF Make, LSF Analyzer, LSF Parallel, Platform Computing, and the Platform Computing and LSF logos are trademarks of Platform Computing Corporation.

Other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

Printed in Canada.

Revision Information for the LSF Parallel User's Guide

Edition	Description
First	This document describes the LSF Parallel system released with LSF Suite version 3.2.

Table of Contents

Preface	vii
Audience	viii
Related Publications	ix
Typographic Conventions	xi
Command Notation	xii
LSF Suite 3.2	xiii
Technical Assistance	xv
1. Introduction	1
What Is the LSF Parallel System?	2
How Does LSF Parallel Fit Into the LSF Batch System?	3
LSF Parallel Architecture	5
2. Getting Started	9
Writing a Distributed Application	10
Compiling and Linking the Application	11
Running the Application	12
3. Building Parallel Applications	13
Including the Header File	14
Compiling and Linking	15
Building a Heterogeneous Parallel Application	17
4. Submitting Parallel Applications	19
Job Submission Methods	20
Batch Execution	22
Batch Job Status	23
Submitting Batch Jobs	26
Suspending Jobs	28
Resuming Jobs	29
Monitoring Job Status	30
Terminating Jobs	31
Running Heterogeneous Parallel Applications	32

Interactive Execution	34
The pam Command	35
Process Status Report	38
Getting Host Information	39
A. Vendor MPI Implementations	41
HP MPI	42
SGI MPI	43
SUN HPC MPI	44
Index	47

Preface

The *LSF Parallel User's Guide* describes how to compile and link, execute, interact and monitor parallel applications submitted through the LSF Suite of products.

For the most part, this guide does not repeat information that is available in detail elsewhere but focuses on what is specific to using the Platform Computing Corporation LSF Parallel system. References to more general sources are provided in *Related Publications* on page ix in this preface.

This preface discusses the following topics:

Audience.....	viii
Related Publications	ix
Related Publications	ix
Typographic Conventions	xi
Command Notation	xii
LSF Suite 3.2.....	xiii
Technical Assistance.....	xv

Audience

This guide provides reference and tutorial material for:

- MPI programmers who want to compile and link MPI programs for use with the LSF Parallel system
- Users of the LSF Parallel system who want to submit (execute), monitor, and interact with parallel applications using the LSF Batch system.

The users of this guide are expected to be familiar with:

- Programming in the C or Fortran 77 language
- Message Passing Interface (MPI) concepts
- The LSF Batch system.

Related Publications

This guide focuses on using parallel applications with the LSF Suite of products, primarily the LSF Batch system. It assumes familiarity with the LSF Suite of products and the MPI standard. The following materials provide useful background about using the LSF Suite of products and MPI.

These guides are available from Platform Computing Corporation:

- *LSF Suite Installation Guide*
- *LSF Batch Administrator's Guide*
- *LSF Batch Administrator's Quick Reference*
- *LSF Batch User's Guide*
- *LSF Batch User's Quick Reference*
- *LSF JobScheduler Administrator's Guide*
- *LSF JobScheduler User's Guide*
- *LSF Analyzer User's Guide*
- *LSF Programmer's Guide*

This document is available on the World Wide Web:

MPI: A Message-Passing Interface Standard, Message Passing Interface Forum (University of Tennessee, 1995)

<http://www.mcs.anl.gov/mpi/mpi-report-1.1/mpi-report.html>

These books should be available at your local bookstore:

MPI: The Complete Reference, by Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, and Jack Dongarra (MIT Press, 1995)

Using MPI, by William Gropp, Ewing Lusk and Anthony Skjellum (MIT Press, 1994)

Parallel Programming with MPI, by Peter Pacheco (Morgan Kaufmann Publishers, Inc., 1997)

Designing and Building Parallel Programs, Ian Foster (Addison-Wesley, 1995)

Preface

Related Publications

Online documentation supplied by Platform Computing Corporation:

Man pages (accessed with the man command) for all commands and MPI functions.

Typographic Conventions

Table 1 The typographic conventions used in this guide

Typeface	Meaning
AaBbCcDdEeFfGg	The names of commands, files, and directories; on-screen computer output
AaBbCcDdEeFfGg	What you type, contrasted with on-screen computer output
<i>AaBbCcDdEeFfGg</i>	Command-line place holder: replace with real name or value
<i>AaBbCcDdEeFfGg</i>	Book titles, new words or terms, or words to be emphasized

Command Notation

Table 2 The command notation used in this guide

Notation	Meaning
Apostrophes " or '	Must be entered exactly as shown
Commas ,	Must be entered exactly as shown
Ellipsis ...	The preceding parameter can be repeated. Do not enter the ellipsis
<i>lower case italics</i>	The parameter must be substituted
OR bar	You must enter one of the items separated by the bar. You cannot enter more than one item
Parenthesis ()	Must be entered exactly as shown
Parameter in square brackets []	The parameters within the brackets is optional. Do not enter the brackets
Stacked braces { } { }	You must enter one of the items within the braces. You cannot enter more than one item. Do not enter the braces
C shell prompt %	Unless otherwise noted, the C shell prompt is used in all command examples

LSF Suite 3.2

LSF is a suite of workload management products including the following:

- | | |
|-------------------------|--|
| LSF Batch | is a batch job processing system for distributed and heterogeneous environments, which ensures optimal resource sharing. |
| LSF JobScheduler | is a distributed production job scheduling application that integrates heterogeneous servers into a virtual mainframe or virtual super computer. |
| LSF MultiCluster | supports resource sharing among multiple clusters of computers using LSF products, while maintaining resource ownership and cluster autonomy. |
| LSF Analyzer | is a graphical tool for comprehensive workload data analysis. It processes cluster-wide job logs from LSF Batch and LSF JobScheduler to produce statistical reports on the usage of system resources by users on different hosts through various queues. |
| LSF Parallel | is a software product that manages parallel job execution in a production network environment. |
| LSF Make | is a distributed and parallel Make based on GNU Make that simultaneously dispatches tasks to multiple hosts. |
| LSF Base | is the software upon which all the other LSF products are based. It includes the network servers (LIM and RES), the LSF API, and load sharing tools. |

There are two editions of the LSF Suite:

LSF Enterprise Edition

Platform's LSF Enterprise Edition provides a reliable, scalable means for organizations to schedule, analyze, and monitor their distributed workloads across

Preface

LSF Suite 3.2

heterogeneous UNIX and Windows NT computing environments. LSF Enterprise Edition includes all the features in LSF Standard Edition (LSF Base and LSF Batch), plus the benefits of LSF Analyzer and LSF MultiCluster.

