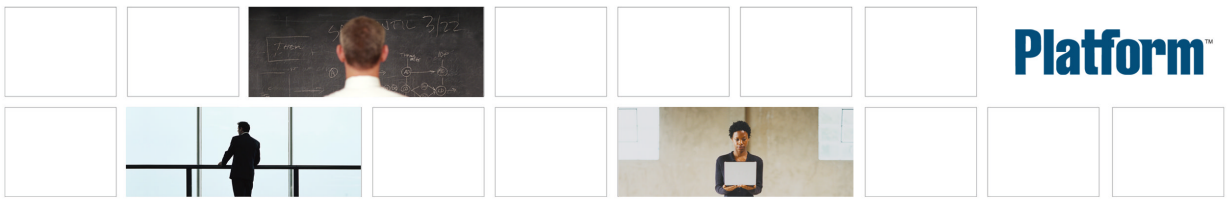
Platform LSF Command Reference

Platform LSF™
Version 7.0 Update 6
Release date: July 2009
Last modified: July 17, 2009



Copyright

© 1994-2009 Platform Computing Inc.

Although the information in this document has been carefully reviewed, Platform Computing Corporation (“Platform”) does not warrant it to be free of errors or omissions. Platform reserves the right to make corrections, updates, revisions or changes to the information in this document.

UNLESS OTHERWISE EXPRESSLY STATED BY PLATFORM, THE PROGRAM DESCRIBED IN THIS DOCUMENT IS PROVIDED “AS IS” AND 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 COMPUTING BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION ANY LOST PROFITS, DATA, OR SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

We’d like to hear from you

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to doc@platform.com.

Your comments should pertain only to Platform documentation. For product support, contact support@platform.com.

Document redistribution and translation

This document is protected by copyright and you may not redistribute or translate it into another language, in part or in whole.

Internal redistribution

You may only redistribute this document internally within your organization (for example, on an intranet) provided that you continue to check the Platform Web site for updates and update your version of the documentation. You may not make it available to your organization over the Internet.

Trademarks

LSF is a registered trademark of Platform Computing Corporation in the United States and in other jurisdictions.

ACCELERATING INTELLIGENCE, PLATFORM COMPUTING, PLATFORM SYMPHONY, PLATFORM JOBSCHEDULER, PLATFORM ENTERPRISE GRID ORCHESTRATOR, PLATFORM EGO, and the PLATFORM and PLATFORM LSF logos are trademarks of Platform Computing Corporation in the United States and in other jurisdictions.

UNIX is a registered trademark of The Open Group in the United States and in other jurisdictions.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Third-party license agreements

http://www.platform.com/Company/third.part.license.htm

Contents

bacct 7
badmin 20
bapp 34
bbot 39
bchkpnt 41
bclusters 43
bgadd 46
bgdel 48
bgmod 50
bhist 52
bhosts 60
bhpart 68
bjdepinfo 70
bjgroup 72
bjobs 76
bkill 91
bladmin 96
blaunch 100
blcollect 102
blhosts 104
blimits 105
blinfo 109
blkill 116
blparams 117
blstat 120
bltasks 126
blusers 129
bmgroup 132
bmig 134
bmod 136
bparams 143
bpeek 144
bpost 146
bqueues 148
bread 163
brequeue 165

bresize 167
bresources 169
brestart 170
bresume 172
brlainfo 174
brsvadd 176
brsvdel 181
brsvmod 182
brsvs 188
brun 190
bsla 192
bslots 196
bstatus 198
bstop 200
bsub 203
bswitch 241
btop 243
bugroup 245
busers 246
ch 248
lsacct 251
lsacctmrg 255
lsadmin 256
lsclusters 265
lseligible 267
lsfinstall 269
lsfmon 272
lsfrestart 273
lsfshutdown 274
lsfstartup 275
lsgrun 276
lshosts 279
lsid 284
lsinfo 285
lsload 287
lsloadadj 292
lslogin 294
lsltasks 296
lsmon 298
lspasswd 302
lsplace 303
lsrcp 305
lsrtasks 308
lsrun 310

lstcsh 313
pam 318
patchinstall 322
perfadmin 324
perfremoverc.sh 326
perfsetrc.sh 327
pmcadmin 328
pmcremoverc.sh 330
pmcsetrc.sh 331
pversions (UNIX) 332
pversions (Windows) 336
ssacct 337
ssched 341
taskman 345
tspeek 346
tssub 347
wgpasswd 349
wguser 350

bacct

Displays accounting statistics about finished jobs.

Synopsis

bacct [**-b** | **-l**] [**-d**] [**-e**] [**-w**] [**-x**] [**-app** *application_profile_name*] [**-C** *time0,time1*] [**-D** *time0,time1*] [**-f** *logfile_name*] [**-Lp** *ls_project_name ...*] [**-m** *host_name ...*|**-M** *host_list_file*] [**-N** *host_name* | **-N** *host_model* | **-N** *cpu_factor*] [**-P** *project_name ...*] [**-q** *queue_name ...*] [**-sla** *service_class_name ...*] [**-S** *time0,time1*] [**-u** *user_name ...* | **-u all**]

bacct [**-b** | **-l**] [**-f** *logfile_name*] [*job_ID ...*]

bacct [**-U** *reservation_ID ...* | **-U all** [**-u** *user_name ...* | **-u all**]

bacct [**-h** | **-V**]

Description

Displays a summary of accounting statistics for all finished jobs (with a DONE or EXIT status) submitted by the user who invoked the command, on all hosts, projects, and queues in the LSF system. `bacct` displays statistics for all jobs logged in the current LSF accounting log file: `LSB_SHAREDIR/cluster_name/logdir/lsb.acct`.

CPU time is not normalized.

All times are in seconds.

Statistics not reported by `bacct` but of interest to individual system administrators can be generated by directly using `awk` or `perl` to process the `lsb.acct` file.

Throughput calculation

The throughput (T) of the LSF system, certain hosts, or certain queues is calculated by the formula:

```
T = N/(ET-BT)
```

where:

- N is the total number of jobs for which accounting statistics are reported
- BT is the Start time : when the first job was logged
- ET is the End time: when the last job was logged

You can use the option `-C` *time0*`,` *time1* to specify the Start time as time0 and the End time as time1. In this way, you can examine throughput during a specific time period.

Jobs involved in the throughput calculation are only those being logged (that is, with a DONE or EXIT status). Jobs that are running, suspended, or that have never been dispatched after submission are not considered, because they are still in the LSF system and not logged in `lsb.acct`.

The total throughput of the LSF system can be calculated by specifying `-u all` without any of the `-m`, `-q`, `-S`, `-D` or *job_ID* options. The throughput of certain hosts can be calculated by specifying `-u all` without the `-q`, `-S`, `-D` or *job_ID* options. The throughput of certain queues can be calculated by specifying `-u all` without the `-m`, `-S`, `-D` or job_ID options.

`bacct` does not show local pending batch jobs killed using `bkill -b`. `bacct` shows MultiCluster jobs and local running jobs even if they are killed using `bkill -b`.

Options

-b

Brief format.

-d

Displays accounting statistics for successfully completed jobs (with a DONE status).

-e

Displays accounting statistics for exited jobs (with an EXIT status).

-l

Long format. Displays detailed information for each job in a multiline format.

If the job was submitted with `bsub -K`, the -l option displays `Synchronous Execution`.

-w

Wide field format.

-x

Displays jobs that have triggered a job exception (overrun, underrun, idle, runtime_est_exceeded). Use with the `-l` option to show the exception status for individual jobs.

-app *application_profile_name*

Displays accounting information about jobs submitted to the specified application profile. You must specify an existing application profile configured in `lsb.applications`.

-C *time0*,*time1*

Displays accounting statistics for jobs that completed or exited during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist`.

-D *time0*,*time1*

Displays accounting statistics for jobs dispatched during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist`.

-f *logfile_name*

Searches the specified job log file for accounting statistics. Specify either an absolute or relative path.

Useful for offline analysis.

The specified file path can contain up to 4094 characters for UNIX, or up to 512 characters for Windows.

-Lp *ls_project_name* ...

Displays accounting statistics for jobs belonging to the specified License Scheduler projects. If a list of projects is specified, project names must be separated by spaces and enclosed in quotation marks (") or (').

-M *host_list_file*

Displays accounting statistics for jobs dispatched to the hosts listed in a file (*host_list_file*) containing a list of hosts. The host list file has the following format:

- Multiple lines are supported
- Each line includes a list of hosts separated by spaces
- The length of each line must be less than 512 characters

-m *host_name* ...

Displays accounting statistics for jobs dispatched to the specified hosts.

If a list of hosts is specified, host names must be separated by spaces and enclosed in quotation marks (") or (').

-N *host_name* | -N *host_model* | -N *cpu_factor*

Normalizes CPU time by the CPU factor of the specified host or host model, or by the specified CPU factor.

If you use `bacct` offline by indicating a job log file, you must specify a CPU factor.

-P *project_name* ...

Displays accounting statistics for jobs belonging to the specified projects. If a list of projects is specified, project names must be separated by spaces and enclosed in quotation marks (") or ('). You cannot use one double quote and one single quote to enclose the list.

-q *queue_name* ...

Displays accounting statistics for jobs submitted to the specified queues.

If a list of queues is specified, queue names must be separated by spaces and enclosed in quotation marks (") or (').

-S *time0*,*time1*

Displays accounting statistics for jobs submitted during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist`.

-sla *service_class_name*

Displays accounting statistics for jobs that ran under the specified service class.

If a default system service class is configured with ENABLE_DEFAULT_EGO_SLA in `lsb.params` but not explicitly configured in `lsb.applications`, `bacct -sla` *service_class_name* displays accounting information for the specified default service class.

-U *reservation_id* ... | -U all

Displays accounting statistics for the specified advance reservation IDs, or for all reservation IDs if the keyword all is specified.

A list of reservation IDs must be separated by spaces and enclosed in quotation marks (") or (’).

The -U option also displays historical information about reservation modifications.

When combined with the `-U` option, `-u` is interpreted as the user name of the reservation creator. For example:

```
bacct -U all -u user2
```

shows all the advance reservations created by user `user2`.

Without the `-u` option, `bacct -U` shows all advance reservation information about jobs submitted by the user.

In a MultiCluster environment, advance reservation information is only logged in the execution cluster, so `bacct` displays advance reservation information for local reservations only. You cannot see information about remote reservations. You cannot specify a remote reservation ID, and the keyword all only displays information about reservations in the local cluster.

-u *user_name* ...|-u all

Displays accounting statistics for jobs submitted by the specified users, or by all users if the keyword all is specified.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or (’). You can specify both user names and user IDs in the list of users.

***job_ID* ...**

Displays accounting statistics for jobs with the specified job IDs.

If the reserved job ID 0 is used, it is ignored.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Default output format (SUMMARY)

Statistics on jobs. The following fields are displayed:

- Total number of done jobs
- Total number of exited jobs
- Total CPU time consumed
- Average CPU time consumed
- Maximum CPU time of a job
- Minimum CPU time of a job
- Total wait time in queues

- Average wait time in queue
- Maximum wait time in queue
- Minimum wait time in queue
- Average turnaround time (seconds/job)
- Maximum turnaround time
- Minimum turnaround time
- Average hog factor of a job (cpu time/turnaround time)
- Maximum hog factor of a job
- Minimum hog factor of a job
- Total throughput
- Beginning time: the completion or exit time of the first job selected
- Ending time: the completion or exit time of the last job selected

The total, average, minimum, and maximum statistics are on all specified jobs.

The wait time is the elapsed time from job submission to job dispatch.

The turnaround time is the elapsed time from job submission to job completion.

The hog factor is the amount of CPU time consumed by a job divided by its turnaround time.

The throughput is the number of completed jobs divided by the time period to finish these jobs (jobs/hour).

Brief format (-b)

In addition to the default format SUMMARY, displays the following fields:

U/UID

Name of the user who submitted the job. If LSF fails to get the user name by `getpwuid`, the user ID is displayed.

QUEUE

Queue to which the job was submitted.

SUBMIT_TIME

Time when the job was submitted.

CPU_T

CPU time consumed by the job.

WAIT

Wait time of the job.

TURNAROUND

Turnaround time of the job.

FROM

Host from which the job was submitted.

EXEC_ON

Host or hosts to which the job was dispatched to run.

JOB_NAME

The job name assigned by the user, or the command string assigned by default at job submission with `bsub`. If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

The displayed job name or job command can contain up to 4094 characters for UNIX, or up to 512 characters for Windows.

Long format (-l)

In addition to the fields displayed by default in SUMMARY and by `-b`, displays the following fields:

JOBID

Identifier that LSF assigned to the job.

PROJECT_NAME

Project name assigned to the job.

STATUS

Status that indicates the job was either successfully completed (DONE) or exited (EXIT).

DISPAT_TIME

Time when the job was dispatched to run on the execution hosts.

COMPL_TIME

Time when the job exited or completed.

HOG_FACTOR

Average hog factor, equal to "CPU time" / "turnaround time".

MEM

Maximum resident memory usage of all processes in a job. By default, memory usage is shown in MB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

CWD

Current working directory of the job.

SWAP

Maximum virtual memory usage of all processes in a job. By default, swap space is shown in MB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

INPUT_FILE

File from which the job reads its standard input (see `bsub -i` *input_file*).

OUTPUT_FILE

File to which the job writes its standard output (see `bsub -o` *output_file*).

ERR_FILE

File in which the job stores its standard error output (see `bsub -e` *err_file*).

EXCEPTION STATUS

Possible values for the exception status of a job include:

idle

The job is consuming less CPU time than expected. The job idle factor (CPU time/ runtime) is less than the configured JOB_IDLE threshold for the queue and a job exception has been triggered.

overrun

The job is running longer than the number of minutes specified by the JOB_OVERRUN threshold for the queue and a job exception has been triggered.

underrun

The job finished sooner than the number of minutes specified by the JOB_UNDERRUN threshold for the queue and a job exception has been triggered.

runtime_est_exceeded

The job is running longer than the number of minutes specified by the runtime estimation and a job exception has been triggered.

SYNCHRONOUS_EXECUTION

Job was submitted with the `-K` option. LSF submits the job and waits for the job to complete.

Advance reservations (-U)

Displays the following fields:

RSVID

Advance reservation ID assigned by `brsvadd` command

TYPE

Type of reservation: user or system

CREATOR

User name of the advance reservation creator, who submitted the `brsvadd` command

USER

User name of the advance reservation user, who submitted the job with `bsub -U`

NCPUS

Number of CPUs reserved

RSV_HOSTS

List of hosts for which processors are reserved, and the number of processors reserved

TIME_WINDOW

Time window for the reservation.

- A one-time reservation displays fields separated by slashes (`month/day/hour/minute`). For example:

  ```
  11/12/14/0-11/12/18/0
  ```

- A recurring reservation displays fields separated by colons (`day:hour:minute`). For example:

  ```
  5:18:0 5:20:0
  ```

Termination reasons displayed by bacct

When LSF detects that a job is terminated, `bacct -l` displays one of the following termination reasons. The corresponding integer value logged to the JOB_FINISH record in `lsb.acct` is given in parentheses.