LSF Standard Edition

The foundation for all LSF products, Platform's Standard Edition consists of two products, LSF Base and LSF Batch. LSF Standard Edition offers users robust load sharing and sophisticated batch scheduling across distributed UNIX and Windows NT computing environments.

Technical Assistance

Contact your reseller for Technical Assistance. You can also contact the Technical Support Department at Platform Computing Corporation in the following ways:

- Mail: LSF Technical Support
Platform Computing Corporation
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7
- Telephone: +1 (905) 948-8448
- Toll-free: 1-87PLATFORM
+1 (877) 528-3676
- Fax: +1 (905) 948-9975
- Email: *support@platform.com*

Note When contacting Platform Computing please include the full name of your company.

Alternatively, you can find answers to many of your questions by visiting the Platform Computing Corporation home page on the World Wide Web. Point your browser to:

<http://www.platform.com>

Platform welcomes your comments and suggestions on this document, send email to *doc@platform.com*, or contact us by mail, fax, or phone.

Preface

Technical Assistance

1 Introduction

This chapter describes the LSF Parallel system and its architecture.

This chapter discusses the following topics:

What Is the LSF Parallel System?	2
How Does LSF Parallel Fit Into the LSF Batch System?	3
LSF Parallel Architecture	5

What Is the LSF Parallel System?

The LSF Parallel system is a fully supported commercial software system that supports the programming, testing, and execution of parallel applications in production environments.

The LSF Parallel system is fully integrated with the LSF Batch system, the de-facto industry standard resource management software product, to provide load sharing in a distributed system and batch scheduling for compute intensive jobs. The LSF Parallel system provides support for:

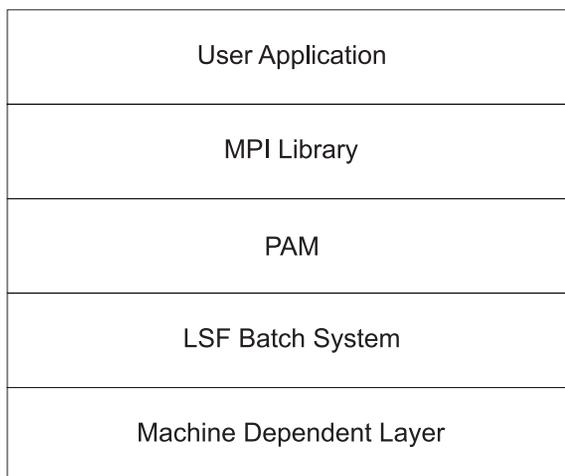
- Dynamic resource discovery and allocation (resource reservation) for parallel batch job execution
- Transparent invocation of the distributed job processes
- Full control of the distributed job processes to ensure no processes will become un-managed. This effectively reduces the possibility of one parallel job causing severe disruption to an organization's computer service.
- The standard MPI interface
- All major UNIX operating systems
- Full integration with the LSF Batch system providing heterogeneous resource-based batch job scheduling.

How Does LSF Parallel Fit Into the LSF Batch System?

The LSF Parallel system adopts a layered approach, shown in *Figure 1*, that is fully integrated with the LSF Batch system. In addition to the LSF Batch system resources, the following components make up the LSF Parallel system:

- The MPI Library
- The Parallel Application Manager (PAM).

Figure 1 Major Components of the LSF Parallel system



MPI Library

The Message Passing Interface (MPI) library is a message-passing library that must be linked to the parallel applications that are to be run in the LSF Batch system. The MPI library translates MPI message calls to messages for the machine-dependent layer and it interfaces the user application to PAM.

See *Vendor MPI Implementations* on page 41 for a description of vendor-specific MPI implementations.

1 Introduction

How Does LSF Parallel Fit Into the LSF Batch System?

PAM

The Parallel Application Manager (PAM) is the point of control for the LSF Parallel system. PAM is fully integrated with the LSF Batch system. PAM interfaces the user application with the LSF Batch system. For all parallel application processes (tasks), PAM:

- Maintains the communication connection map
- Monitors and forwards control signals
- Receives requests to add, delete, start, and connect tasks
- Monitors resource usage while the user application is running
- Enforces job-level resource limits
- Collects resource usage information and exit status upon termination
- Handles standard I/O.

LSF Batch System

The LSF Batch system is a sophisticated resource-based batch job scheduling system. It accepts user jobs and holds them in queues until suitable hosts are available and resource requirements are satisfied. Host selection is based on up-to-the-minute load information provided by the master Load Information Manager (LIM).

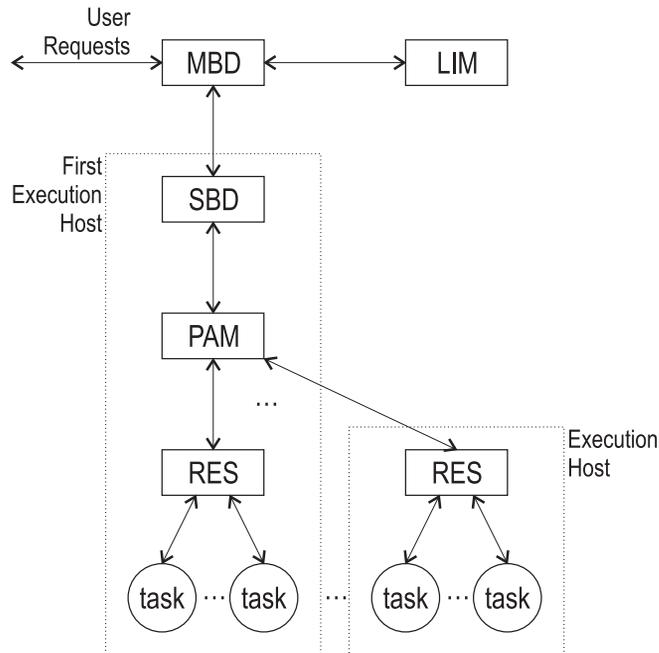
LSF Batch runs user jobs on batch server hosts. It has sophisticated controls for sharing hosts with interactive users; there is no need to set aside dedicated hosts for processing batch jobs.

See the *LSF Batch User's Guide* and the *LSF Batch Administrator's Guide* for a detailed description of the LSF Batch system.

LSF Parallel Architecture

The LSF Parallel system fully utilizes the resources of the LSF Batch System for resource selection and process invocation and control. The process of parallel batch job invocation and control is shown in *Figure 2* and described in *Table 1* on page 6. The LSF components shown in *Figure 2* are described in *Table 2* on page 6.

Figure 2 LSF Parallel Architecture



The LSF Parallel system also supports interactive parallel job submission. The process is similar to that shown in *Figure 2* except the user request is submitted directly to PAM which makes a simple placement query to LIM. Job queuing and resource reservations are not supported in interactive mode.

Table 1 LSF Parallel Batch Job Invocation

Stage	Description
1	User submits a parallel batch job to the LSF Batch system
2	MBD retrieves a list of suitable execution hosts from the master LIM
3	MBD allocates (schedules, reserves) the execution hosts for the parallel batch job
3	MBD dispatches the parallel batch job to the SBD on the first execution host that was allocated to the batch job
4	SBD starts PAM on the same execution host
5	PAM starts RES on each execution host allocated to the batch job
6	RES starts the tasks on each execution host

Table 2 Description of LSF Parallel Components

Part	Function
User Request	Batch job submission to the LSF Batch system using the <code>bsub</code> command.
MBD	The Master Batch Daemon is the policy center for the LSF Batch system. It maintains information about batch jobs, hosts, users, and queues. All of this information is used in scheduling batch jobs to hosts.
LIM	<p>The Load Information Manager is a daemon process running on each execution host. LIM monitors the load on its host and exchanges this information with the Master LIM.</p> <p>The Master LIM resides on one execution host and collects information from the LIMs on all other hosts in the LSF cluster. If the master LIM becomes unavailable, another host will automatically take over.</p> <p>For batch submission the master LIM provides this information is provided to the MBD.</p> <p>For interactive execution the Master LIM provides simple placement advice.</p>

Table 2 Description of LSF Parallel Components

SBD	The Slave Batch Daemons are batch job execution agents residing on the execution hosts. SBD receives jobs from the MBD in the form of a job specification and starts RES to run the job according the specification. SBD reports the batch job status to the MBD whenever job state changes.
PAM	The Parallel Application Manager is the point of control for the LSF Parallel system. PAM is fully integrated with the LSF System. PAM interfaces the user application with the LSF system. If PAM or its host crashes, each RES will terminate all tasks under its management. This avoids the problem of orphaned processes.
RES	The Remote Execution Servers reside on each execution host. RES manages all remote tasks and forwards signals, standard I/O, resources consumption data, and parallel job information between PAM and the tasks.
application task	The individual process of a parallel application
execution hosts	The most suitable hosts to execute the batch job as determined by the LSF Batch system.
first execution host	The host name at the top of the execution host list as determined by the LSF Batch system

1 **Introduction**

LSF Parallel Architecture

2 Getting Started

The purpose of this chapter is to quickly introduce the concepts needed to start using the LSF Parallel system. They are: compiling, linking, and submitting parallel applications. The example used in this chapter is a distributed version of the *Hello World* program named *myjob*, written in C.

This chapter discusses the following topics:

Writing a Distributed Application.....	10
Compiling and Linking the Application.....	11
Running the Application.....	12

Note If the commands cannot be executed or the man pages cannot be viewed, the appropriate directories may need to be added to the system's path; check with the system administrator.

Writing a Distributed Application

This example program, written in C, is a distributed version of the *Hello World* program named *myjob*. Use an editor to enter the code for this application. After the code is entered, save it in a file named `myjob.c`

```
/*
 * File: myjob.c
 */
#include <stdio.h>
#include "mpi.h"          /* MPI header file */

int
main(int argc, char **argv)
{
    int myrank;          /* Rank of this process */
    int n_processes;    /* Number of Processes */
    int srcrank;        /* Rank of the Sender */
    int destrank;       /* Rank of the receiver */
    char mbuf[512];     /* Message buffer */
    MPI_Status mstat;   /* Return Status of an MPI operation */

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    MPI_Comm_size(MPI_COMM_WORLD, &n_processes);

    if (myrank != 0) {
        sprintf(mbuf, "Hello, from process %d!", myrank);
        destrank = 0;
        MPI_Send(mbuf, strlen(mbuf)+1, MPI_CHAR,
                 destrank, 90, MPI_COMM_WORLD);
    } else {
        for (srcrank = 1; srcrank < n_processes; srcrank++) {
            MPI_Recv(mbuf, 512, MPI_CHAR,
                    srcrank, 90, MPI_COMM_WORLD, &mstat);
            printf("From process %d: %s\n", srcrank, mbuf);
        }
    }
    MPI_Finalize();
}
```

Compiling and Linking the Application

After the example program is entered and saved as `myjob.c`, use the `mpicc` script to compile and link the application. The `mpicc` script is used in a similar manner to other UNIX-based C compilers. This script provides the options and special libraries needed to compile and link a parallel application for the LSF Parallel environment.

Compile and Link

To compile and link the source code in the `myjob.c` file in one step, enter the following command:

```
% mpicc myjob.c -o myjob
```

The binary created is called `myjob`.

Running the Application

Submit to the LSF Batch System

To submit the parallel application `myjob` to the LSF Batch system, requesting three processors, enter the following command:

```
% bsub -n 3 pam myjob
Job <1288> is submitted to default queue <normal>.
```

This command creates three processes and each runs an instance of `myjob`. The `bsub` command has a number of command line options, which are discussed in more detail in *Submitting Batch Jobs* on page 26. To view the status of the parallel batch job, enter the following command:

```
% bjobs
JOBID USER      STAT  QUEUE          FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
1288  user1    PEND  normal         hopper     host1     myjob     Apr 16 14:43
                                     host2
                                     host3
```

The `bjobs` command has a number of command line options, which are discussed in more detail in *Monitoring Job Status* on page 30.

Execute Interactively

To interactively execute the parallel application `myjob` on three processors, enter the following command:

```
% pam -n 3 myjob
Server PAM contact address is klee:2801
From process 1: Hello, world from process 1!
From process 2: Hello, world from process 2!
From process 3: Hello, world from process 3!

TID  HOST_NAME  COMMAND_LINE  STATUS  TERMINATION_TIME
===  =====  =====  =====  =====
1    host1     myjob        Completed  04/16/98 15:05:56
2    host2     myjob        Completed  04/16/98 15:05:56
3    host3     myjob        Completed  04/16/98 15:05:56
```

The `pam` command has a number of command line options, which are discussed in more detail in *The pam Command* on page 35.

3 Building Parallel Applications

The LSF Parallel systems provides tools to help build a parallel application to take full advantage of the LSF Batch system. Most parallel applications can be reused by simply re-linking with the PAM-aware MPI library, in some instances there may not even be a need to re-compile.

This chapter discusses the basic steps in building a parallel application. We discuss the basic structure of the application and how it is compiled and linked.