- TERM_ADMIN: Job killed by root or LSF administrator (15)
- TERM_BUCKET_KILL: Job killed with `bkill -b` (23)
- TERM_CHKPNT: Job killed after checkpointing (13)
- TERM_CWD_NOTEXIST: current working directory is not accessible or does not exist on the execution host (25)
- TERM_CPULIMIT: Job killed after reaching LSF CPU usage limit (12)
- TERM_DEADLINE: Job killed after deadline expires (6)
- TERM_EXTERNAL_SIGNAL: Job killed by a signal external to LSF (17)
- TERM_FORCE_ADMIN: Job killed by root or LSF administrator without time for cleanup (9)
- TERM_FORCE_OWNER: Job killed by owner without time for cleanup (8)
- TERM_LOAD: Job killed after load exceeds threshold (3)
- TERM_MEMLIMIT: Job killed after reaching LSF memory usage limit (16)
- TERM_OWNER: Job killed by owner (14)
- TERM_PREEMPT: Job killed after preemption (1)
- TERM_PROCESSLIMIT: Job killed after reaching LSF process limit (7)
- TERM_REQUEUE_ADMIN: Job killed and requeued by root or LSF administrator (11)
- TERM_REQUEUE_OWNER: Job killed and requeued by owner (10)
- TERM_RUNLIMIT: Job killed after reaching LSF run time limit (5)
- TERM_SLURM: Job terminated abnormally in SLURM (node failure) (22)
- TERM_SWAP: Job killed after reaching LSF swap usage limit (20)
- TERM_THREADLIMIT: Job killed after reaching LSF thread limit (21)
- TERM_UNKNOWN: LSF cannot determine a termination reason—0 is logged but TERM_UNKNOWN is not displayed (0)
- TERM_WINDOW: Job killed after queue run window closed (2)
- TERM_ZOMBIE: Job exited while LSF is not available (19)

Tip:

The integer values logged to the JOB_FINISH record in `lsb.acct` and termination reason keywords are mapped in `lsbatch.h`.

Example: Default format

bacct

```
Accounting information about jobs that are:
  - submitted by users user1.
  - accounted on all projects.
  - completed normally or exited.
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done jobs:      60      Total number of exited jobs:   118
 Total CPU time consumed:    1011.5      Average CPU time consumed:     5.7
 Maximum CPU time of a job:   991.4      Minimum CPU time of a job:     0.0
 Total wait time in queues: 134598.0
 Average wait time in queue:  756.2
 Maximum wait time in queue: 7069.0      Minimum wait time in queue:    0.0
 Average turnaround time:      3585 (seconds/job)
 Maximum turnaround time:     77524      Minimum turnaround time:         6
 Average hog factor of a job:  0.00 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.56      Minimum hog factor of a job:  0.00
 Total throughput:             0.67 (jobs/hour)  during  266.18 hours
 Beginning time:       Aug  8 15:48      Ending time:          Aug 19 17:59
```

Example: Jobs that have triggered job exceptions

bacct -x -l

```
Accounting information about jobs that are:
  - submitted by users user1,
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------

Job <1743>, User <user1>, Project <default>, Status <DONE>, Queue <normal>,
Command<sleep 30>
Mon Aug 11 18:16:17: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/
null>;
Mon Aug 11 18:17:22: Dispatched to <hostC>;
Mon Aug 11 18:18:54: Completed <done>.

 EXCEPTION STATUS:  underrun

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.19       65            157     done         0.0012     4M      5M
------------------------------------------------------------------------------

Job <1948>, User <user1>, Project <default>, Status <DONE>, Queue <normal>,Command <sleep
550>
Tue Aug 12 14:15:03: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/
null>;
Tue Aug 12 14:15:15: Dispatched to <hostC>;
Tue Aug 12 14:25:08: Completed <done>.

 EXCEPTION STATUS:  overrun  idle

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.20       12            605     done         0.0003     4M      5M
------------------------------------------------------------------------------

Job <1949>, User <user1>, Project <default>, Status <DONE>, Queue <normal>,Command <sleep
400>
Tue Aug 12 14:26:11: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/
null>;
Tue Aug 12 14:26:18: Dispatched to <hostC>;
Tue Aug 12 14:33:16: Completed <done>.

 EXCEPTION STATUS:  idle

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.17        7            425     done         0.0004     4M      5M
Job <719[14]>, Job Name <test[14]>, User <user1>, Project <default>, Status <EXIT>, Queue
```

```
<normal>, Command </home/user1/job1>
Mon Aug 18 20:27:44: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/
null>;
Mon Aug 18 20:31:16: [14] dispatched to <hostA>;
Mon Aug 18 20:31:18: Completed <exit>.

 EXCEPTION STATUS:  underrun

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.19      212            214     exit         0.0009     2M      4M
------------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done jobs:      45      Total number of exited jobs:    56
 Total CPU time consumed:    1009.1      Average CPU time consumed:    10.0
 Maximum CPU time of a job:   991.4      Minimum CPU time of a job:     0.1
 Total wait time in queues: 116864.0
 Average wait time in queue: 1157.1
 Maximum wait time in queue: 7069.0      Minimum wait time in queue:    7.0
 Average turnaround time:      1317 (seconds/job)
 Maximum turnaround time:      7070      Minimum turnaround time:        10
 Average hog factor of a job:  0.01 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.56      Minimum hog factor of a job:  0.00
 Total throughput:             0.59 (jobs/hour)  during  170.21 hours
 Beginning time:       Aug 11 18:18      Ending time:          Aug 18 20:31
```

Example: Advance reservation accounting information

bacct -U user1#2

```
Accounting for:
  - advanced reservation IDs: user1#2
  - advanced reservations created by user1
------------------------------------------------------------------------------
RSVID       TYPE      CREATOR    USER    NCPUS       RSV_HOSTS     TIME_WINDOW
user1#2     user        user1   user1      1           hostA:1    9/16/17/36-9/16/17/38
SUMMARY:
Total number of jobs:               4
Total CPU time consumed:      0.5 second
Maximum memory of a job:     4.2 MB
Maximum swap of a job:         5.2 MB
Total duration time:                0 hour    2 minute    0 second
```

Example: LSF job termination reason logging

When a job finishes, LSF reports the last job termination action it took against the job and logs it into `lsb.acct`.

If a running job exits because of node failure, LSF sets the correct exit information in `lsb.acct`, `lsb.events`, and the job output file.

Use `bacct -l` to view job exit information logged to `lsb.acct`:

```
bacct -l 7265

Accounting information about jobs that are:
  - submitted by all users.
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------

Job <7265>, User <lsfadmin>, Project <default>, Status <EXIT>, Queue <normal>, Command
<srun sleep 100000>
Thu Sep 16 15:22:09: Submitted from host <hostA>, CWD <$HOME>;
Thu Sep 16 15:22:20: Dispatched to 4 Hosts/Processors <4*hostA>;
Thu Sep 16 15:22:20: slurm_id=21793;ncpus=4;slurm_alloc=n[13-14];
Thu Sep 16 15:23:21: Completed <exit>; TERM_RUNLIMIT: job killed after reaching LSF run time limit.

Accounting information about this job:
     Share group charged </lsfadmin>
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.04       11             72     exit         0.0006     0K      0K
------------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done jobs:       0      Total number of exited jobs:     1
 Total CPU time consumed:       0.0      Average CPU time consumed:     0.0
 Maximum CPU time of a job:     0.0      Minimum CPU time of a job:     0.0
 Total wait time in queues:    11.0
 Average wait time in queue:   11.0
 Maximum wait time in queue:   11.0      Minimum wait time in queue:   11.0
 Average turnaround time:        72 (seconds/job)
 Maximum turnaround time:        72      Minimum turnaround time:        72
 Average hog factor of a job:  0.00 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.00      Minimum hog factor of a job:  0.00
```

Example: Resizable job information

Use `bacct -l` to view resizable job information logged to `lsb.acct`:

- The autoresizable attribute of a job and the resize notification command if `bsub -ar` and `bsub -rnc` *resize_notification_command* are specified.
- Job allocation changes whenever a JOB_RESIZE event is logged to `lsb.acct`.

When an allocation grows, `bacct` shows:

```
Additional allocation on num_hosts Hosts/Processors host_list
```

When an allocation shrinks, `bacct` shows

```
Release allocation on num_hosts Hosts/Processors host_list
```

For example, assume, a job submitted as

```
bsub -n 1, 5 -ar myjob
```

and the initial allocation is on hostA and hostB. The first resize request is allocated on hostC and hostD. A second resize request is allocated on hostE. `bacct -l` displays:

```
bacct -l 205

Accounting information about jobs that are:
  - submitted by all users.
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------

Job <1150>, User <user2>, Project <default>, Status <DONE>, Queue <normal>,
Command <sleep 10>
Mon Jun  2 11:42:00: Submitted from host <hostA>, CWD <$HOME>;
Mon Jun  2 11:43:00: Dispatched to 2 Hosts/Processors <hostA> <hostB>;
Mon Jun  2 11:43:52: Additional allocation on 2 Hosts/Processors <hostC> <hostD>;
Mon Jun  2 11:44:55: Additional allocation on <hostE>;
Mon Jun  2 11:51:40: Completed <done>.
...
```

Files

Reads `lsb.acct`, `lsb.acct.`*n*.

See also

bhist, bsub, bjobs, `lsb.acct`, brsvadd, brsvs, bsla, `lsb.serviceclasses`

badmin

Administrative tool for LSF.

Synopsis

badmin *subcommand*

badmin [**-h** | **-V**]

Description

Important:

This command can only be used by LSF administrators.

badmin provides a set of subcommands to control and monitor LSF. If no subcommands are supplied for badmin, badmin prompts for a subcommand from standard input.

Information about each subcommand is available through the help command.

The badmin subcommands include privileged and non-privileged subcommands. Privileged subcommands can only be invoked by root or LSF administrators. Privileged subcommands are:

`reconfig`

`mbdrestart`

`qopen`

`qclose`

`qact`

`qinact`

`hopen`

`hclose`

`hrestart`

`hshutdown`

`hstartup`

`hghostadd`

`hghostdel`

`diagnose`

The configuration file `lsf.sudoers`(5) must be set to use the privileged command `hstartup` by a non-root user.

All other commands are non-privileged commands and can be invoked by any LSF user. If the privileged commands are to be executed by the LSF administrator, `badmin` must be installed, because it needs to send the request using a privileged port.

For subcommands for which multiple hosts can be specified, do not enclose the host names in quotation marks.

Subcommand synopsis

ckconfig [**-v**]

diagnose [*job_ID* ... | **"***job_ID***[***index***]"** ...]

reconfig [**-v**] [**-f**]

mbdrestart [**-C** *comment*] [**-v**] [**-f**]

qopen [**-C** *comment*] [*queue_name* ... | **all**]

qclose [**-C** *comment*] [*queue_name* ... | **all**]

qact [**-C** *comment*] [*queue_name* ... | **all**]

qinact [**-C** *comment*] [*queue_name* ... | **all**]

qhist [**-t** *time0*,*time1*] [**-f** *logfile_name*] [*queue_name* ...]

hopen [**-C** *comment*] [*host_name* ... | *host_group* ... | *compute_unit* ... | **all**]

hclose [**-C** *comment*] [*host_name* ... | *host_group* ... | *compute_unit* ... | **all**]

hrestart [**-f**] [*host_name* ... | **all**]

hshutdown [**-f**] [*host_name* ... | **all**]

hstartup [**-f**] [*host_name* ... | **all**]

hhist [**-t** *time0*,*time1*] [**-f** *logfile_name*] [*host_name* ...]

mbdhist [**-t** *time0*,*time1*] [**-f** *logfile_name*]

hist [**-t** *time0*,*time1*] [**-f** *logfile_name*]

hghostadd [**-C** *comment*] *host_group* | *compute_unit* | *host_name* [*host_name* ...]

hghostdel [**-f**] [**-C** *comment*] *host_group* | *compute_unit* | *host_name* [*host_name* ...]

help [*command* ...] | **?** [*command* ...]

quit

mbddebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**]

mbdtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**]

sbddebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [*host_name* ...]

sbdtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [*host_name* ...]

schddebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**]

schdtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**]

showconf mbd | [**sbd** [*host_name* … | **all**]]

perfmon start [*sample_period*]| **stop** | **view** | **setperiod** *sample_period*

-h

-V

Options

subcommand

Executes the specified subcommand. See Usage section.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Usage

ckconfig [-v]

Checks LSF configuration files located in the `LSB_CONFDIR/cluster_name/configdir` directory, and checks `LSF_ENVDIR/lsf.licensescheduler`.

The LSB_CONFDIR variable is defined in `lsf.conf` (see `lsf.conf`(5)), which is in LSF_ENVDIR or `/etc` (if LSF_ENVDIR is not defined).

By default, `badmin ckconfig` displays only the result of the configuration file check. If warning errors are found, `badmin` prompts you to display detailed messages.

-v

Verbose mode. Displays detailed messages about configuration file checking to `stderr`.

diagnose [*job_ID* ... | "*job_ID*]" ...][

Displays full pending reason list if CONDENSE_PENDING_REASONS=Y is set in `lsb.params`. For example:

```
badmin diagnose 1057
```

reconfig [-v] [-f]

Dynamically reconfigures LSF.

Configuration files are checked for errors and the results displayed to `stderr`. If no errors are found in the configuration files, a reconfiguration request is sent to `mbatchd` and configuration files are reloaded.

With this option, `mbatchd` is not restarted and `lsb.events` is not replayed. To restart `mbatchd` and replay `lsb.events`, use `badmin mbdrestart`.

When you issue this command, `mbatchd` is available to service requests while reconfiguration files are reloaded. Configuration changes made since system boot or the last reconfiguration take effect.

If warning errors are found, `badmin` prompts you to display detailed messages. If fatal errors are found, reconfiguration is not performed, and `badmin` exits.

If you add a host to a queue or to a host group or compute unit, the new host is not recognized by jobs that were submitted before you reconfigured. If you want the new host to be recognized, you must use the command `badmin mbdrestart`.

Resource requirements determined by the queue no longer apply to a running job after running `badmin reconfig`, For example, if you change the RES_REQ parameter in a queue and reconfigure the cluster, the previous queue-level resource requirements for running jobs are lost.

-v

Verbose mode. Displays detailed messages about the status of the configuration files. Without this option, the default is to display the results of configuration file checking. All messages from the configuration file check are printed to `stderr`.

-f

Disables interaction and proceeds with reconfiguration if configuration files contain no fatal errors.

mbdrestart [-C *comment*] [-v] [-f]

Dynamically reconfigures LSF and restarts `mbatchd` and `mbschd`.

Configuration files are checked for errors and the results printed to `stderr`. If no errors are found, configuration files are reloaded, `mbatchd` and `mbschd` are restarted, and events in `lsb.events` are replayed to recover the running state of the last `mbatchd`. While `mbatchd` restarts, it is unavailable to service requests.

If warning errors are found, `badmin` prompts you to display detailed messages. If fatal errors are found, `mbatchd` and `mbschd` restart is not performed, and `badmin` exits.

If `lsb.events` is large, or many jobs are running, restarting `mbatchd` can take several minutes. If you only need to reload the configuration files, use `badmin reconfig`.

-C comment

Logs the text of comment as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

-v

Verbose mode. Displays detailed messages about the status of configuration files. All messages from configuration checking are printed to `stderr`.

-f

Disables interaction and forces reconfiguration and `mbatchd` restart to proceed if configuration files contain no fatal errors.

qopen [-C *comment*] [*queue_name* ... | all]

Opens specified queues, or all queues if the reserved word all is specified. If no queue is specified, the system default queue is assumed. A queue can accept batch jobs only if it is open.

-C *comment*

Logs the text of comment as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

qclose [-C *comment*] [*queue_name* ... | all]

Closes specified queues, or all queues if the reserved word all is specified. If no queue is specified, the system default queue is assumed. A queue does not accept any job if it is closed.

-C *comment*

Logs the text as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

qact [-C *comment*] [*queue_name* ... | all]

Activates specified queues, or all queues if the reserved word all is specified. If no queue is specified, the system default queue is assumed. Jobs in a queue can be dispatched if the queue is activated.

A queue inactivated by its run windows cannot be reactivated by this command.

-C *comment*

Logs the text of the comment as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

qinact [-C *comment*] [*queue_name* ... | all]

Inactivates specified queues, or all queues if the reserved word all is specified. If no queue is specified, the system default queue is assumed. No job in a queue can be dispatched if the queue is inactivated.

-C *comment*

Logs the text as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

qhist [-t *time0*,*time1*] [-f *logfile_name*] [*queue_name* ...]

Displays historical events for specified queues, or for all queues if no queue is specified. Queue events are queue opening, closing, activating and inactivating.

-t *time0*,*time1*

Displays only those events that occurred during the period from *time0* to *time1*. See `bhist(1)` for the time format. The default is to display all queue events in the event log file.

-f *logfile_name*

Specify the file name of the event log file. Either an absolute or a relative path name may be specified. The default is to use the event log file currently used by the LSF system: `LSB_SHAREDIR/`*cluster_name*`/logdir/lsb.events`. Option -f is useful for offline analysis.

If you specified an administrator comment with the `-C` option of the queue control commands `qclose`, `qopen`, `qact`, and `qinact`, `qhist` displays the comment text.

hopen [-C *comment*] [*host_name* ... | *host_group* ... | *compute_unit* ... | all]

Opens batch server hosts. Specify the names of any server hosts, host groups, or compute units. All batch server hosts are opened if the reserved word all is specified. If no host, host group, or compute unit is specified, the local host is assumed. A host accepts batch jobs if it is open.

Important:

If EGO-enabled SLA scheduling is configured through ENABLE_DEFAULT_EGO_SLA in `lsb.params`, and a host is closed by EGO, it cannot be reopened by `badmin hopen`. Hosts closed by EGO have status closed_EGO in `bhosts -l` output.

-C *comment*

Logs the text as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

If you open a host group or compute unit, each member displays with the same comment string.

hclose [-C *comment*] [*host_name* ... | *host_group* ... | *compute_unit* ... | all]

Closes batch server hosts. Specify the names of any server hosts, host groups, or compute units. All batch server hosts are closed if the reserved word all is specified. If no argument is specified, the local host is assumed. A closed host does not accept any new job, but jobs already dispatched to the host are not affected. Note that this is different from a host closed by a window; all jobs on it are suspended in that case.

-C *comment*

Logs the text as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

If you close a host group or compute unit, each member displays with the same comment string.

hghostadd [-C *comment*] *host_group* | *compute_unit* |*host_name* [*host_name* ...]

If dynamic host configuration is enabled, dynamically adds hosts to a host group or compute unit. After receiving the host information from the master LIM, `mbatchd` dynamically adds the host without triggering a `reconfig`.

Once the host is added to the host group or compute unit, it is considered part of that group with respect to scheduling decision making for both newly submitted jobs and for existing pending jobs.

This command fails if any of the specified host groups, compute units, or host names are not valid.

Restriction:

If EGO-enabled SLA scheduling is configured through ENABLE_DEFAULT_EGO_SLA in `lsb.params`, you cannot use `hghostadd` because all host allocation is under control of Platform EGO.

-C *comment*

Logs the text as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

hghostdel [-f] [-C *comment*] *host_group* | *compute_unit* |*host_name* [*host_name* ...]

Dynamically deletes hosts from a host group or compute unit by triggering an `mbatchd reconfig`.

This command fails if any of the specified host groups, compute units, or host names are not valid.

Caution:

If you want to change a dynamic host to a static host, first use the command `badmin hghostdel` to remove the dynamic host from any host group or compute unit that it belongs to, and then configure the host as a static host in `lsf.cluster.`*`cluster_name`*.

Restriction:

If EGO-enabled SLA scheduling is configured through ENABLE_DEFAULT_EGO_SLA in `lsb.params`, you cannot use `hghostdel` because all host allocation is under control of Platform EGO.

-f

Disables interaction and does not ask for confirmation when reconfiguring the mbatchd.

-C *comment*

Logs the text as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

hrestart [-f] [*host_name* ... | all]

Restarts `sbatchd` on the specified hosts, or on all server hosts if the reserved word all is specified. If no host is specified, the local host is assumed. `sbatchd` reruns itself from the beginning. This allows new `sbatchd` binaries to be used.

-f

Specify the name of the file into which timing messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file created has the following format:

logfile_name.daemon_name.`log.`*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Note: Both timing and debug messages are logged in the same files.

Default: current LSF system log file in the LSF system log file directory, in the format *daemon_name.*`log.`*host_name*.

hshutdown [-f] [*host_name* ... | all]

Shuts down `sbatchd` on the specified hosts, or on all batch server hosts if the reserved word all is specified. If no host is specified, the local host is assumed. `sbatchd` exits upon receiving the request.

-f

Disables interaction and does not ask for confirmation for shutting down `sbatchd`.

hstartup [-f] [*host_name* ... | all]

Starts `sbatchd` on the specified hosts, or on all batch server hosts if the reserved word all is specified. Only root and users listed in the file `lsf.sudoers(5)` can use the all and -f options. These users must be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. If no host is specified, the local host is assumed.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

-f

Disables interaction and does not ask for confirmation for starting `sbatchd`.

hhist [-t *time0*,*time1*] [-f *logfile_name*] [*host_name* ...]

Displays historical events for specified hosts, or for all hosts if no host is specified. Host events are host opening and closing.

-t *time0*,*time1*

Displays only those events that occurred during the period from *time0* to *time1*. See `bhist(1)` for the time format. The default is to display all queue events in the event log file.

-f *logfile_name*

Specify the file name of the event log file. Either an absolute or a relative path name may be specified. The default is to use the event log file currently used by the LSF system: `LSB_SHAREDIR/`*cluster_name*`/logdir/lsb.events`. Option -f is useful for offline analysis.

If you specified an administrator comment with the `-C` option of the host control commands `hclose` or `hopen`, `hhist` displays the comment text.

mbdhist [-t *time0*,*time1*] [-f *logfile_name*]

Displays historical events for `mbatchd`. Events describe the starting and exiting of `mbatchd`.

-t *time0*,*time1*

Displays only those events that occurred during the period from *time0* to *time1*. See `bhist(1)` for the time format. The default is to display all queue events in the event log file.

-f *logfile_name*

Specify the file name of the event log file. Either an absolute or a relative path name may be specified. The default is to use the event log file currently used by the LSF

system: `LSB_SHAREDIR/`*cluster_name*`/logdir/lsb.events`. Option -f is useful for offline analysis.

If you specified an administrator comment with the `-C` option of the `mbdrestart` command, `mbdhist` displays the comment text.

hist [-t *time0*,*time1*] [-f *logfile_name*]

Displays historical events for all the queues, hosts and `mbatchd`.

-t *time0*,*time1*

Displays only those events that occurred during the period from *time0* to *time1*. See `bhist(1)` for the time format. The default is to display all queue events in the event log file.

-f *logfile_name*

Specify the file name of the event log file. Either an absolute or a relative path name may be specified. The default is to use the event log file currently used by the LSF system: `LSB_SHAREDIR/`*cluster_name*`/logdir/lsb.events`. Option -f is useful for offline analysis.

If you specified an administrator comment with the -C option of the queue, host, and `mbatchd` commands, `hist` displays the comment text.

help [*command* ...] | ? [*command* ...]

Displays the syntax and functionality of the specified commands.

quit

Exits the `badmin` session.

mbddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o]

Sets message log level for `mbatchd` to include additional information in log files. You must be root or the LSF administrator to use this command.

See `sddebug` for an explanation of options.

mbdtime [-l *timing_level*] [-f *logfile_name*] [-o]

Sets timing level for `mbatchd` to include additional timing information in log files. You must be root or the LSF administrator to use this command.

sbddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o] [*host_name* ...]

Sets the message log level for `sbatchd` to include additional information in log files. You must be root or the LSF administrator to use this command.

In MultiCluster, debug levels can only be set for hosts within the same cluster. For example, you cannot set debug or timing levels from a host in clusterA for a host in clusterB. You need to be on a host in clusterB to set up debug or timing levels for clusterB hosts.

If the command is used without any options, the following default values are used:

class_name=0 (no additional classes are logged)

debug_level=0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

logfile_name=current LSF system log file in the LSF system log file directory, in the format *daemon_name*.`log`.*host_name*

host_name=local host (host from which command was submitted)

-c *class_name* ...

Specifies software classes for which debug messages are to be logged.

Format of *class_name* is the name of a class, or a list of class names separated by spaces and enclosed in quotation marks. Classes are also listed in `lsf.h`.

Valid log classes are:

- LC_ADVRSV - Log advance reservation modifications
- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - Log checkpointing messages
- LC_COMM - Log communication messages
- LC_CONF - Print out all parameters in `lsb.params`
- LC_DCE - Log messages pertaining to DCE support
- LC_EEVENTD - Log eeventd messages
- LC_ELIM - Log ELIM messages
- LC_EXEC - Log significant steps for job execution
- LC_FAIR - Log fairshare policy messages
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_JARRAY - Log job array messages
- LC_JLIMIT - Log job slot limit messages
- LC_LICENSE - Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_LOADINDX - Log load index messages
- LC_M_LOG - Log multievent logging messages
- LC_MPI - Log MPI messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PEND - Log messages related to job pending reasons
- LC_PERFM - Log performance messages
- LC_PIM - Log PIM messages
- LC_PREEMPT - Log preemption policy messages
- LC_RESREQ - Log resource requirement messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_SYS - Log system call messages
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Default: 0 (no additional classes are logged)

-l *debug_level*

Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

Possible values:

0 LOG_DEBUG level in parameter LSF_LOG_MASK in `lsf.conf`.

1 LOG_DEBUG1 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

`2` LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

`3` LOG_DEBUG3 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2, LOG_DEBUG1, and LOG_DEBUG levels.

Default: 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

-f *logfile_name*

Specify the name of the file into which debugging messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file that is created has the following format:

logfile_name.daemon_name.`log`.*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Default: current LSF system log file in the LSF system log file directory.

-o

Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LSF_LOG_MASK and classes are reset to the value of LSB_DEBUG_MBD, LSB_DEBUG_SBD.

The log file is also reset back to the default log file.

***host_name* ...**

Optional. Sets debug settings on the specified host or hosts.

Lists of host names must be separated by spaces and enclosed in quotation marks.

Default: local host (host from which command was submitted)

sbdtime [-l *timing_level*] [-f *logfile_name*] [-o] [*host_name* ...]

Sets the timing level for `sbatchd` to include additional timing information in log files. You must be root or the LSF administrator to use this command.

In MultiCluster, timing levels can only be set for hosts within the same cluster. For example, you could not set debug or timing levels from a host in clusterA for a host in clusterB. You need to be on a host in clusterB to set up debug or timing levels for clusterB hosts.

If the command is used without any options, the following default values are used:

timing_level=no timing information is recorded

logfile_name=current LSF system log file in the LSF system log file directory, in the format *daemon_name*.log.*host_name*

host_name=local host (host from which command was submitted)

-l *timing_level*

Specifies detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

Valid values: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

Default: undefined (no timing information is logged)

-f *logfile_name*

Specify the name of the file into which timing messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file created has the following format:

logfile_name.daemon_name.log.*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Note: Both timing and debug messages are logged in the same files.

Default: current LSF system log file in the LSF system log file directory, in the format *daemon_name*.log.*host_name*.

-o

Optional. Turn off temporary timing settings and reset them to the daemon starting state. The timing level is reset back to the value of the parameter for the corresponding daemon (LSB_TIME_MBD, LSB_TIME_SBD).

The log file is also reset back to the default log file.

***host_name* ...**

Sets the timing level on the specified host or hosts.

Lists of hosts must be separated by spaces and enclosed in quotation marks.

Default: local host (host from which command was submitted)

schddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o]

Sets message log level for `mbschd` to include additional information in log files. You must be root or the LSF administrator to use this command.

See `sbddebug` for an explanation of options.

schdtime [-l *timing_level*] [-f] [-o]

Sets timing level for `mbschd` to include additional timing information in log files. You must be root or the LSF administrator to use this command.

See `sbdtime` for an explanation of options.

showconf mbd | [sbd [*host_name* … | all]]

Display all configured parameters and their values set in `lsf.conf` or `ego.conf` that affect `mbatchd` and `sbatchd`.

In a MultiCluster environment, `badmin showconf` only displays the parameters of daemons on the local cluster.

Running `badmin showconf` from a master candidate host reaches all server hosts in the cluster. Running `badmin showconf` from a slave-only host may not be able to reach other slave-only hosts.

`badmin showconf` only displays the values used by LSF.

For example, if you define LSF_MASTER_LIST in `lsf.conf`, and EGO_MASTER_LIST in `ego.conf`, `badmin showconf` displays the value of EGO_MASTER_LIST.

`badmin showconf` displays the value of EGO_MASTER_LIST from wherever it is defined. You can define either LSF_MASTER_LIST or EGO_MASTER_LIST in `lsf.conf`. LIM reads `lsf.conf` first, and ego.conf if EGO is enabled in the LSF cluster. The value of LSF_MASTER_LIST is displayed only if EGO_MASTER_LIST is not defined at all in `ego.conf`.

For example, if EGO is enabled in the LSF cluster, and you define LSF_MASTER_LIST in `lsf.conf`, and EGO_MASTER_LIST in `ego.conf`, `badmin showconf` displays the value of EGO_MASTER_LIST in ego.conf.

If EGO is disabled, `ego.conf` not loaded, so whatever is defined in `lsf.conf` is displayed.

perfmon start [*sample_period*] | stop | view | setperiod *sample_period*

Dynamically enables and controls scheduler performance metric collection.

Collecting and recording performance metric data may affect the performance of LSF. Smaller sampling periods results in the `lsb.streams` file growing faster.

The following metrics are collected and recorded in each sample period:

- The number of queries handled by `mbatchd`
- The number of queries for each of jobs, queues, and hosts. (`bjobs`, `bqueues`, and `bhosts`, as well as other daemon requests)
- The number of jobs submitted (divided into job submission requests and jobs actually submitted)
- The number of jobs dispatched
- The number of jobs completed
- The numbers of jobs sent to remote cluster
- The numbers of jobs accepted by from cluster
- The file descriptors used by `mbatchd`

start [*sample_period*]

Start performance metric collection dynamically and specifies an optional sampling period in seconds for performance metric collection.

If no sampling period is specified, the default period set in SCHED_METRIC_SAMPLE_PERIOD in `lsb.params` is used.

stop

Stop performance metric collection dynamically.

view

Display real time performance metric information for the current sampling period

setperiod *sample_period*

Set a new sampling period in seconds.

See also

bqueues, bhosts, `lsb.params`, `lsb.queues`, `lsb.hosts`, `lsf.conf`, `lsf.cluster`, `sbatchd`, `mbatchd`, `mbschd`

bapp

Displays information about application profile configuration.

Synopsis

bapp [**-l** | **-w**] [*application_profile_name* ...]

bapp [**-h** | **-V**]

Description

Displays information about application profiles configured in `lsb.applications`.

Returns application name, job slot statistics, and job state statistics for all application profiles:

In MultiCluster, returns the information about all application profiles in the local cluster.

CPU time is normalized.

Options

-w

Wide format. Fields are displayed without truncation.

-l

Long format with additional information.

Displays the following additional information: application profile description, application profile characteristics and statistics, parameters, resource usage limits, associated commands, binding policy, and job controls.

***application_profile_name* ...**

Displays information about the specified application profile.

-h

Prints command usage to `stderr` and exits.

-V

Prints product release version to `stderr` and exits.

Default output format

Displays the following fields:

APPLICATION_NAME

The name of the application profile. Application profiles are named to correspond to the type of application that usually runs within them.

NJOBS

The total number of job slots held currently by jobs in the application profile. This includes pending, running, suspended and reserved job slots. A parallel job that is

running on *n* processors is counted as *n* job slots, since it takes *n* job slots in the application.