Note This chapter focuses on building a parallel application to make optimal use of the LSF Batch system. It assumes familiarity with the LSF Suite of products and standard MPI. Therefore it does not discuss writing MPI programs.

This chapter contains the following topics:

Including the Header File.....	14
Compiling and Linking	15
Building a Heterogeneous Parallel Application.....	17

Including the Header File

A set of PAM aware header files are included with the LSF Parallel system installation. They are typically located in the `LSF_INCLUDEDIR/lsf/mpi/` directory. The header files contain the MPI definitions, macros, and function prototypes necessary for using the LSF Parallel system.

Include Syntax

The *include* syntax must be placed at the top of any parallel application that calls MPI routines. The include statement looks like this:

In C applications:

```
#include <mpi.h>
```

In Fortran 77 applications:

```
INCLUDE 'mpif.h'
```

Note If the header files are not located in the `LSF_INCLUDEDIR/lsf/mpi/` directory, check with your system administrator.

Compiling and Linking

The LSF Parallel system provides a set of scripts that help with the creation of executable objects. They are: `mpicc` for C programs and `mpif77` for Fortran 77 programs. These scripts provide the options and special libraries needed to compile and link MPI programs for use with the LSF Parallel system. Applications are linked to system-dependent libraries and the appropriate MPI library.

C Programs

The LSF Parallel C compiler, `mpicc`, is used to compile MPI C source files. It is used in a similar manner to other UNIX-based C compilers.

For example: To compile the sample program contained in a file `myjob.c` enter:

```
% mpicc -c myjob.c
```

This command produces the `myjob.o` that contains the object code for this LSF Parallel source file.

To link the `myjob.o` object file with the LSF Parallel libraries to create an executable, enter:

```
% mpicc -o myjob myjob.o
```

As with most C compilers, the `-o` flag specifies that the name of the executable produced by the linker is to be `myjob`.

The C source file can be compiled and linked in one step using the following command:

```
% mpicc myjob -o myjob
```

Fortran 77 Programs

The LSF Parallel Fortran 77 compiler, `mpif77`, is used to compile MPI Fortran 77 source files. It is used in a similar manner to other UNIX-based Fortran 77 compilers.

For example: To compile the sample program contained in a file `myjob.f` enter:

```
% mpif77 -c myjob.f
```

3 Building Parallel Applications

Compiling and Linking

This command produces the `myjob.o` that contains the object code for this LSF Parallel source file.

To link the `myjob.o` object file with the LSF Parallel libraries to create an executable, enter:

```
% mpif77 -o myjob myjob.o
```

As with most Fortran 77 compilers, the `-o` flag specifies that the name of the executable produced by the linker is to be `myjob`.

The Fortran 77 source file can be compiled and linked in one step using the following command:

```
% mpif77 myjob -o myjob
```

Building a Heterogeneous Parallel Application

The LSF Parallel system provides a host type substitution facility to allow a heterogeneous multiple-architecture distributed application to be submitted to the LSF Batch system. The following steps outline how to build and deploy a heterogeneous application:

1. Design the parallel application.
2. Compile the application on all LSF host-type architectures that will be used to support this application.

Note The binaries must either be named with valid LSF host-type extensions or placed in directories named with valid LSF host-type path names.

3. Place binaries in the appropriate shared file system or distribute them accordingly.
4. Use the %a annotation to submit the parallel application to the LSF Batch system.

LSF Host Type Naming Convention

Binaries must be compiled on the target host type architectures. The binary must be named using a valid LSF host type string as the extension to its name or the name of a directory in its path (`lshosts` displays a list of valid LSF host types). When the %a notation is used to submit a parallel application to the LSF Batch system the target host type string is substituted.

All binaries for a specific application must be named using the same host type substitution format (i.e., binary extension or path name).

Example: The following binaries are named with appropriate host type extensions to identify the target platform on which they are to run. These binaries are named to use Sun Solaris and RS6000 architecture machines:

- `myjob.SUNSOL`
- `myjob.RS6K`

3 Building Parallel Applications

Building a Heterogeneous Parallel Application

Example: The following binaries are named with appropriate path names to identify the target platform on which they are to run. These binaries are named to use Sun Solaris and RS6000 architecture machines:

- /user/batch/SUNSOL/myjob
- /user/batch/RS6K/myjob

%a Notation

After a parallel application is submitted to the LSF Batch system, the Parallel Application Manager (PAM) replaces the %a annotation with the appropriate LSF host type string. PAM then launches the individual tasks of the application on the remote hosts using the correct binaries.

Note Use the `lshosts` command to determine which LSF hosts are available. For example:

```
% lshosts
HOST_NAME  type    model    cpuf  ncpus  maxmem  maxswp  server  RESOURCES
host1      SUNSOL  SunSparc 6.0   1      64M     112M   Yes (solaris cserver)
host2      RS6K    IBM350   7.0   1      64M     124M   Yes    (cserver aix)
```

Example: To submit the `myjob` application from the same directory using LSF host type extensions the following command is used:

```
% pam -n 2 myjob.%a
```

PAM will make the following substitutions for the %a notation:

- myjob.SUNSOL
- myjob.RS6K

Example: To submit the `myjob` application from different directories using host type path names the following command is used:

```
% pam -n 2 /user/batch/%a/myjob
```

PAM will make the following substitutions for the %a notation:

- /user/batch/SUNSOL/myjob
- /user/batch/RS6K/myjob

4 Submitting Parallel Applications

This chapter describes how to submit and interact with parallel applications in the LSF Batch system.

An extensive and flexible set of tools is provided that allows parallel applications to be submitted through the LSF Batch system. Parallel applications can also be executed interactively under control of the Parallel Application Manger (PAM). These tools allow the specification of how, when, and where a parallel application is to be run.

This chapter discusses the following topics:

Job Submission Methods	20
Batch Execution	22
Batch Job Status	23
Submitting Batch Jobs	26
Suspending Jobs	28
Resuming Jobs	29
Monitoring Job Status	30
Terminating Jobs	31
Running Heterogeneous Parallel Applications	32
Interactive Execution	34
The pam Command	35
Process Status Report	38
Getting Host Information	39

Job Submission Methods

The LSF Parallel system supports batch submission of parallel applications (batch jobs) using the facilities of the LSF Batch System. Interactive execution of parallel applications is also supported under control of the Parallel Application Manager (PAM).

Batch Execution

When submitting a parallel batch job, the LSF Parallel system uses the advanced features of the LSF Batch system to select, submit, and interact with the individual tasks of the parallel batch job. The batch job is submitted to a queue using the `bsub` command and the LSF Batch system attends to the details.

A parallel batch job is submitted to a queue, where it waits until it reaches the front of the queue and the appropriate resources become available. Then the batch job will be dispatched to the most suitable hosts for execution. This sophisticated queuing system allows batch jobs to run as soon as the suitable host resources becomes available.

Note The batch job may not be run immediately, it may be queued until the appropriate resources become available.

To use the `bsub` command to submit a parallel batch job to the LSF Batch system, see *Submitting Batch Jobs* on page 26.

Interactive Execution

When interactively executing a parallel batch job, the `pam` command is used to invoke PAM. When submitting batch jobs using the `pam` command, the LSF Batch system is bypassed; the jobs are not queued. Batch jobs are run immediately upon entering the command if the specified resource requirements are met. If the resources are not available the job is not run.

Since the jobs do not wait, interactive job execution is beneficial for debugging parallel applications. Direct interaction is supported. All the input and output is handled transparently between the local and execution hosts.

To use the `pam` command to execute a parallel batch job interactively, see *Interactive Execution* on page 34.

LSF Parallel and LSF Batch Commands

The LSF Batch and LSF Parallel products provide commands and man pages for these commands.

If these commands cannot be executed or the man pages cannot be viewed, the appropriate directories may need to be added to the systems path; check with the system administrator.

Batch Execution

The LSF Parallel system uses the features of the LSF Batch system to select the most suitable hosts, submit, and interact with parallel batch jobs. The batch job is submitted to a queue using the `bsub` command, as described in *Submitting Batch Jobs* on page 26, and the LSF Batch and LSF Parallel systems attend to the rest.

Like serial batch jobs, parallel batch jobs pass through many states. See *Batch Job Status* on page 23.

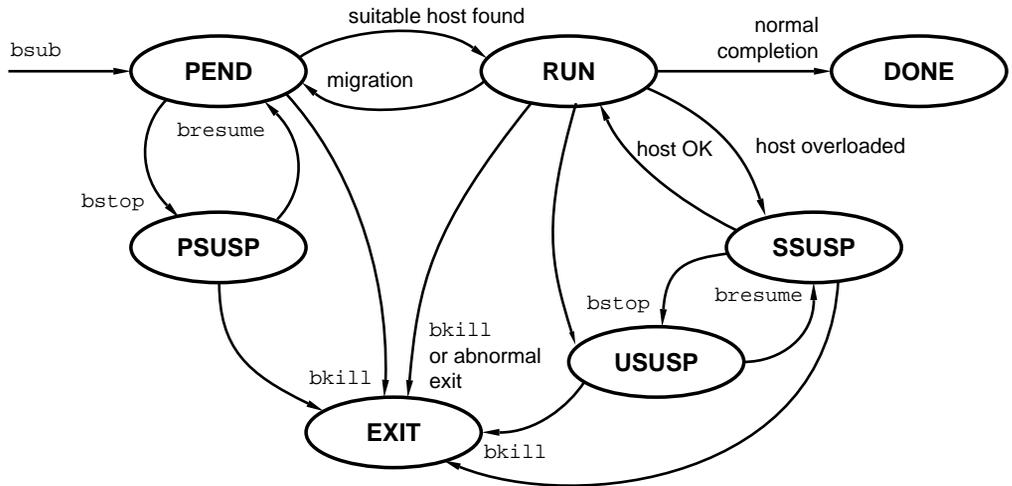
This part of the chapter discusses the following topics:

Batch Job Status	23
Submitting Batch Jobs	26
Suspending Jobs	28
Resuming Jobs	29
Monitoring Job Status	30
Terminating Jobs	31
Running Heterogeneous Parallel Applications	32

Batch Job Status

Each batch job submitted to the LSF Batch system passes through a series of states until the job completes normally (success) or abnormally (failure). The `bjobs` command allows the status of the batch jobs to be monitored; see *Monitoring Job Status* on page 30. The ability to monitor batch job status extends to the individual processes (tasks) of the parallel application.

Figure 3 Batch Job State Diagram



4 Submitting Parallel Applications

Batch Job Status

Figure 3 shows the possible states a batch job can pass through when submitted to the LSF Batch system. The diagram also shows the activities and commands that cause the state transitions. The batch job states are described in Table .

Table 3 Batch Job State Descriptions

State	Description
PEND	<p>A batch job is pending when it is submitted (using the <code>bsub</code> command) and waiting in a queue. It remains pending until it moves to the head of the queue and all conditions for its execution are met. The conditions may include:</p> <ul style="list-style-type: none">• Start time specified by the user when the job is submitted• Load conditions on qualified hosts• Time windows during which:<ul style="list-style-type: none">• The queue can dispatch jobs• Qualified hosts can accept jobs• Relative priority to other users and jobs• Availability of the specified resources.
RUN	<p>A batch job is running when it has been dispatched to a host.</p>
DONE	<p>A batch job is done when it has normally completed its execution.</p>
PSUSP	<p>The job owner or the LSF Administrator can suspend (using the <code>bstop</code> command) a batch job while it is pending.</p> <p>Also, the job owner or the LSF Administrator can resume (using the <code>brresume</code> command) a batch that is in the PSUSP state, then the batch job state transitions to PEND.</p>
USUSP	<p>The job owner or the LSF Administrator can suspend (using the <code>bstop</code> command) a batch job after it has been dispatched.</p> <p>Also, the owner or the LSF Administrator can resume (using the <code>brresume</code> command) a batch that is in the USUSP state, then the batch job state transitions to SSUSP.</p>

Table 3 Batch Job State Descriptions

State	Description
SSUSP	<p>A batch job can be suspended by the LSF Batch system after it has been dispatched. This is done if the load on the execution host or hosts becomes too high in order to maximize host performance or to guarantee interactive response time.</p> <p>The LSF Batch system suspends batch jobs according to their priority unless the scheduling policy associated with the job dictates otherwise. A batch job may also be suspended if the job queue has a time window and the current time exceeds the window.</p> <p>The LSF Batch system can later resume a system suspended (SSUSP) job if the load condition on the execution host decreases or the time window of the queue opens.</p>
EXIT	<p>A batch job can terminate abnormally (fail) from any state for many reasons. Abnormal job termination can occur when:</p> <ul style="list-style-type: none"> • Cancelled (using the <code>bkill</code> command) by owner or LSF administrator while in PEND, RUN, or USUSP state. • Aborted by LSF because job cannot be dispatched before a termination deadline • Fails to start successfully (e.g., the wrong executable was specified at time of job submission) • Crashes during execution.