PEND

The number of job slots used by pending jobs in the application profile.

RUN

The number of job slots used by running jobs in the application profile.

SUSP

The number of job slots used by suspended jobs in the application profile.

Long output format (-l)

In addition to the above fields, the -l option displays the following:

Description

A description of the typical use of the application profile.

PARAMETERS/ STATISTICS

SSUSP

The number of job slots in the application profile allocated to jobs that are suspended by LSF because of load levels or run windows.

USUSP

The number of job slots in the application profile allocated to jobs that are suspended by the job submitter or by the LSF administrator.

RSV

The number of job slots in the application profile that are reserved by LSF for pending jobs.

Per-job resource usage limits

The soft resource usage limits that are imposed on the jobs associated with the application profile. These limits are imposed on a per-job and a per-process basis.

The possible per-job limits are:

CPULIMIT

The maximum CPU time a job can use, in minutes, relative to the CPU factor of the named host. CPULIMIT is scaled by the CPU factor of the execution host so that jobs are allowed more time on slower hosts.

MEMLIMIT

The maximum running set size (RSS) of a process.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

MEMLIMIT_TYPE

A memory limit is the maximum amount of memory a job is allowed to consume. Jobs that exceed the level are killed. You can specify different types of memory limits to enforce, based on PROCESS, TASK, or JOB (or any combination of the three).

PROCESSLIMIT

The maximum number of concurrent processes allocated to a job.

PROCLIMIT

The maximum number of processors allocated to a job.

SWAPLIMIT

The swap space limit that a job may use.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

THREADLIMIT

The maximum number of concurrent threads allocated to a job.

Per-process resource usage limits

The possible UNIX per-process resource limits are:

CORELIMIT

The maximum size of a core file.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

DATALIMIT

The maximum size of the data segment of a process, in KB. This restricts the amount of memory a process can allocate.

FILELIMIT

The maximum file size a process can create, in KB.

RUNLIMIT

The maximum wall clock time a process can use, in minutes. RUNLIMIT is scaled by the CPU factor of the execution host.

STACKLIMIT

The maximum size of the stack segment of a process. This restricts the amount of memory a process can use for local variables or recursive function calls.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

BIND_JOB

The processor binding policy for sequential and parallel job processes enabled in the application profile. Displays one of: NONE, BALANCE, PACK, ANY, USER, or USER_CPU_LIST.

For example:

bapp -l app1

```
APPLICATION NAME: app1
 -- test processor binding options
……
PARAMETERS:
BIND_JOB: ANY
```

For backwards compatibility, `bapp -l` displays "Y" or "N" if BIND_JOB is defined with those values in the application profile.

CHKPNT_DIR

The checkpoint directory, if automatic checkpointing is enabled for the application profile.

CHKPNT_INITPERIOD

The initial checkpoint period in minutes. The periodic checkpoint does not happen until the initial period has elapsed.

CHKPNT_PERIOD

The checkpoint period in minutes. The running job is checkpointed automatically every checkpoint period.

CHKPNT_METHOD

The checkpoint method.

MIG

The migration threshold in minutes. A value of 0 (zero) specifies that a suspended job should be migrated immediately.

Where a host migration threshold is also specified, and is lower than the job value, the host value is used.

PRE_EXEC

The pre-execution command for the application profile. The PRE_EXEC command runs on the execution host before the job associated with the application profile is dispatched to the execution host (or to the first host selected for a parallel batch job).

POST_EXEC

The post-execution command for the application profile. The POST_EXEC command runs on the execution host after the job finishes.

JOB_INCLUDE_POSTPROC

If JOB_INCLUDE_POSTPROC= Y, post-execution processing of the job is included as part of the job.

JOB_POSTPROC_TIMEOUT

Timeout in minutes for job post-execution processing. If post-execution processing takes longer than the timeout, `sbatchd` reports that post-execution has failed (POST_ERR status), and kills the process group of the job's post-execution processes.

REQUEUE_EXIT_VALUES

Jobs that exit with these values are automatically requeued.

RES_REQ

Resource requirements of the application profile. Only the hosts that satisfy these resource requirements can be used by the application profile.

JOB_STARTER

An executable file that runs immediately prior to the batch job, taking the batch job file as an input argument. All jobs submitted to the application profile are run via the job starter, which is generally used to create a specific execution environment before processing the jobs themselves.

CHUNK_JOB_SIZE

Chunk jobs only. Specifies the maximum number of jobs allowed to be dispatched together in a chunk job. All of the jobs in the chunk are scheduled and dispatched as a unit rather than individually.

RERUNNABLE

If the RERUNNABLE field displays `yes`, jobs in the application profile are automatically restarted or rerun if the execution host becomes unavailable. However, a job in the application profile is not restarted if you use `bmod` to remove the rerunnable option from the job.

RESUME_CONTROL

The configured actions for the resume job control.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is RESUME.

SUSPEND_CONTROL

The configured actions for the suspend job control.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is SUSPEND.

TERMINATE_CONTROL

The configured actions for terminate job control.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is TERMINATE.

See also

`lsb.applications`, `lsb.queues`, bsub, bjobs, badmin, mbatchd

bbot

Moves a pending job relative to the last job in the queue.

Synopsis

bbot *job_ID* | **"***job_ID***[***index_list***]"** [*position*]

bbot -h | **-V**

Description

Changes the queue position of a pending job or job array element, to affect the order in which jobs are considered for dispatch.

By default, LSF dispatches jobs in a queue in the order of arrival (that is, first-come, first-served), subject to availability of suitable server hosts.

The `bbot` command allows users and the LSF administrator to manually change the order in which jobs are considered for dispatch. Users can only operate on their own jobs, whereas the LSF administrator can operate on any user's jobs.

If invoked by the LSF administrator, `bbot` moves the selected job after the last job with the same priority submitted to the queue.

If invoked by a user, `bbot` moves the selected job after the last job with the same priority submitted by the user to the queue.

Pending jobs are displayed by `bjobs` in the order in which they are considered for dispatch.

A user may use `bbot` to change the dispatch order of their jobs scheduled using a fairshare policy. However, if a job scheduled using a fairshare policy is moved by the LSF administrator using `btop`, the job is not subject to further fairshare scheduling unless the same job is subsequently moved by the LSF administrator using `bbot`; in this case the job is scheduled again using the same fairshare policy.

To prevent users from changing the queue position of a pending job with `bbot`, configure JOB_POSITION_CONTROL_BY_ADMIN=Y in `lsb.params`.

You cannot run `bbot` on jobs pending in an absolute priority scheduling (APS) queue.

Options

***job_ID* | "*job_ID*[*index_list*]"**

Required. Job ID of the job or job array on which to operate.

For a job array, the index list, the square brackets, and the quotation marks are required. An index list is used to operate on a job array. The index list is a comma separated list whose elements have the syntax start_index[-end_index[:step]] where start_index, end_index and step are positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one. The maximum job array index is 1000. All jobs in the array share the same job_ID and parameters. Each element of the array is distinguished by its array index.

position

Optional. The position argument can be specified to indicate where in the queue the job is to be placed. position is a positive number that indicates the target position of the job from the end of the queue. The positions are relative to only the applicable jobs

in the queue, depending on whether the invoker is a regular user or the LSF administrator. The default value of 1 means the position is after all other jobs with the same priority.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

bjobs(1), bswitch(1), btop(1), JOB_POSITION_CONTROL_BY_ADMIN in `lsb.params`

bchkpnt

checkpoints one or more checkpointable jobs

Synopsis

bchkpnt [**-f**] [**-k**] [**-app** *application_profile_name*] [**-p** *minutes* | **-p 0**] *job_ID* | "*job_ID*[*index_list*]" ...

bchkpnt [**-f**] [**-k**] [**-app** *application_profile_name*] [**-p** *minutes* | **-p 0**] **-J** *job_name* |**-m** *host_name* | **-m** *host_group* |**-q** *queue_name* |**-u** "*user_name*" | **-u all** [**0**]

bchkpnt -h | **-V**

Description

Checkpoints the most recently submitted running or suspended checkpointable job.

LSF administrators and `root` can checkpoint jobs submitted by other users.

Jobs continue to execute after they have been checkpointed.

LSF invokes the echkpnt(8) executable found in LSF_SERVERDIR to perform the checkpoint.

Only running members of a chunk job can be checkpointed. For chunk jobs in WAIT state, mbatchd rejects the checkpoint request.

Options

0

(Zero). Checkpoints all of the jobs that satisfy other specified criteria.

-f

Forces a job to be checkpointed even if non-checkpointable conditions exist (these conditions are OS-specific).

-app *application_profile_name*

Operates only on jobs associated with the specified application profile. You must specify an existing application profile. If *job_ID* or 0 is not specified, only the most recently submitted qualifying job is operated on.

-k

Kills a job after it has been successfully checkpointed.

-p *minutes* | -p 0

Enables periodic checkpointing and specifies the checkpoint period, or modifies the checkpoint period of a checkpointed job. Specify `-p 0` (zero) to disable periodic checkpointing.

Checkpointing is a resource-intensive operation. To allow your job to make progress while still providing fault tolerance, specify a checkpoint period of 30 minutes or longer.

-J *job_name*

Checkpoints only jobs that have the specified job name.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-m *host_name* | -m *host_group*

Checkpoints only jobs dispatched to the specified hosts.

-q *queue_name*

Checkpoints only jobs dispatched from the specified queue.

-u "*user_name*" | -u all

Checkpoints only jobs submitted by the specified users. The keyword all specifies all users. Ignored if a job ID other than 0 (zero) is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME\user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME\\user_name*) in a UNIX command line.

***job_ID* | "*job_ID*[*index_list*]"**

Checkpoints only the specified jobs.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

bchkpnt 1234

Checkpoints the job with job ID 1234.

bchkpnt -p 120 1234

Enables periodic checkpointing or changes the checkpoint period to 120 minutes (2 hours) for a job with job ID 1234.

bchkpnt -m hostA -k -u all 0

When issued by root or the LSF administrator, checkpoints and kills all checkpointable jobs on `hostA`. This is useful when a host needs to be shut down or rebooted.

See also

bsub(1), bmod(1), brestart(1), bjobs(1), bqueues(1), bhosts(1), `libckpt.a`(3), lsb.queues(5), `echkpnt`(8), `erestart`(8), mbatchd(8)

bclusters

displays MultiCluster information

Synopsis

bclusters [**-app**]

bclusters [**-h** | **-V**]

Description

For the job forwarding model, displays a list of MultiCluster queues together with their relationship with queues in remote clusters.

For the resource leasing model, displays remote resource provider and consumer information, resource flow information, and connection status between the local and remote cluster.

Options

-app

Displays available application profiles in remote clusters.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output: Job Forwarding Information

Displays a list of MultiCluster queues together with their relationship with queues in remote clusters.

Information related to the job forwarding model is displayed under the heading Job Forwarding Information.

LOCAL_QUEUE

Name of a local MultiCluster send-jobs or receive-jobs queue.

JOB_FLOW

Indicates direction of job flow.

send

The local queue is a MultiCluster send-jobs queue (SNDJOBS_TO is defined in the local queue).

recv

The local queue is a MultiCluster receive-jobs queue (RCVJOBS_FROM is defined in the local queue).

REMOTE

For send-jobs queues, shows the name of the receive-jobs queue in a remote cluster.

For receive-jobs queues, always “-”.

CLUSTER

For send-jobs queues, shows the name of the remote cluster containing the receive-jobs queue.

For receive-jobs queues, shows the name of the remote cluster that can send jobs to the local queue.

STATUS

Indicates the connection status between the local queue and remote queue.

ok

The two clusters can exchange information and the system is properly configured.

disc

Communication between the two clusters has not been established. This could occur because there are no jobs waiting to be dispatched, or because the remote master cannot be located.

reject

The remote queue rejects jobs from the send-jobs queue. The local queue and remote queue are connected and the clusters communicate, but the queue-level configuration is not correct. For example, the send-jobs queue in the submission cluster points to a receive-jobs queue that does not exist in the remote cluster.

If the job is rejected, it returns to the submission cluster.

Output: Resource Lease Information

Displays remote resource provider and consumer information, resource flow information, and connection status between the local and remote cluster.

Information related to the resource leasing model is displayed under the heading Resource Lease Information.

REMOTE_CLUSTER

For borrowed resources, name of the remote cluster that is the provider.

For exported resources, name of the remote cluster that is the consumer.

RESOURCE_FLOW

Indicates direction of resource flow.

IMPORT

Local cluster is the consumer and borrows resources from the remote cluster (HOSTS parameter in one or more local queue definitions includes remote resources).

EXPORT

Local cluster is the provider and exports resources to the remote cluster.

STATUS

Indicates the connection status between the local and remote cluster.

ok

MultiCluster jobs can run.

disc

No communication between the two clusters. This could be a temporary situation or could indicate a MultiCluster configuration error.

conn

The two clusters communicate, but the lease is not established. This should be a temporary situation.

Output: Remote Cluster Application Information

`bcluster -app` displays information related to application profile configuration under the heading Remote Cluster Application Information. Application profile information is only displayed for the job forwarding model. `bclusters` does not show local cluster application profile information.

REMOTE_CLUSTER

The name of the remote cluster.

APP_NAME

The name of the application profile available in the remote cluster.

DESCRIPTION

The description of the application profile.

Files

Reads `lsb.queues` and `lsb.applications`.

See also

bapp, bhosts, bqueues, lsclusters, lsinfo, lsb.queues

bgadd

creates job groups

Synopsis

bgadd [**-L** *limit*] [**-sla** *service_class_name*] *job_group_name*

bgadd [**-h** | **-V**]

Description

Creates a job group with the job group name specified by *job_group_name*.

You must provide full group path name for the new job group. The last component of the path is the name of the new group to be created.

You do not need to create the parent job group before you create a sub-group under it. If no groups in the job group hierarchy exist, all groups are created with the specified hierarchy.

Options

-L *limit*

Specifies the maximum number of concurrent jobs allowed to run under the job group (including child groups) -L limits the number of started jobs (RUN, SSUSP, USUSP) under the job group. Specify a positive number between 0 and 2147483647. If the specified limit is zero (0), no jobs under the job group can run.

You cannot specify a limit for the root job group. The root job group has no job limit. Job groups added with no limits specified inherit any limits of existing parent job groups. The -L option only limits the lowest level job group created.

If a parallel job requests 2 CPUs (`bsub -n 2`), the job group limit is per job, not per slots used by the job.

By default, a job group has no job limit. Limits persist across `mbatchd` restart or reconfiguration.

-sla *service_class_name*

The name of a service class defined in `lsb.serviceclasses`, or the name of the SLA defined in ENABLE_DEFAULT_EGO_SLA in `lsb.params`. The job group is attached to the specified SLA.

job_group_name

Full path of the job group name.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

- Create a job group named `risk_group` under the root group /:

  ```
  bgadd /risk_group
  ```

- Create a job group named `portfolio1` under job group `/risk_group`:

  ```
  bgadd /risk_group/portfolio1
  ```

See also

bgdel, bjgroup

bgdel

deletes job groups

Synopsis

bgdel [**-u** *user_name* | **-u all**] *job_group_name* | **0**

bgdel -c *job_group_name*

bgdel [**-h** | **-V**]

Description

Deletes a job group with the job group name specified by *job_group_name* and all its subgroups.

You must provide full group path name for the job group to be deleted. Deletion only takes effect after all jobs belonging to the group are cleaned out of `mbatchd` memory after the clean period.

Users can only delete their own job groups. LSF administrators can delete any job groups.

Job groups can be created explicitly or implicitly:

- A job group is created explicitly with the `bgadd` command.
- A job group is created implicitly by the `bsub -g` or `bmod -g` command when the specified group does not exist. Job groups are also created implicitly when a default job group is configured (DEFAULT_JOBGROUP in `lsb.params` or LSB_DEFAULT_JOBGROUP environment variable).