Parallel Batch Job Behavior

- When one task exits with a none-zero return value all the other tasks will run until they complete (DONE) or fail (EXIT).
- When one task is killed by a signal or core dumps, all the other tasks will be shut down.

Submitting Batch Jobs

The bsub Command

The `bsub` command is used to submit parallel batch jobs to the LSF Batch system. The syntax for using `bsub` when submitting parallel applications is the same as the LSF Batch system with the addition of the `pam` option:

```
% bsub [options] pam [options] job
```

The pam Option

The `pam` options used with the `bsub` command are a subset of the `pam` command options, see *The pam Command* on page 35. Since the LSF Batch system does all of the resource allocation and scheduling, the `pam` options `-m`, `-f`, and `-n` are not necessary and are ignored by the `bsub` command. The syntax for `bsub pam` is:

```
pam [-h][--V][--t][--v]
```

The `bsub pam` options are:

Option	Description
-h	Print command usage to standard error and exit.
--V	Print LSF version to standard error and exit.
--t	Suppress the printing of the process status summary on job completion.
--v	Specifies the job is to be run in verbose mode. The names of the selected hosts are displayed.

Example: The following command submits a parallel batch job named `myjob` to the LSF Batch system and requests four processors of any type to run the job:

```
% bsub -n 4 pam myjob
```

When the parallel batch job named `myjob` is submitted to the LSF Batch system and dispatched to `host1`, `host2`, `host3` and `host4`, the `bjobs` command will display:

```
% bjobs
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
713	user1	RUN	batch	host99	host1	myjob	Sep 12 16:30
					host2		
					host3		
					host4		

Suspending Jobs

The bstop Command

The `bstop` command is used to suspend parallel batch jobs running in the LSF Batch system. The syntax for using the `bstop` command in the LSF Parallel system is:

```
% bstop jobId
```

Example: The following command suspends the parallel batch job named `myjob` running in the LSF Batch system with job id of 713:

```
% bstop 713
```

When the parallel batch job named `myjob` is suspended the `bjobs` command will display the batch job state of `USUSP`:

```
% bjobs
JOBID USER      STAT  QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
713  user1      USUSP  batch      host99     host1      myjob     Sep 12 16:32
                                     host2
                                     host3
                                     host4
```

Resuming Jobs

The bresume Command

The `bresume` command is used to resume suspended parallel batch jobs running in the LSF Batch system. The syntax for using the `bresume` command in the LSF Parallel system is:

```
% bresume jobID
```

Example: The following command resumes the suspended parallel batch job named `myjob` running in the LSF Batch system with job ID of 713:

```
% bresume 713
```

When the parallel batch job named `myjob` is resumed the `bjobs` command will display the batch job state of `RUN` or `PEND`:

```
% bjobs
JOBID USER      STAT  QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
713  user1      RUN   batch      host99     host1      myjob     Sep 12 16:34
                                     host2
                                     host3
                                     host4
```

Monitoring Job Status

The bjobs Command

The `bjobs` command is used to view the running status and resource usage of parallel batch jobs running in the LSF Batch system. The syntax for using the `bjobs` command in the LSF Parallel system is:

```
% bjobs [options]
```

Example: The following command displays the running status and resource usage of the jobs running in the LSF Batch system:

```
% bjobs
JOBID USER      STAT  QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
713  user1      RUN   batch      host99     host1      myjob     Sep 12 16:34
                               host2
                               host3
                               host4
```

Example: The following command uses the `-l` option to display run-time resource usage (*CPU*, *memory*, and *swap*) as well as the running status of the jobs running in the LSF Batch system:

```
% bjobs -l
Job Id <713>, User, Project, Status, Queue, Interactive pseudo-terminal mode,
Command <myjob>

Thu Sep 12 16:39:17: Submitted from host <host99>, CWD <${HOME}/Work/utopia/pass/
pam>, 2-4 Processors Requested;
Thu Sep 12 16:39:18: Started on 4 Hosts/Processors host1 host2 host3 host4,
Execution Home < /pcc/s/user1, Execution CWD < /pcc/s/user1/W
ork/utopia/pass/pam;
Thu Sep 12 16:40:41: Resource usage collected.
The CPU time used is 2 seconds.
MEM: 281 Kbytes; SWAP: 367 Kbytes
PGIDs: 4 PIDs: 4

SCHEDULING PARAMETERS:
          r15s  r1m  r15m  ut      pg    io   ls    it     tmp     swp     mem
loadSched  -    -    -    -      -    -   -    -     -     -     -
loadStop   -    -    -    -      -    -   -    -     -     -     -

          nresj
loadSched  -
loadStop   -
```

Terminating Jobs

The bkill Command

The `bkill` command is used to terminate parallel batch jobs running in the LSF Batch system. The syntax for using the `bkill` command in the LSF Parallel system is:

```
% bkill jobID [options]
```

Example: The following command terminates the parallel batch job named `myjob` running in the LSF Batch system with a job ID of 713:

```
% bkill 713
```

When the parallel batch job named `myjob` is terminated the `bjobs` command will display the batch job state of `EXIT`:

```
% bkill 713
JOBID USER      STAT  QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
713  user1      EXIT  batch      host99     host1      myjob     Sep 12 16:30
                                     host2
                                     host3
                                     host4
```

Note The time taken to terminate a parallel batch job varies and depends on the number of parallel processes.

Running Heterogeneous Parallel Applications

The LSF Parallel system provides an LSF host type substitution facility to allow a heterogeneous multiple-architecture distributed application to be submitted to the LSF Batch system.

Assumptions:

1. The binary will run on each specified platform, or a binary exists for each platform.
2. The binaries for the Parallel application are specified using the %a notation format, see *Building a Heterogeneous Parallel Application* on page 17.

Example: Using the **LSF host type extension format** to specify the batch job named myjob to run on any two available processors having either Sun Solaris (SUNSOL) or RS6000 (RS6K) architectures, the following command can be used:

```
% bsub -n 2 pam myjob.%a
```

To specify SUNSOL and RS6K in an environment with other architectures, the following command is specified with the -R (resource) option:

```
% bsub -n 2 -R "type==SUNSOL || type==RS6K" pam myjob.%a
```

For both these examples, the Parallel Application Manager (PAM) substitutes the %a notation with the correct LSF host type extension. The binaries used are named:

- myjob.SUNSOL
- myjob.RS6K

Example: Using the **LSF host type path name format** to specify the batch job named `myjob` to run on any two processors having either SUNSOL or RS6K architectures, the following command can be used:

```
% bsub -n 2 pam /user/batch/%a/myjob
```

To specify SUNSOL and RS6K in an environment with other architectures, the following command is specified with the `-R` (resource) option:

```
% bsub -n 2 -R "type==SUNSOL || type==RS6K" pam /user/batch/%a/myjob
```

For both these examples, the Parallel Application Manager (PAM) substitutes the `%a` notation with the correct LSF host type path name. The paths used to select the binaries are:

- `/user/batch/SUNSOL/myjob`
- `/user/batch/RS6K/myjob`

Interactive Execution

The LSF Parallel system uses the Parallel Application Manager (PAM) to control the execution of parallel batch jobs interactively. Batch jobs are executed interactively using the `pam` command, see *The pam Command* on page 35. When submitting batch jobs using the `pam` command, the LSF Batch system is bypassed, the jobs are not queued. Batch jobs are run immediately upon entering the command if the resource requirements specified are met. If the resources are not available the job is not run. Since the jobs do not wait, interactive job execution is beneficial for debugging parallel applications.

To successfully execute an interactive parallel batch job, the `pam` command must be reissued at a time when the resources are available. If specific resources are not requested the LSF Parallel system will run the batch job on the least loaded hosts that meet the batch jobs criteria.

Direct interaction is supported. All the input and output is handled transparently between local and execution hosts. All job control signals (e.g., `ctrl+x`, `ctrl+z`, and `ctrl+l`) are propagated to the execution hosts; this allows interaction with the job as if it were a being executed locally.

This part of the chapter discusses the following topics:

The <code>pam</code> Command	35
Process Status Report.....	38
Getting Host Information.....	39

The pam Command

The pam command is used to interactively execute parallel batch jobs in the LSF Parallel system. A subset of the pam command is used as a command option for the bsub command (see *The bsub Command* on page 26). The syntax for using the pam command is:

```
pam [-h][-V][-i][-t][-v]
    [-server_addr location    ]
    [| -server_jobid location ]
    [| -server_jobname location ]
    { -m "host ..."        }
    { | [-R req] -n num      }
    job [arg ...]
```

Option	Description
-h	Print command usage to standard error and exit.
-V	Print LSF version to standard error and exit.
-i	Specifies interactive operation mode, the user will be asked if application is to be executed on all hosts. If yes (y) the task is started on all hosts specified in the list. If no (n) the user must interactively specify the hosts.
-t	Suppress the printing of the job task summary report to the standard output at job completion.
-v	Specifies the job is to be run in verbose mode. The names of the selected hosts are displayed.
-server_addr <i>location</i>	Specifies the location of the PAM server. The location is specified in the hostname:port_no format.
-server_jobid <i>location</i>	Specifies the location of the PAM server. The location is specified using the jobid for the server PAM job.
-server_jobname <i>location</i>	Specifies the location of the PAM server. The location is specified using the jobname for the server PAM job.

4 Submitting Parallel Applications

The pam Command

Option	Description
-m " <i>host ...</i> "	Specifies the list of hosts on which to run the parallel batch job tasks. The number of host names specified indicates the number of processors requested. Note This option cannot be used with options -R or -n, and is ignored when pam is used as a bsub option.
[-R <i>req</i>] -n <i>num</i>	Specifies the number of processors required to run the parallel job. Note This option cannot be used with option -m, and is ignored when pam is used as a bsub option.
-R <i>req</i>	Specify the resource requirements for host selection. Execute the parallel job on the hosts that meet these requirements. <u>Default:</u> r15s:pg Note This option is ignored when pam is used as a bsub option.
<i>job</i> [<i>arg ...</i>]	The name of the parallel job to be run. Note This must be the last argument on the pam command line.

Example: The following command executes the parallel batch job named myjob on the LSF Parallel system requesting four processors of any type:

```
% pam -n 4 myjob
```

TID	HOST_NAME	COMMAND_LINE	STATUS	TERMINATION_TIME
1	host1	myjob	Completed	03/31/98 10:31:58
2	host2	myjob	Completed	03/31/98 10:31:59
3	host3	myjob	Completed	03/31/98 10:31:59
4	host4	myjob	Completed	03/31/98 10:31:58

Example: The following command uses the `-m` option to execute the parallel batch job named `myjob` on *host1*, *host2*, and *host3*:

```
% pam -m "host1 host2 host3" myjob
```

TID	HOST_NAME	COMMAND_LINE	STATUS	TERMINATION_TIME
1	host1	myjob	Completed	03/31/98 10:31:58
2	host2	myjob	Completed	03/31/98 10:31:59
3	host3	myjob	Completed	03/31/98 10:31:59

Process Status Report

After a parallel batch job terminates in a successful (Done) or failed (EXIT) state the LSF Parallel system displays the status of all the processes. For example:

```
% pam -n 4 myjob
```

TID	HOST_NAME	COMMAND_LINE	STATUS	TERMINATION_TIME
1	host1	myjob	Done	03/31/98 10:31:58
2	host2	myjob	Done	03/31/98 10:31:59
3	host3	myjob	Done	03/31/98 10:31:59
3	host4	myjob	Done	03/31/98 10:31:59

Table 4 pam -n Job Status

Status	Description
Done	Process successfully completed with exit code of 0
Exit (<i>code</i>)	Process unsuccessfully completed with an exit code of <i>code</i>
Signaled (<i>signal</i>)	Process was terminated by <i>signal</i>
Exit (status unknown)	Connection broken; exit status unknown
Killed by PAM (<i>signal</i>)	PAM shutdown process using <i>signal</i>
Unreachable	PAM is unable to reach host after broken connection. No way to determine the state of the process
Runaway	Process is still running; cannot be killed by PAM
Suspend	Process suspended
Undefined	???
Run	Process running
Local RES died	???

Note Use the `-t` option to suppress the process Status report. For example:

```
pam -t -n 4 myjob
```

Getting Host Information

The `lshosts` command is used to display information about LSF host configurations including name, type, model, CPU normalization factor, number of CPUs, total memory, and available resources.

Example:

```
% lshosts
```

HOST_NAME	type	model	cpuf	ncpus	maxmem	maxswp	server	RESOURCES
host1	SGI64	SGI4D35	2.0	1	96M	153M	Yes	(lsf_js irix gla)
host99	SUNSOL	SunSparc	12.0	4	1024M	1930M	Yes	(solaris cs bigmem)
host2	LINUX	I486_33	14.0	1	30M	64M	Yes	(linux)
host7	SUN41	SPARC5LC	3.0	1	15M	29M	Yes	(sparc bsd sun41)
host3	ALPHA~1	DEC5000	5.0	1	88M	384M	Yes	(cs bigmem alpha gla)
host6	ALPHA~1	DEC5000	5.0	1	84M	350M	Yes	(gla)
host4	SUNSOL	SunSparc	12.0	2	256M	733M	Yes	(solaris cs bigmem)
host5	SGI	SGIINDIG	15.0	1	96M	300M	Yes	(irix)
host8	SUNSOL	SunSparc	12.0	1	56M	90M	Yes	(solaris cs bigmem)

4 **Submitting Parallel Applications**

Getting Host Information

A Vendor MPI Implementations

HP MPI

When you use `mpirun` in stand-alone mode, you provide it the host names to be used by the MPI job. To achieve better resource utilization, you can have LSF manage the allocation of hosts, coordinating the start-up phase with `mpirun`. This is done by preceding the regular HP MPI `mpirun` command with:

```
% bsub pam -mpi
```

Example: To run a single-host job and have the LSF Batch system select the host, the command:

```
% mpirun -np 14 a.out
```

is entered as:

```
% bsub pam -mpi mpirun -np 14 a.out
```

Example: To run a multi-host job and have the LSF Batch system select the hosts, the command:

```
% mpirun -f appfile
```

is entered as:

```
% bsub pam -mpi mpirun -f appfile
```

where `appfile` contains the following entries:

```
-h foo -np 8 a.out
-h bar -np 4 b.out
-h foo -np 2 c.out
```

In this example, the hosts `foo` and `bar` are treated as symbolic names and refer to the actual hosts that the LSF Batch system allocates to the job. The `a.out` and `c.out` processes are guaranteed to run on the same host. The `b.out` processes may run on a different host, depending on the resources available and the LSF Batch system scheduling algorithms.

For a complete list of `mpirun` options and environment variable controls, refer to the `mpirun` man page and the *HP MPI User's Guide* version 1.4.

SGI MPI

The `-mpi` argument on the `bsub` and `pam` command-line is a replacement for `mpirun` in the HP environment. Everything after `-mpi` shall be exactly as it would normally appear if `mpirun` were being used.

Example: To run a the `a.out` job and have the LSF Batch system select the host, the command:

```
% mpirun -np 4 a.out
```

is entered as:

```
% mpirun pam -mpi -np 4 a.out
```

Example: To run a multihost job and have the LSF Batch system select the hosts, the following command:

```
% mpirun -f appfile
```

is entered as:

```
% bsub pam -mpi -f appfile
```

where `appfile` contains the following entries:

```
foo -np 4 a.out  
bar -np 4 b.out  
foo -np 2 c.out
```

For a complete list of `mpirun` options and environment variable controls refer to the `mpirun` man page.

SUN HPC MPI

When running LSF Batch jobs on Sun platforms, you can include the Sun-specific argument `-sunhpc` on the `bsub` command line, after any other `bsub` arguments. The following arguments to `-sunhpc` provide additional control over `bsub` behavior in a Sun HPC environment.

`-n processes`

Specify the number of processes to run. Note that the `bsub -n` argument specifies the number of CPUs to be used for the job.

Example: To start a 48-process interactive job on PAM-enabled queue `hpc` that will wrap over at least 4, and as many as 16, CPUs:

```
% bsub -I -n 4,16 -q hpc -sunhpc -n 48 jobname
```

Note Setting the minimum number of CPUs to a number greater than 1 raises the possibility that, if there are fewer CPUs available than the minimum number you specify, the job may fail to start. In this example, if fewer than 4 CPUs are available, the job will not start. You can avoid this potential problem by setting the minimum number of CPUs to 1. However, this introduces the potential cost to performance of having the processes wrapped over a smaller number of CPUs.

`-P host:port`

Specify the PAM address of another job with which the new job should colocate. The PAM address is the TCP socket used for communications between the job and PAM.

Example: To start a 4-CPU interactive job on PAM-enabled queue `hpc`:

```
% bsub -I -n 4 -q hpc -sunhpc -P Athos:123 jobname
```

The new job is colocated with the job whose PAM is running on host `Athos`, using port 123.

`-j job_ID`

Specify the job ID of another job with which the new job should colocate.

`-J job_name`

Specify the job name of another job with which the new job should colocate.

-s

Specify that the job is to be spawned in the STOPPED state.

To identify processes in the STOPPED state, issue the `ps` command with the `-el` argument:

```
orpheus 215 => ps -el
F S  UID  PID  PPID  C  PRI  NI     ADDR     SZ  WCHAN  TTY  TIME CMD
19 T   0    0    0    0  0    SY  f0274e38    0             ?    0:00 sched
```

Here, the `sched` command is in STOPPED state, as indicated by the T entry in the S (State) column.

Note that, when spawning a process in the STOPPED state under LSF, the name of your program will not appear in the `ps` output. Instead, the stopped process will be identified as a RES daemon.

Example: To start a 1-CPU interactive job on PAM-enabled queue `hpc`, in the STOPPED state:

```
% bsub -I -n 1 -q hpc -sunhpc -s jobname
```


Index

Symbols

%a notation 18

B

batch job

 interactive execution 35

 monitor 30

 resource usage 30

 resume 29

 submit 26

 suspend 28

 terminate 31

batch job state 23

batch job status 23

bjobs 30

bkill 31

bresume 29

bstop 28

bsub 26

 pam option 26

C

C program

 compile 15

command syntax xii

compile 15

 C program 15

 Fortran 77 program 15

D

DONE 24

E

execution host substitution 18

EXIT 25

F

Fortran 77 program

 compile 15

H

host type substitution 18

HP MPI 42

I

interactive execution

 batch job 35

J

job

 interactive execution 35

 monitor 30

 resource usage 30

 resume 29

 submit 26

 suspend 28

 terminate 31

job state 23

 DONE 24

 EXIT 25

PEND	24
PSUSP	24
RUN	24
SSUSP	25
USUSP	24
job status	23

L

link	15
C program	15
Fortran 77 program	15
lshosts command	39

M

monitor	
batch job	30
MPI	
HP	42
SGI	43
SUN HPC	44
mpicc	15
mpif77	15

N

notation	
%a	18

P

pam	
bsub option	26
command	35

pam option	
%a	18
PEND	24
PSUSP	24

R

resource usage	
batch job	30
resume	
batch job	29
RUN	24

S

SGI MPI	43
SSUSP	25
submit	
batch job	26
substitution	
host type	18
SUN HPC MPI	44
suspend	
batch job	28
syntax	xii

T

terminate	
batch job	31
typographic conventions	xi

U

USUSP	24
-------	----