Options

0

Delete the empty job groups. These groups can be explicit or implicit.

-u *user_name*

Delete empty job groups owned by the specified user. Only administrators can use this option. These groups can be explicit or implicit. If you specify a job group name, the -u option is ignored.

-u all

Delete empty job groups and their sub groups for all users. Only administrators can use this option. These groups can be explicit or implicit. If you specify a job group name, the -u option is ignored.

-c *job_group_name*

Delete all the empty groups below the requested *job_group_name* including the *job_group_name* itself. These groups can be explicit or implicit.

job_group_name

Full path of the job group name.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Example

```
bgdel /risk_group
Job group /risk_group is deleted.
```

deletes the job group `/risk_group` and all its subgroups.

See also

bgadd, bjgroup

bgmod

modifies job groups

Synopsis

bgmod [**-L** *limit* | **-Ln**] *job_group_name*

bgmod [**-h** | **-V**]

Description

Modifies the job group with the job group name specified by *job_group_name*.

Only root, LSF administrators, the job group creator, or the creator of the parent job groups can use `bgmod` to modify a job group limit.

You must provide full group path name for the modified job group. The last component of the path is the name of the job group to be modified.

Options

-L *limit*

Changes the limit of *job_group_name* to the specified *limit* value. If the job group has parent job groups, the new limit cannot exceed the limits of any higher level job groups. Similarly, if the job group has child job groups, the new value must be greater than any limits on the lower level job groups.

limit specifies the maximum number of concurrent jobs allowed to run under the job group (including child groups) -L limits the number of started jobs (RUN, SSUSP, USUSP) under the job group. Specify a positive number between 0 and 2147483647. If the specified limit is zero (0), no jobs under the job group can run.

You cannot specify a limit for the root job group. The root job group has no job limit. The -L option only limits the lowest level job group specified.

If a parallel job requests 2 CPUs (`bsub -n 2`), the job group limit is per job, not per slots used by the job.

-Ln

Removes the existing job limit for the job group. If the job group has parent job groups, the job modified group automatically inherits any limits from its direct parent job group.

job_group_name

Full path of the job group name.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

The following command only modifies the limit of group `/canada/projects/test1`. It does not modify limits of `/canada` or`/canada/projects`.

```
bgmod -L 6 /canada/projects/test1
```

To modify limits of `/canada` or`/canada/projects`, you must specify the exact group name:

```
bgmod -L 6 /canada
```

or

```
bgmod -L 6 /canada/projects
```

See also

bgadd, bgdel, bjgroup

bhist

displays historical information about jobs

Synopsis

bhist [**-a** |**-d** | **-e** |**-p** | **-r** | **-s**] [**-b** | **-w**] [**-l**] [**-C** *start_time,end_time*] [**-D** *start_time,end_time*] [**-f** *logfile_name* | **-n** *number_logfiles* | **-n 0**] [**-S** *start_time,end_time*] [**-J** *job_name*] [**-Lp** *ls_project_name*] [**-m "***host_name***"** ...] [**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [**-P** *project_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u all** | **-G** *user_group*]

bhist [**-t**] [**-f** *logfile_name*] [**-T** *start_time,end_time*]

bhist [**-J** *job_name*] [**-N** *host_name* | **-N** *host_model* | **-N** *cpu_factor*] [*job_ID* ... | **"***job_ID***[***index***]"** ...]

bhist [**-h** | **-V**]

Description

By default:

- Displays information about your own pending, running and suspended jobs. Groups information by job
- CPU time is not normalized
- Searches the event log file currently used by the LSF system: `$LSB_SHAREDIR/`*cluster_name*`/logdir/lsb.events` (see `lsb.events`(5))
- Displays events occurring in the past week, but this can be changed by setting the environment variable LSB_BHIST_HOURS to an alternative number of hours

Options

-a

Displays information about both finished and unfinished jobs.

This option overrides -d, -p, -s, and -r.

-b

Brief format.

-d

Only displays information about finished jobs.

-e

Only displays information about exited jobs.

-l

Long format.

If the job was submitted `bsub -K`, the `-l` option displays `Synchronous execution`.

If you submitted a job using the OR (||) expression to specify alternative resources, this option displays the successful Execution rusage string with which the job ran.

If you submitted a job with multiple resource requirement strings using the `bsub` -R option for the order, same, rusage, and select sections, `bhist -l` displays a single,

merged resource requirement string for those sections, as if they were submitted using a single -R.

`bhist -l` displays job exit codes.

`bhist -l` displays jobs exceptions. For example, if a job's runtime exceeds the runtime estimate, a job exception of `runtime_est_exceeded` displays.

`bhist -l` displays resizable job information.

`bhist -l` display changes to pending jobs as a result of the following `bmod` options:

- Absolute priority scheduling (-aps | -apsn)
- Runtime estimate (-We | -Wen)
- Post-execution command (-Ep | -Epn)
- User limits (-ul | -uln)
- Current working directory (-cwd | -cwdn)
- Checkpoint options (-k | -kn)
- Migration threshold (-mig | -mign)
- Autoreszizable job attribute (-ar | -arn)
- Job resize notification command (-rnc | -rncn)

-p

Only displays information about pending jobs.

-r

Only displays information about running jobs.

-s

Only displays information about suspended jobs.

-t

Displays job events chronologically.

-w

Wide format. Displays the information in a wide format.

-C *start_time*,*end_time*

Only displays jobs that completed or exited during the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format "`yyyy/mm/dd/HH:MM`". Do not specify spaces in the time interval string.

The time interval can be specified in many ways.

-D *start_time*,*end_time*

Only displays jobs dispatched during the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the

start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format "`yyyy/mm/dd/HH:MM`". Do not specify spaces in the time interval string.

The time interval can be specified in many ways.

-G *user_group*

Only displays jobs associated with a user group submitted with `bsub -G` for the specified user group. The –G option does not display jobs from subgroups within the specified user group.

The -G option cannot be used together with the -u option. You can only specify a user group name. The keyword all is not supported for -G.

-S *start_time*,*end_time*

Only displays information about jobs submitted during the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format "`yyyy/mm/dd/HH:MM`". Do not specify spaces in the time interval string.

The time interval can be specified in many ways.

-T *start_time*,*end_time*

Used together with -t.

Only displays information about job events within the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format `yyyy/mm/dd/HH:MM`. Do not specify spaces in the time interval string.

The time interval can be specified in many ways.

-f *logfile_name*

Searches the specified event log. Specify either an absolute or a relative path.

Useful for analysis directly on the file.

The specified file path can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

-J *job_name*

Only displays the jobs that have the specified job name.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-Lp *ls_project_name*

Only displays information about jobs belonging to the specified License Scheduler project.

-m "*host_name*" ...

Only displays jobs dispatched to the specified host.

-n *number_logfiles* | -n 0

Searches the specified number of event logs, starting with the current event log and working through the most recent consecutively numbered logs. The maximum number of logs you can search is 100. Specify 0 to specify all the event log files in `$(LSB_SHAREDIR)/cluster_name/logdir` (up to a maximum of 100 files).

If you delete a file, you break the consecutive numbering, and older files are inaccessible to `bhist`.

For example, if you specify 3, LSF searches `lsb.events`, `lsb.events.1`, and `lsb.events.2`. If you specify 4, LSF searches `lsb.events`, `lsb.events.1`, `lsb.events.2`, and `lsb.events.3`. However, if `lsb.events.2` is missing, both searches include only `lsb.events` and `lsb.events.1`.

-N *host_name* | -N *host_model* | -N *cpu_factor*

Normalizes CPU time by the specified CPU factor, or by the CPU factor of the specified host or host model.

If you use `bhist` directly on an event log, you must specify a CPU factor.

Use `lsinfo` to get host model and CPU factor information.

-P *project_name*

Only displays information about jobs belonging to the specified project.

-q *queue_name*

Only displays information about jobs submitted to the specified queue.

-u *user_name* | -u all

Displays information about jobs submitted by the specified user, or by all users if the keyword all is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single back slash (*DOMAIN_NAME\user_name*) in a Windows command line or a double back slash (*DOMAIN_NAME\\user_name*) in a UNIX command line.

***job_ID* | "*job_ID*[*index*]"**

Searches all event log files and only displays information about the specified jobs. If you specify a job array, displays all elements chronologically.

This option overrides all other options except -J, -N, -h, and -V. When it is used with -J, only those jobs listed here that have the specified job name are displayed.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

Output: Default format

Statistics of the amount of time that a job has spent in various states:

PEND

The total waiting time excluding user suspended time before the job is dispatched.

PSUSP

The total user suspended time of a pending job.

RUN

The total run time of the job.

USUSP

The total user suspended time after the job is dispatched.

SSUSP

The total system suspended time after the job is dispatched.

UNKWN

The total unknown time of the job (job status becomes unknown if `sbatchd` on the execution host is temporarily unreachable).

TOTAL

The total time that the job has spent in all states; for a finished job, it is the turnaround time (that is, the time interval from job submission to job completion).

Output: Long format (-l)

The `-l` option displays a long format listing with the following additional fields:

Project

The project the job was submitted from.

Application Profile

The application profile the job was submitted to.

Command

The job command.

Detailed history includes job group modification, the date and time the job was forwarded and the name of the cluster to which the job was forwarded.

The displayed job command can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

Initial checkpoint period

The initial checkpoint period specified at the job level, by `bsub -k`, or in an application profile with CHKPNT_INITPERIOD.

Checkpoint period

The checkpoint period specified at the job level, by `bsub -k`, in the queue with CHKPNT, or in an application profile with CHKPNT_PERIOD.

Checkpoint directory

The checkpoint directory specified at the job level, by `bsub -k`, in the queue with CHKPNT, or in an application profile with CHKPNT_DIR.

Migration threshold

The migration threshold specified at the job level, by `bsub -mig`.

Resizable job information

- For JOB_NEW events, `bhist` displays the auto resizable attribute and resize notification command in the submission line.
- For JOB_MODIFY2 events (`bmod`), `bhist` displays the auto resizable attribute and resize notification command in the submission line.
  - `bmod -arn` *jobID*:

    ```
    Parameters of Job are changed: Autoresizable attribute is removed;
    ```
  - `bmod -ar` *jobID*:

    ```
    Parameters of Job are changed: Job changes to autoresizable;
    ```
  - `bmod -rnc` *resize_notification_cmd jobID*:

    ```
    Parameters of Job are changed: Resize notification command changes to:
    <resize_notification_cmd>;
    ```
  - `bmod -rncn` *jobID*:

    ```
    Parameters of Job are changed: Resize notification command is removed;
    ```
- For JOB_RESIZE_NOTIFY_START event, `bhist` displays:

  ```
  Additional allocation on <num_hosts> Hosts/Processors <host_list>
  ```
- For JOB_RESIZE_NOTIFY_ACCEPT event, `bhist` displays the following:
  - If the notification command is configured and `sbatchd` successfully initializes notification command. `bhist` displays

    ```
    Resize notification accepted. Notification command initialized (Command
    PID: 123456)
    ```
  - If a notification command is not defined, `bhist` displays

    ```
    Resize notification accepted
    ```
  - If `sbatchd` reports failure for whatever reason, `bhist` displays

    ```
    Resize notification failed
    ```
- For JOB_RESIZE_NOTIFY_DONE event, `bhist` displays the following:
  - `Resize notification command completed` if status is 0

- `Resize notification command failed` if status is 1

- For JOB_RESIZE_RELEASE event, `bhist` displays

```
Release allocation on <num_hosts> Hosts/Processors <host_list> by user or
administrator <user_name>, Resize notification command: <command_line>,
Cancel pending allocation request;
```

  For `bmod -rncn`, `bhist` displays

```
Resize notification command disabled
```

- For JOB_RESIZE_CANCEL event, `bhist` displays

```
Cancel pending allocation request
```

Synchronous execution

Job was submitted with the `-K` option. LSF submits the job and waits for the job to complete.

Files

Reads `lsb.events`

See also

`lsb.events`, bgadd, bgdel, bjgroup, bsub, bjobs, lsinfo

Time interval format

You use the time interval to define a start and end time for collecting the data to be retrieved and displayed. While you can specify both a start and an end time, you can also let one of the values default. You can specify either of the times as an absolute time, by specifying the date or time, or you can specify them relative to the current time.

Specify the time interval is follows:

start_time,*end_time*|*start_time*,|,*end_time*|*start_time*

Specify *start_time* or *end_time* in the following format:

[*year*/][*month*/][*day*][/*hour*:*minute*|/*hour*:]|.|.-*relative_int*

Where:

- *year* is a four-digit number representing the calendar year.
- *month* is a number from 1 to 12, where 1 is January and 12 is December.
- *day* is a number from 1 to 31, representing the day of the month.
- *hour* is an integer from 0 to 23, representing the hour of the day on a 24-hour clock.
- *minute* is an integer from 0 to 59, representing the minute of the hour.
- . (period) represents the current month/day/hour:minute.
- .-*relative_int* is a number, from 1 to 31, specifying a relative start or end time prior to now.

start_time,end_time

Specifies both the start and end times of the interval.

start_time,

Specifies a start time, and lets the end time default to now.

,end_time

Specifies to start with the first logged occurrence, and end at the time specified.

start_time

Starts at the beginning of the most specific time period specified, and ends at the maximum value of the time period specified. For example, `2/` specifies the month of February—start February 1 at 00:00 a.m. and end at the last possible minute in February: February 28th at midnight.

Absolute time examples

Assume the current time is May 9 17:06 2008:

1,8 = May 1 00:00 2008 to May 8 23:59 2008

,4 = the time of the first occurrence to May 4 23:59 2008

6 = May 6 00:00 2008 to May 6 23:59 2008

2/ = Feb 1 00:00 2008 to Feb 28 23:59 2008

/12: = May 9 12:00 2008 to May 9 12:59 2008

2/1 = Feb 1 00:00 2008 to Feb 1 23:59 2008

2/1, = Feb 1 00:00 to the current time

,. = the time of the first occurrence to the current time

,2/10: = the time of the first occurrence to May 2 10:59 2008

2001/12/31,2008/5/1 = from Dec 31, 2001 00:00:00 to May 1st 2008 23:59:59

Relative time examples

.-9, = April 30 17:06 2008 to the current time

,.-2/ = the time of the first occurrence to Mar 7 17:06 2008

.-9,.-2 = nine days ago to two days ago (April 30 17:06 2008 to May 7 17:06 2008)

bhosts

displays hosts and their static and dynamic resources

Synopsis

bhosts [**-e** | **-l** | **-w**] [**-x**] [**-X**] [**-R "***res_req***"**] [*host_name* | *host_group* | *compute_unit*] ...

bhosts [**-e** | **-l** | **-w**] [**-X**] [**-R "***res_req***"**] [*cluster_name*]

bhosts [**-e**] **-s** [*resource_name* ...]

bhosts [**-h** | **-V**]

Description

By default, returns the following information about all hosts: host name, host status, job state statistics, and job slot limits.

`bhosts` displays output for condensed host groups and compute units. These host groups and compute units are defined by CONDENSE in the HostGroup and ComputeUnit sectiosn of `lsb.hosts`. Condensed host groups and compute units are displayed as a single entry with the name as defined by GROUP_NAME or NAME in `lsb.hosts`.

When LSF adds more resources to a running resizable job, `bhosts` displays the added resources. When LSF removes resources from a running resizable job, `bhosts` displays the updated resources.

The -l and -X options display uncondensed output.

The -s option displays information about the numeric resources (shared or host-based) and their associated hosts.

With MultiCluster, displays the information about hosts available to the local cluster. Use -e to view information about exported hosts.

Options

-e

MultiCluster only. Displays information about resources that have been exported to another cluster.

-l

Displays host information in a (long) multi-line format. In addition to the default fields, displays information about the CPU factor, the current load, and the load thresholds.

Also displays information about the dispatch windows.

If you specified an administrator comment with the `-C` option of the host control commands `hclose` or `hopen`, `-l` displays the comment text.

-w

Displays host information in wide format. Fields are displayed without truncation.

For condensed host groups and compute units , the `-w` option displays the overall status and the number of hosts with the `ok`, `unavail`, `unreach`, and `busy` status in the following format:

host_group_status num_ok/*num_unavail*/*num_unreach*/*num_busy*

where

- *host_group_status* is the overall status of the host group or compute unit. If a single host in the group or unit is `ok`, the overall status is also `ok`.
- *num_ok*, *num_unavail*, *num_unreach*, and *num_busy* are the number of hosts that are `ok`, `unavail`, `unreach`, and `busy`, respectively.

For example, if there are five `ok`, two `unavail`, one `unreach`, and three `busy` hosts in a condensed host group `hg1`, its status is displayed as the following:

```
hg1 ok 5/2/1/3
```

If any hosts in the host group or compute unit are closed, the status for the host group is displayed as closed, with no status for the other states:

```
hg1 closed
```

-x

Display hosts whose job exit rate has exceeded the threshold configured by EXIT_RATE in `lsb.hosts` for longer than JOB_EXIT_RATE_DURATION configured in `lsb.params`, and are still high. By default, these hosts are closed the next time LSF checks host exceptions and invokes `eadmin`.

Use with the `-l` option to show detailed information about host exceptions.

If no hosts exceed the job exit rate, `bhosts -x` displays:

```
There is no exceptional host found
```

-X

Displays uncondensed output for host groups and compute units.

-R "*res_req*"

Only displays information about hosts that satisfy the resource requirement expression. For more information about resource requirements, see *Administering Platform LSF*. The size of the resource requirement string is limited to 512 bytes.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

-s [*resource_name* ...]

Displays information about the specified resources (shared or host-based). The resources must have numeric values. Returns the following information: the resource names, the total and reserved amounts, and the resource locations.

`bhosts -s` only shows consumable resources.

When LOCAL_TO is configured for a license feature in `lsf.licensescheduler`, `bhosts -s` shows different resource information depending on the cluster locality of the features. For example:

From clusterA:

```
bhosts -s
RESOURCE                TOTAL       RESERVED       LOCATION
hspice                  36.0        0.0            host1
```

From clusterB in siteB:

```
bhosts -s
RESOURCE                TOTAL       RESERVED        LOCATION
hspice                  76.0        0.0             host2
```

host_name ... | host_group ... | compute unit ...

Only displays information about the specified hosts. Do not use quotes when specifying multiple hosts.

For host groups and compute units, the names of the member hosts are displayed instead of the name of the host group or compute unit. Do not use quotes when specifying multiple host groups or compute units.

cluster_name

MultiCluster only. Displays information about hosts in the specified cluster.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output: Host-Based Default

Displays the following fields:

HOST_NAME

The name of the host. If a host has batch jobs running and the host is removed from the configuration, the host name is displayed as lost_and_found.

For condensed host groups, this is the name of host group.

STATUS

With MultiCluster, not shown for fully exported hosts.

The current status of the host and the `sbatchd` daemon. Batch jobs can only be dispatched to hosts with an `ok` status. The possible values for host status are as follows:

ok

The host is available to accept batch jobs.

For condensed host groups, if a single host in the host group is `ok`, the overall status is also shown as `ok`.

If any host in the host group or compute unit is not ok, `bhosts` displays the first host status it encounters as the overall status for the condensed host group. Use `bhosts -X` to see the status of individual hosts in the host group or compute unit.

unavail

The host is down, or LIM and `sbatchd` on the host are unreachable.

unreach

LIM on the host is running but sbatchd is unreachable.

closed

The host is not allowed to accept any remote batch jobs. There are several reasons for the host to be closed (see Host-Based -l Options).

unlicensed

The host does not have a valid LSF license.

closed_cu_excl

This host is a member ofa compute unit running an exclusive compute unit job.

JL/U

With MultiCluster, not shown for fully exported hosts.

The maximum number of job slots that the host can process on a per user basis. If a dash (-) is displayed, there is no limit.

For condensed host groups or compute units, this is the total number of job slots that all hosts in the group or unit can process on a per user basis.

The host does not allocate more than JL/U job slots for one user at the same time. These job slots are used by running jobs, as well as by suspended or pending jobs that have slots reserved for them.

For preemptive scheduling, the accounting is different. These job slots are used by running jobs and by pending jobs that have slots reserved for them (see the description of PREEMPTIVE in `lsb.queues`(5) and JL/U in `lsb.hosts(5)`).

MAX

The maximum number of job slots available. If a dash (-) is displayed, there is no limit.

For condensed host groups and compute units, this is the total maximum number of job slots available in all hosts in the host group or compute unit.

These job slots are used by running jobs, as well as by suspended or pending jobs that have slots reserved for them.

If preemptive scheduling is used, suspended jobs are not counted (see the description of PREEMPTIVE in `lsb.queues(5)` and MXJ in `lsb.hosts(5)`).

A host does not always have to allocate this many job slots if there are waiting jobs; the host must also satisfy its configured load conditions to accept more jobs.

NJOBS

The number of job slots used by jobs dispatched to the host. This includes running, suspended, and chunk jobs.

For condensed host groups and compute units, this is the total number of job slots used by jobs dispatched to any host in the host group or compute unit.

RUN

The number of job slots used by jobs running on the host.

For condensed host groups and compute units, this is the total number of job slots used by jobs running on any host in the host group or compute unit.

SSUSP

The number of job slots used by system suspended jobs on the host.

For condensed host groups and compute units, this is the total number of job slots used by system suspended jobs on any host in the host group or compute unit.

USUSP

The number of job slots used by user suspended jobs on the host. Jobs can be suspended by the user or by the LSF administrator.

For condensed host groups and compute units, this is the total number of job slots used by user suspended jobs on any host in the host group or compute unit.

RSV

The number of job slots used by pending jobs that have jobs slots reserved on the host.

For condensed host groups and compute units, this is the total number of job slots used by pending jobs that have job slots reserved on any host in the host group or compute unit.

Output: Host-Based -l Option

In addition to the above fields, the -l option also displays the following:

loadSched, loadStop

The scheduling and suspending thresholds for the host. If a threshold is not defined, the threshold from the queue definition applies. If both the host and the queue define a threshold for a load index, the most restrictive threshold is used.

The migration threshold is the time that a job dispatched to this host can remain suspended by the system before LSF attempts to migrate the job to another host.

If the host's operating system supports checkpoint copy, this is indicated here. With checkpoint copy, the operating system automatically copies all open files to the checkpoint directory when a process is checkpointed. Checkpoint copy is currently supported only on Cray systems.

STATUS

The long format shown by the -l option gives the possible reasons for a host to be closed:

closed_Adm

The host is closed by the LSF administrator or root (see `badmin(8)`). No job can be dispatched to the host, but jobs that are executing on the host are not affected.

closed_Lock

The host is locked by the LSF administrator or root (see `lsadmin(8)`). All batch jobs on the host are suspended by LSF.

closed_Wind

The host is closed by its dispatch windows, which are defined in the configuration file `lsb.hosts(5)`. Jobs already started are not affected by the dispatch windows.

closed_Full

The configured maximum number of batch job slots on the host has been reached (see MAX field below).

closed_Excl

The host is currently running an exclusive job.

closed_Cu_Excl

The host belongs to a compute unit currently running an exclusive job.

closed_Busy

The host is overloaded, because some load indices go beyond the configured thresholds (see `lsb.hosts(5)`). The displayed thresholds that cause the host to be busy are preceded by an asterisk (*).

closed_LIM

LIM on the host is unreachable, but `sbatchd` is ok.

closed_EGO

For EGO-enabled SLA scheduling, host is closed because it has not been allocated by EGO to run LSF jobs. Hosts allocated from EGO display status ok.

CPUF

Displays the CPU normalization factor of the host (see `lshosts(1)`).

DISPATCH_WINDOW

Displays the dispatch windows for each host. Dispatch windows are the time windows during the week when batch jobs can be run on each host. Jobs already started are not affected by the dispatch windows. When the dispatch windows close, jobs are not suspended. Jobs already running continue to run, but no new jobs are started until the windows reopen. The default for the dispatch window is no restriction or always open (that is, twenty-four hours a day and seven days a week). For the dispatch window specification, see the description for the DISPATCH_WINDOWS keyword under the `-l` option in `bqueues(1)`.

CURRENT LOAD

Displays the total and reserved host load.

Reserved

You specify reserved resources by using `bsub -R`. These resources are reserved by jobs running on the host.

Total

The total load has different meanings depending on whether the load index is increasing or decreasing.

For increasing load indices, such as run queue lengths, CPU utilization, paging activity, logins, and disk I/O, the total load is the consumed plus the reserved amount. The total load is calculated as the sum of the current load and the reserved load. The current load is the load seen by `lsload(1)`.

For decreasing load indices, such as available memory, idle time, available swap space, and available space in tmp, the total load is the available amount. The total load is the difference between the current load and the reserved load. This difference is the available resource as seen by `lsload(1)`.

LOAD THRESHOLD

Displays the scheduling threshold `loadSched` and the suspending threshold `loadStop`. Also displays the migration threshold if defined and the checkpoint support if the host supports checkpointing.

The format for the thresholds is the same as for batch job queues (see `bqueues(1)`) and `lsb.queues(5)`). For an explanation of the thresholds and load indices, see the description for the "QUEUE SCHEDULING PARAMETERS" keyword under the -l option in `bqueues(1)`.

THRESHOLD AND LOAD USED FOR EXCEPTIONS

Displays the configured threshold of EXIT_RATE for the host and its current load value for host exceptions.

ADMIN ACTION COMMENT

If the LSF administrator specified an administrator comment with the -C option of the `badmin` host control commands `hclose` or `hopen`, the comment text is displayed.

Output: Resource-Based -s Option

The -s option displays the following: the amounts used for scheduling, the amounts reserved, and the associated hosts for the resources. Only resources (shared or host-based) with numeric values are displayed. See `lim(8)`, and `lsf.cluster(5)` on how to configure shared resources.

The following fields are displayed:

RESOURCE

The name of the resource.

TOTAL

The total amount free of a resource used for scheduling.

RESERVED

The amount reserved by jobs. You specify the reserved resource using `bsub -R`.

LOCATION

The hosts that are associated with the resource.

Files

Reads `lsb.hosts`.

See also

`lsb.hosts`, bqueues, lshosts, badmin, lsadmin

bhpart

displays information about host partitions

Synopsis

bhpart [**-r**] [*host_partition_name* ...]

bhpart [**-h** | **-V**]

Description

By default, displays information about all host partitions. Host partitions are used to configure host-partition fairshare scheduling.

Options

-r

Displays the entire information tree associated with the host partition recursively.

host_partition_name ...

Displays information about the specified host partitions only.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

The following fields are displayed for each host partition:

HOST_PARTITION_NAME

Name of the host partition.

HOSTS

Hosts or host groups that are members of the host partition. The name of a host group is appended by a slash (`/`) (see `bmgroup`(1)).

USER/GROUP

Name of users or user groups who have access to the host partition (see `bugroup`(1)).

SHARES

Number of shares of resources assigned to each user or user group in this host partition, as configured in the file `lsb.hosts`. The shares affect dynamic user priority for when fairshare scheduling is configured at the host level.

PRIORITY

Dynamic user priority for the user or user group. Larger values represent higher priorities. Jobs belonging to the user or user group with the highest priority are considered first for dispatch.

In general, users or user groups with larger SHARES, fewer STARTED and RESERVED, and a lower CPU_TIME and RUN_TIME have higher PRIORITY.

STARTED

Number of job slots used by running or suspended jobs owned by users or user groups in the host partition.

RESERVED

Number of job slots reserved by the jobs owned by users or user groups in the host partition.

CPU_TIME

Cumulative CPU time used by jobs of users or user groups executed in the host partition. Measured in seconds, to one decimal place.

LSF calculates the cumulative CPU time using the actual (not normalized) CPU time and a decay factor such that 1 hour of recently-used CPU time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default).

RUN_TIME

Wall-clock run time plus historical run time of jobs of users or user groups that are executed in the host partition. Measured in seconds.

LSF calculates the historical run time using the actual run time of finished jobs and a decay factor such that 1 hour of recently-used run time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default). Wall-clock run time is the run time of running jobs.

ADJUST

Dynamic priority calculation adjustment made by the user-defined fairshare plugin (`libfairshareadjust.*`).

The fairshare adjustment is enabled and weighted by the parameter FAIRSHARE_ADJUSTMENT_FACTOR in `lsb.params`.

Files

Reads `lsb.hosts`.

See also

bugroup(1), bmgroup(1), lsb.hosts(5)

bjdepinfo

Displays job dependencies.

Synopsis

bjdepinfo [-r *level*] [-l] [-p] *job_ID* | "*job_ID*[*index*]"

bjdepinfo -c [-r *level*] *job_ID* | "*job_ID*[*index*]"

bjdepinfo [-h] [-V]

Description

The command `bjdepinfo` allows you to display all or seleted job dependencies. You can get a list of other jobs that a job depends on (parent jobs) and jobs that depend on your job (child jobs).

Note:

The parent-child relationship does not indicate that one or more jobs are spawned by other jobs. A job dependency is when the start of a job depends on the status of other jobs.

Options

***job_ID* | "*job_ID*[*index*]"**

Required. Job ID of the job or job array on which to operate.

If you specify only a job ID for a job array, information about all jobs in the array displays.

Displays all jobs that this job depends on.

-r *level*

When combined with -p, prints the parent jobs that cause the current job to pend recursively.

When combined with -c, prints the child jobs that depend on the current job recursively.

When combined with -l, prints detailed parent job dependency information recursively. When combined with -l and -p, prints detailed information about the parent jobs that cause the current job to pend recursively.

In each case, you can specify the level using a positive integer. Level indicates the number of degrees of separation from the original job.

For example, specify level 1 to see any jobs that directly depend on this job or that this job depends on. Specify level 2 if you also want to see all dependencies on the jobs that have a dependency on the originally specified job.

If the job has been partially cleaned, an asterisk (*) displays before the status and the job name is unavailable (-).

-l

For the job you specify, prints detailed parent job dependency information including the condition of the job and whether or not a job's dependency requirements have been satisfied.

-p

Prints the parent jobs that cause the current job to pend.

-c

Prints any child jobs of the job you specify (as well as any dependencies they have).

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

Output

JOBID

The job ID of the job with a parent or child dependency.

PARENT

The job ID of the job that has other jobs depending on it.

CHILD

The job ID of the job that depends on other jobs.

PARENT_STATUS

The status of the parent job listed. If the job has been partially cleaned, an asterisk (*) displays before the status and the job name is unavailable (-).

CHILD_STATUS

The status of the child job listed.

PARENT_NAME

The name of the parent job listed. If the job has been partially cleaned, the job name is unavailable (-) and an asterisk (*) displays before the status.

CHILD_NAME

The name of the child job listed.

LEVEL

The degrees of separation of job dependencies. 1 means a job is directly dependent; other numbers indicate the levels of indirect dependency.

bjgroup

displays information about job groups

Synopsis

bjgroup [**-N**] [**-s** [*group_name*]]

bjgroup [**-h** | **-V**]

Description

Displays job group information.

When LSF adds more resources to a running resizable job, `bjgroups` displays the added resources. When LSF removes resources from a running resizable job, `bjgroups` displays the updated resources.

Options

-s

Sorts job groups by group hierarchy.

For example, for job groups named `/A`, `/A/B`, `/X` and `/X/Y`, `bjgroup` without `-s` displays:

bjgroup

```
GROUP_NAME      NJOBS   PEND    RUN     SSUSP   USUSP   FINISH  SLA    JLIMIT  OWNER
/A              0       0       0       0       0       0       ()       0/10  user1
/X              0       0       0       0       0       0       ()        0/-  user2
/A/B            0       0       0       0       0       0       ()        0/5  user1
/X/Y            0       0       0       0       0       0       ()        0/5  user2
```

For the same job groups, `bjgroup -s` displays:

bjgroup -s

```
GROUP_NAME      NJOBS   PEND    RUN     SSUSP   USUSP   FINISH  SLA    JLIMIT  OWNER
/A              0       0       0       0       0       0       ()       0/10  user1
/A/B            0       0       0       0       0       0       ()        0/5  user1
/X              0       0       0       0       0       0       ()        0/-  user2
/X/Y            0       0       0       0       0       0       ()        0/5  user2
```

Specify a job group name to show the hierarchy of a single job group:

bjgroup -s /X

```
GROUP_NAME   NJOBS  PEND   RUN   SSUSP  USUSP  FINISH     SLA   JLIMIT  OWNER
/X              25     0    25       0      0       0  puccini  25/100  user1
/X/Y            20     0    20       0      0       0  puccini   20/30  user1
/X/Z             5     0     5       0      0       0  puccini    5/10  user2
```

Specify a job group name with a trailing slash character (/) to show only the root job group:

```
bjgroup -s /X/
GROUP_NAME   NJOBS  PEND   RUN   SSUSP  USUSP  FINISH     SLA   JLIMIT  OWNER
/X              25     0    25       0      0       0  puccini  25/100  user1
```

-N

Displays job group information by job slots instead of number of jobs. NSLOTS, PEND, RUN, SSUSP, USUSP, RSV are all counted in slots rather than number of jobs:

```
bjgroup -N
GROUP_NAME NSLOTS PEND   RUN   SSUSP  USUSP   RSV     SLA     OWNER
/X             25    0    25       0      0     0  puccini     user1
/A/B           20    0    20       0      0     0   wagner     batch
```

-N by itself shows job slot info for all job groups, and can combine with -s to sort the job groups by hierarchy:

```
bjgroup -N -s
GROUP_NAME NSLOTS PEND   RUN   SSUSP   USUSP  RSV     SLA     OWNER
/A             0    0     0       0       0    0   wagner      batch
/A/B           0    0     0       0       0    0   wagner      user1
/X            25    0    25       0       0    0   puccini     user1
/X/Y          20    0    20       0       0    0   puccini     batch
/X/Z           5    0     5       0       0    0   puccini     batch
```

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Default output

A list of job groups is displayed with the following fields:

GROUP_NAME

The name of the job group.

NJOBS

The current number of jobs in the job group. A parallel job is counted as 1 job, regardless of the number of job slots it uses.

PEND

The number of pending jobs in the job group.

RUN

The number of running jobs in the job group.

SSUSP

The number of system-suspended jobs in the job group.

USUSP

The number of user-suspended jobs in the job group.

FINISH

The number of jobs in the specified job group in EXITED or DONE state.

SLA

The name of the service class that the job group is attached to with `bgadd -sla` *service_class_name*. If the job group is not attached to any service class, empty parentheses `()` are displayed in the SLA name column.

JLIMIT

The job group limit set by `bgadd -L` or `bgmod -L`. Job groups that have no configured limits or no limit usage are indicated by a dash (`-`). Job group limits are displayed in a USED/LIMIT format. For example, if a limit of 5 jobs is configured and 1 job is started, `bjgroup` displays the job limit under JLIMIT as `1/5`.

OWNER

The job group owner.

Example

bjgroup

```
GROUP_NAME  NJOBS  PEND   RUN   SSUSP  USUSP  FINISH      SLA   JLIMIT  OWNER
/fund1_grp      5     4     0       1      0       0   Venezia     1/5   user1
/fund2_grp     11     2     5       0      0       4   Venezia     5/5   user1
/bond_grp       2     2     0       0      0       0   Venezia     0/-   user2
/risk_grp       2     1     1       0      0       0        ()     1/-   user2
/admi_grp       4     4     0       0      0       0        ()     0/-   user2
```

Job slots (-N) output

NSLOTS, PEND, RUN, SSUSP, USUSP, RSV are all counted in slots rather than number of jobs. A list of job groups is displayed with the following fields:

GROUP_NAME

The name of the job group.

NSLOTS

The total number of job slots held currently by jobs in the job group. This includes pending, running, suspended and reserved job slots. A parallel job that is running on *n* processors is counted as *n* job slots, since it takes *n* job slots in the job group.

PEND

The number of job slots used by pending jobs in the job group.

RUN

The number of job slots used by running jobs in the job group.

SSUSP

The number of job slots used by system-suspended jobs in the job group.

USUSP

The number of job slots used by user-suspended jobs in the job group.

RSV

The number of job slots in the job group that are reserved by LSF for pending jobs.

SLA

The name of the service class that the job group is attached to with `bgadd -sla` *service_class_name*. If the job group is not attached to any service class, empty parentheses `()` are displayed in the SLA name column.

OWNER

The job group owner.

Example

bjgroup -N

```
GROUP_NAME NSLOTS PEND   RUN   SSUSP  USUSP   RSV      SLA     OWNER
/X             25    0    25       0      0     0  puccini     user1
/A/B           20    0    20       0      0     0   wagner     batch
```

See also

bgadd, bgdel, bgmod

bjobs

displays information about LSF jobs

Synopsis

bjobs [**-A**] [**-a**] [**-d**] [**-p**] [**-s**] [**-r**] [**-W**] [**-w** | **-l** | **-aps**] [**-X**] [**-x**] [**-app** *application_profile_name*] [**-g** *job_group_name*] [**-sla** *service_class_name*] [**-J** *job_name*] [**-Lp** *ls_project_name*] [**-m "***host_name ...***"** | **-m** *host_group* | **-m** *compute_unit* | **-m** *cluster_name*] [**-N** *host_name* | **-N** *host_model* | **-N** *cpu_factor*] [**-P** *project_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all** | **-G** *user_group*] *job_ID* | **"***job_ID***[***index_list***]"** ...

bjobs [**-a**] [**-aps**] [**-d**] [**-p**] [**-s**] [**-r**] [**-WL**] [**-WP**] [**-WF**] [**-X**] [**-x**] [**-app** *application_profile_name*] [**-g** *job_group_name*] [**-sla** *service_class_name*] [**-J** *job_name*] [**-Lp** *ls_project_name*] [**-m "***host_name ...***"** | **-m** *host_group* | **-m** *compute_unit* | **-m** *cluster_name*] [**-P** *project_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all** | **-G** *user_group*] *job_ID* | **"***job_ID***[***index_list***]"** ...

bjobs [**-a**] [**-w** | **-l**] [**-d**] [**-p**] [**-s**] [**-ss**] [**-r**] [**-X**] [**-x**] [**-app** *application_profile_name*] [**-g** *job_group_name*] [**-sla** *service_class_name*] [**-J** *job_name*] [**-Lp** *ls_project_name*] [**-m "***host_name ...***"** | **-m** *host_group* | **-m** *compute_unit* | **-m** *cluster_name*] [**-P** *project_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all** | **-G** *user_group*] *job_ID* | **"***job_ID***[***index_list***]"** ...

bjobs [**-h** | **-V**]

Description

By default, displays information about your own pending, running and suspended jobs.

`bjobs` displays output for condensed host groups and compute units. These host groups and compute units are defined by CONDENSE in the `HostGroup` or `ComputeUnit` section of `lsb.hosts`. These groups are displayed as a single entry with the name as defined by GROUP_NAME or NAME in `lsb.hosts`. The -l and -X options display uncondensed output.

If you defined LSB_SHORT_HOSTLIST=1 in `lsf.conf`, parallel jobs running in the same condensed host group or compute unit are displayed as an abbreviated list.

For resizable jobs, `bjobs` displays the autoresizable attribute and the resize notification command.

To display older historical information, use `bhist`.

Options

-A

Displays summarized information about job arrays. If you specify job arrays with the job array ID, and also specify -A, do not include the index list with the job array ID.

You can use -w to show the full array specification, if necessary.

-a

Displays information about jobs in all states, including finished jobs that finished recently, within an interval specified by CLEAN_PERIOD in `lsb.params` (the default period is 1 hour).

Use -a with -x option to display all jobs that have triggered a job exception (overrun, underrun, idle).

-aps

Displays absolute priority scheduling (APS) information for pending jobs in a queue with APS_PRIORITY enabled. The APS value is calculated based on the current scheduling cycle, so jobs are not guaranteed to be dispatched in this order.

Pending jobs are ordered by APS value. Jobs with system APS values are listed first, from highest to lowest APS value. Jobs with calculated APS values are listed next ordered from high to low value. Finally, jobs not in an APS queue are listed. Jobs with equal APS values are listed in order of submission time. APS values of jobs not in an APS queue are shown with a dash (-).

If queues are configured with the same priority, `bjobs -aps` may not show jobs in the correct expected dispatch order. Jobs may be dispatched in the order the queues are configured in `lsb.queues`. You should avoid configuring queues with the same priority.

For resizable jobs, -aps displays the latest APS information for running jobs with active resize allocation requests. LSF handles the dynamic priority for running jobs with active resize requests. The displayed job priority can change from time to time.

-d

Displays information about jobs that finished recently, within an interval specified by CLEAN_PERIOD in `lsb.params` (the default period is 1 hour).

-l

Long format. Displays detailed information for each job in a multiline format.

The -l option displays the following additional information: project name, job command, current working directory on the submission host, initial checkpoint period, checkpoint directory, migration threshold, pending and suspending reasons, job status, resource usage, resource usage limits information, and runtime resource usage information on the execution hosts.

If the job was submitted with `bsub -K`, the -l option displays `Synchronous Execution`.

Use `bjobs -A -l` to display detailed information for job arrays including job array job limit (*%job_limit*) if set.

Use `bjobs -ss -l` to display detailed information for session scheduler jobs.

If JOB_IDLE is configured in the queue, use `bjobs -l` to display job idle exception information.

If you submitted your job with the -U option to use advance reservations created with the `brsvadd` command, `bjobs -l` shows the reservation ID used by the job.

If LSF_HPC_EXTENSIONS="SHORT_PIDLIST" is specified in `lsf.conf`, the output from bjobs is shortened to display only the first PID and a count of the process group IDs (PGIDs) and process IDs for the job. Without SHORT_PIDLIST, all of the process IDs (PIDs) for a job are displayed.

If you submitted a job with multiple resource requirement strings using the `bsub` -R option for the order, same, rusage, and select sections, `bjobs -l` displays a single,

merged resource requirement string for those sections, as if they were submitted using a single -R.

If you submitted a job using the OR (||) expression to specify alternative resources, this option displays the Execution rusage string with which the job runs.

For resizable jobs, the -l option displays active pending resize allocation requests, and the latest job priority for running jobs with active pending resize requests.

For jobs with user-based fairshare scheduling, displays the charging SAAP (share attribute account path).

For jobs submitted to an absolute priority scheduling (APS) queue, -l shows the ADMIN factor value and the system APS value if they have been set by the administrator for the job:

-p

Displays pending jobs, together with the pending reasons that caused each job not to be dispatched during the last dispatch turn. The pending reason shows the number of hosts for that reason, or names the hosts if -l is also specified.

With MultiCluster, `-l` shows the names of hosts in the local cluster.

Each pending reason is associated with one or more hosts and it states the cause why these hosts are not allocated to run the job. In situations where the job requests specific hosts (using `bsub -m`), users may see reasons for unrelated hosts also being displayed, together with the reasons associated with the requested hosts.

The life cycle of a pending reason ends after the time indicated by PEND_REASON_UPDATE_INTERVAL in `lsb.params`.

When the job slot limit is reached for a job array (`bsub -J "jobArray[indexList] %job_slot_limit"`) the following message is displayed:

```
The job array has reached its job slot limit.
```

-r

Displays running jobs.

-s

Displays suspended jobs, together with the suspending reason that caused each job to become suspended.

The suspending reason may not remain the same while the job stays suspended. For example, a job may have been suspended due to the paging rate, but after the paging rate dropped another load index could prevent the job from being resumed. The suspending reason is updated according to the load index. The reasons could be as old as the time interval specified by SBD_SLEEP_TIME in `lsb.params`. So the reasons shown may not reflect the current load situation.

-ss

Displays summary information for session scheduler tasks including the job ID, the owner, the job name (useful for job arrays), the total number of tasks, the state of pending, done, running, and exited session scheduler tasks.

The frequency of the updates of this information is based on the parameters SSCHED_UPDATE_SUMMARY_INTERVAL and SSCHED_UPDATE_SUMMARY_BY_TASK.

The following options cannot be used with -ss.

- -A
- -W
- -WL
- -WF
- -WP
- -N
- -aps

-W

Provides resource usage information for: PROJ_NAME, CPU_USED, MEM, SWAP, PIDS, START_TIME, FINISH_TIME.

-WF

Displays an estimated finish time for running or pending jobs. For done or exited jobs, displays the actual finish time.

-WL

Displays the estimated remaining run time of jobs.

-WP

Displays the current estimated completion percentage of jobs.

-w

Wide format. Displays job information without truncating fields.

-X

Displays uncondensed output for host groups and compute units.

-x

Displays unfinished jobs that have triggered a job exception (overrun, underrun, idle, runtime_est_exceeded). Use with the -l option to show the actual exception status. Use with -a to display all jobs that have triggered a job exception.

-app *application_profile_name*

Displays information about jobs submitted to the specified application profile. You must specify an existing application profile.

-G *user_group*

Only displays jobs associated with a user group submitted with `bsub -G` for the specified user group. The –G option does not display jobs from subgroups within the specified user group.

If ENFORCE_ONE_UG_LIMITS is enabled in `lsb.params`, using the -G option enforces any limits placed on the specified user group only even if the user or user group belongs to more than one group.

If ENFORCE_ONE_UG_LIMITS is disabled in `lsb.params` (default), using the -G option enforces the strictest limit that is set on any of the groups that the user or user group belongs to.

The -G option cannot be used together with the -u option. You can only specify a user group name. The keyword all is not supported for -G.

-g *job_group_name*

Displays information about jobs attached to the job group specified by *job_group_name*. For example:

```
bjobs -g /risk_group
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
113     user1   PEND  normal     hostA                   myjob      Jun 17 16:15
111     user2   RUN   normal     hostA       hostA       myjob      Jun 14 15:13
110     user1   RUN   normal     hostB       hostA       myjob      Jun 12 05:03
104     user3   RUN   normal     hostA       hostC       myjob      Jun 11 13:18
```

Use `-g` with `-sla` to display job groups attached to a service class. Once a job group is attached to a service class, all jobs submitted to that group are subject to the SLA.

`bjobs -l` with `-g` displays the full path to the group to which a job is attached. For example:

```
bjobs -l -g /risk_group
Job <101>, User <user1>, Project <default>, Job Group </risk_group>, Status
<RUN>, Queue <normal>, Command <myjob>
Tue Jun 17 16:21:49: Submitted from host <hostA>, CWD </home/user1;
Tue Jun 17 16:22:01: Started on <hostA>;
...
```

-J *job_name*

Displays information about the specified jobs or job arrays. Only displays jobs that were submitted by the user running this command.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-Lp *ls_project_name*

Displays jobs that belong to the specified LSF License Scheduler project.

-m *host_name* ... | -m *host_group* ... | -m *cluster_name* ...

Only displays jobs dispatched to the specified hosts. To see the available hosts, use `bhosts`.

If a host group or compute unit is specified, displays jobs dispatched to all hosts in the group. To determine the available host groups, use `bmgroup`. To determine the available compute units, use `bmgroup -cu`.

With MultiCluster, displays jobs in the specified cluster. If a remote cluster name is specified, you see the remote job ID, even if the execution host belongs to the local cluster. To determine the available clusters, use `bclusters`.

-N *host_name* | -N *host_model* | -N *cpu_factor*

Displays the normalized CPU time consumed by the job. Normalizes using the CPU factor specified, or the CPU factor of the host or host model specified.

-P *project_name*

Only displays jobs that belong to the specified project.

-q *queue_name*

Only displays jobs in the specified queue.

The command `bqueues` returns a list of queues configured in the system, and information about the configurations of these queues.

In MultiCluster, you cannot specify remote queues.

-sla *service_class_name*

Displays jobs belonging to the specified service class.

`bjobs` also displays information about jobs assigned to a default SLA configured with ENABLE_DEFAULT_EGO_SLA in `lsb.params`.

Use `-sla with -g` to display job groups attached to a service class. Once a job group is attached to a service class, all jobs submitted to that group are subject to the SLA.

Use `bsla` to display the configuration properties of service classes configured in `lsb.serviceclasses`, the default SLA configured in `lsb.params`, and dynamic information about the state of each service class.

-u *user_name*... | -u *user_group*... | -u all

Only displays jobs that have been submitted by the specified users or user groups. The keyword all specifies all users. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

The -u option cannot be used with the -G option.

***job_ID* | "*job_ID*[*index*]"**

Displays information about the specified jobs or job arrays.

If you use `-A`, specify job array IDs without the index list.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output: Default Display

Pending jobs are displayed in the order in which they are considered for dispatch. Jobs in higher priority queues are displayed before those in lower priority queues. Pending jobs in the same priority queues are displayed in the order in which they were submitted but this order can be changed by using the commands `btop` or `bbot`. If more than one job is dispatched to a host, the jobs on that host are listed in the order in which they are considered for scheduling on this host by their queue priorities and dispatch times. Finished jobs are displayed in the order in which they were completed.

A listing of jobs is displayed with the following fields:

JOBID

The job ID that LSF assigned to the job.

USER

The user who submitted the job.

STAT

The current status of the job (see JOB STATUS below).

QUEUE

The name of the job queue to which the job belongs. If the queue to which the job belongs has been removed from the configuration, the queue name is displayed as `lost_and_found`. Use `bhist` to get the original queue name. Jobs in the `lost_and_found` queue remain pending until they are switched with the `bswitch` command into another queue.

In a MultiCluster resource leasing environment, jobs scheduled by the consumer cluster display the remote queue name in the format *queue_name@cluster_name*. By default, this field truncates at 10 characters, so you might not see the cluster name unless you use -w or -l.

FROM_HOST

The name of the host from which the job was submitted.

With MultiCluster, if the host is in a remote cluster, the cluster name and remote job ID are appended to the host name, in the format *host_name@cluster_name:job_ID*. By default, this field truncates at 11 characters; you might not see the cluster name and job ID unless you use -w or -l.

EXEC_HOST

The name of one or more hosts on which the job is executing (this field is empty if the job has not been dispatched). If the host on which the job is running has been removed

from the configuration, the host name is displayed as `lost_and_found`. Use `bhist` to get the original host name.

If the host is part of a condensed host group or compute unit, the host name is displayed as the name of the condensed group.

If you configure a host to belong to more than one condensed host groups using wildcards, `bjobs` can display any of the host groups as execution host name.

JOB_NAME

The job name assigned by the user, or the command string assigned by default at job submission with `bsub`. If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

The displayed job name or job command can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

SUBMIT_TIME

The submission time of the job.

Output: -l

The `-l` option displays a long format listing with the following additional fields:

Project

The project the job was submitted from.

Application Profile

The application profile the job was submitted to.

Command

The job command.

CWD

The current working directory on the submission host.

Initial checkpoint period

The initial checkpoint period specified at the job level, by `bsub -k`, or in an application profile with CHKPNT_INITPERIOD.

Checkpoint period

The checkpoint period specified at the job level, by `bsub -k`, in the queue with CHKPNT, or in an application profile with CHKPNT_PERIOD.

Checkpoint directory

The checkpoint directory specified at the job level, by `bsub -k`, in the queue with CHKPNT, or in an application profile with CHKPNT_DIR.

Migration threshold

The migration threshold specified at the job level, by `bsub -mig`.

Post-execute Command

The post-execution command specified at the job-level, by `bsub -Ep`.

PENDING REASONS

The reason the job is in the PEND or PSUSP state. The names of the hosts associated with each reason are displayed when both `-p` and `-l` options are specified.

SUSPENDING REASONS

The reason the job is in the USUSP or SSUSP state.

loadSched

The load scheduling thresholds for the job.

loadStop

The load suspending thresholds for the job.

JOB STATUS

Possible values for the status of a job include:

PEND

The job is pending. That is, it has not yet been started.

PSUSP

The job has been suspended, either by its owner or the LSF administrator, while pending.

RUN

The job is currently running.

USUSP

The job has been suspended, either by its owner or the LSF administrator, while running.

SSUSP

The job has been suspended by LSF. The job has been suspended by LSF due to either of the following two causes:

- The load conditions on the execution host or hosts have exceeded a threshold according to the `loadStop` vector defined for the host or queue.
- The run window of the job's queue is closed. See `bqueues(1)`, `bhosts(1)`, and `lsb.queues(5)`.

DONE

The job has terminated with status of 0.

EXIT

The job has terminated with a non-zero status – it may have been aborted due to an error in its execution, or killed by its owner or the LSF administrator.

For example, exit code 131 means that the job exceeded a configured resource usage limit and LSF killed the job.

UNKWN

mbatchd has lost contact with the sbatchd on the host on which the job runs.

WAIT

For jobs submitted to a chunk job queue, members of a chunk job that are waiting to run.

ZOMBI

A job becomes ZOMBI if:

- A non-rerunnable job is killed by `bkill` while the sbatchd on the execution host is unreachable and the job is shown as UNKWN.
- The host on which a rerunnable job is running is unavailable and the job has been requeued by LSF with a new job ID, as if the job were submitted as a new job.
- After the execution host becomes available, LSF tries to kill the ZOMBI job. Upon successful termination of the ZOMBI job, the job's status is changed to EXIT.

  With MultiCluster, when a job running on a remote execution cluster becomes a ZOMBI job, the execution cluster treats the job the same way as local ZOMBI jobs. In addition, it notifies the submission cluster that the job is in ZOMBI state and the submission cluster requeues the job.

RUNTIME

Estimated run time for the job, specified by `bsub -We` or `bmod -We`, -We+, -Wep.

The following information is displayed when running `bjobs -WL`, `-WF`, or `-WP`.

TIME_LEFT

The estimated run time that the job has remaining. Along with the time if applicable, one of the following symbols may also display.

- E: The job has an estimated run time that has not been exceeded.
- L: The job has a hard run time limit specified but either has no estimated run time or the estimated run time is more than the hard run time limit.
- X: The job has exceeded its estimated run time and the time displayed is the time remaining until the job reaches its hard run time limit.
- A dash indicates that the job has no estimated run time and no run limit, or that it has exceeded its run time but does not have a hard limit and therefore runs until completion.

If there is less than a minute remaining, 0:0 displays.

FINISH_TIME

The estimated finish time of the job. For done/exited jobs, this is the actual finish time. For running jobs, the finish time is the start time plus the estimated run time (where set and not exceeded) or the start time plus the hard run limit.

- E: The job has an estimated run time that has not been exceeded.
- L: The job has a hard run time limit specified but either has no estimated run time or the estimated run time is more than the hard run time limit.
- X: The job has exceeded its estimated run time and had no hard run time limit set. The finish time displayed is the estimated run time remaining plus the start time.
- A dash indicates that the pending, suspended, or job with no run limit has no estimated finish time.

%COMPLETE

The estimated completion percentage of the job.

- E: The job has an estimated run time that has not been exceeded.
- L: The job has a hard run time limit specified but either has no estimated run time or the estimated run time is more than the hard run time limit.
- X: The job has exceeded its estimated run time and had no hard run time limit set.
- A dash indicates that the jobs is pending, or that it is running or suspended, but has no run time limit specified.

Note:

For jobs in the state UNKNOWN, the job run time estimate is based on internal counting by the job's mbatchd.

RESOURCE USAGE

For the MultiCluster job forwarding model, this information is not shown if MultiCluster resource usage updating is disabled.

The values for the current usage of a job include:

CPU time

Cumulative total CPU time in seconds of all processes in a job.

IDLE_FACTOR

Job idle information (CPU time/runtime) if JOB_IDLE is configured in the queue, and the job has triggered an idle exception.

MEM

Total resident memory usage of all processes in a job. By default, memory usage is shown in MB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

SWAP

Total virtual memory usage of all processes in a job. By default, swap space is shown in MB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

NTHREAD

Number of currently active threads of a job.

PGID

Currently active process group ID in a job.

PIDs

Currently active processes in a job.

RESOURCE LIMITS

The hard resource usage limits that are imposed on the jobs in the queue (see getrlimit (2) and `lsb.queues`(5)). These limits are imposed on a per-job and a per-process basis.

The possible per-job resource usage limits are:

- CPULIMIT
- PROCLIMIT
- MEMLIMIT
- SWAPLIMIT
- PROCESSLIMIT
- THREADLIMIT
- OPENFILELIMIT

The possible UNIX per-process resource usage limits are:

- RUNLIMIT
- FILELIMIT
- DATALIMIT
- STACKLIMIT
- CORELIMIT

If a job submitted to the queue has any of these limits specified (see `bsub`(1)), then the lower of the corresponding job limits and queue limits are used for the job.

If no resource limit is specified, the resource is assumed to be unlimited. User shell limits that are unlimited are not displayed.

EXCEPTION STATUS

Possible values for the exception status of a job include:

idle

The job is consuming less CPU time than expected. The job idle factor (CPU time/ runtime) is less than the configured JOB_IDLE threshold for the queue and a job exception has been triggered.

overrun

The job is running longer than the number of minutes specified by the JOB_OVERRUN threshold for the queue and a job exception has been triggered.

underrun

The job finished sooner than the number of minutes specified by the JOB_UNDERRUN threshold for the queue and a job exception has been triggered.

Synchronous Execution

Job was submitted with the -K option. LSF submits the job and waits for the job to complete.

Output: Job array summary information

If you use -A, displays summary information about job arrays. The following fields are displayed:

JOBID

Job ID of the job array.

ARRAY_SPEC

Array specification in the format of *name*[*index*]. The array specification may be truncated, use -w option together with -A to show the full array specification.

OWNER

Owner of the job array.

NJOBS

Number of jobs in the job array.

PEND

Number of pending jobs of the job array.

RUN

Number of running jobs of the job array.

DONE

Number of successfully completed jobs of the job array.

EXIT

Number of unsuccessfully completed jobs of the job array.

SSUSP

Number of LSF system suspended jobs of the job array.

USUSP

Number of user suspended jobs of the job array.

PSUSP

Number of held jobs of the job array.

Output: Session Scheduler job summary information

JOBID

Job ID of the Session Scheduler job.

OWNER

Owner of the Session Scheduler job.

JOB_NAME

The job name assigned by the user, or the command string assigned by default at job submission with `bsub`. If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

The displayed job name or job command can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

NTASKS

The total number of tasks for this Session Scheduler job.

PEND

Number of pending tasks of the Session Scheduler job.

RUN

Number of running tasks of the Session Scheduler job.

DONE

Number of successfully completed tasks of the Session Scheduler job.

EXIT

Number of unsuccessfully completed tasks of the Session Scheduler job.

Examples

bjobs -pl

Displays detailed information about all pending jobs of the invoker.

bjobs -ps

Display only pending and suspended jobs.

bjobs -u all -a

Displays all jobs of all users.

bjobs -d -q short -m hostA -u user1

Displays all the recently finished jobs submitted by `user1` to the queue `short`, and executed on the host `hostA`.

bjobs 101 102 203 509

Display jobs with job_ID 101, 102, 203, and 509.

bjobs -X 101 102 203 509

Display jobs with job ID 101, 102, 203, and 509 as uncondensed output even if these jobs belong to hosts in condensed groups.

bjobs -sla Uclulet

Displays all jobs belonging to the service class `Uclulet`.

bjobs -app fluent

Displays all jobs belonging to the application profile `fluent`.

See also

bsub, bkill, bhosts, bmgroup, bclusters, bqueues, bhist, bresume, bsla, bstop, `lsb.params`, `lsb.serviceclasses`, `mbatchd`

bkill

sends signals to kill, suspend, or resume unfinished jobs

Synopsis

bkill [**-l**] [**-app** *application_profile_name*] [**-g** *job_group_name*] [**-sla** *service_class_name*] [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group*] [**-q** *queue_name*] [**-r** | **-s** *signal_value* | *signal_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [*job_ID* ... | **0** | **"***job_ID***[***index***]"** ...]

bkill [**-l**] [**-b**] [**-app** *application_profile_name*] [**-g** *job_group_name*] [**-sla** *service_class_name*] [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [*job_ID* ... | **0** | **"***job_ID***[***index***]"** ...]

bkill [**-h** | **-V**]

Description

By default, sends a set of signals to kill the specified jobs. On UNIX, SIGINT and SIGTERM are sent to give the job a chance to clean up before termination, then SIGKILL is sent to kill the job. The time interval between sending each signal is defined by the JOB_TERMINATE_INTERVAL parameter in `lsb.params(5)`.

By default, kills the last job submitted by the user running the command. You must specify a job ID or -app, -g, -J, -m, -u, or -q. If you specify -app, -g, -J, -m, -u, or -q without a job ID, `bkill` kills the last job submitted by the user running the command. Specify job ID `0` (zero) to kill multiple jobs.

On Windows, job control messages replace the SIGINT and SIGTERM signals (but only customized applications can process them) and the `TerminateProcess()` system call is sent to kill the job.

Exit code 130 is returned when a dispatched job is killed with `bkill`.

If PRIVILEGED_USER_FORCE_BKILL=y in `lsb.params`, only root and LSF administrators can run `bkill -r`. The -r option is ignored for other users.

Users can only operate on their own jobs. Only root and LSF administrators can operate on jobs submitted by other users.

If a signal request fails to reach the job execution host, LSF tries the operation later when the host becomes reachable. LSF retries the most recent signal request.

If a job is running in a queue with CHUNK_JOB_SIZE set, `bkill` has the following results depending on job state:

PEND

Job is removed from chunk (NJOBS -1, PEND -1)

RUN

All jobs in the chunk are suspended (NRUN -1, NSUSP +1)

USUSP

Job finishes, next job in the chunk starts if one exists (NJOBS -1, PEND -1, SUSP -1, RUN +1)

WAIT

Job finishes (NJOBS-1, PEND -1)

If the job cannot be killed, use `bkill -r` to remove the job from the LSF system without waiting for the job to terminate, and free the resources of the job.

Options

0

Kills all the jobs that satisfy other options (-app. -g, -m, -q, -u, and -J).

-b

Kills large numbers of jobs as soon as possible. Local pending jobs are killed immediately and cleaned up as soon as possible, ignoring the time interval specified by CLEAN_PERIOD in `lsb.params`. Jobs killed in this manner are not logged to `lsb.acct`.

Other jobs, such as running jobs, are killed as soon as possible and cleaned up normally.

If the -b option is used with the 0 subcommand, `bkill` kills all applicable jobs and silently skips the jobs that cannot be killed.

```
bkill -b 0
Operation is in progress
```

The -b option is ignored if used with the -r or -s options.

-l

Displays the signal names supported by `bkill`. This is a subset of signals supported by `/bin/kill` and is platform-dependent.

-r

Removes a job from the LSF system without waiting for the job to terminate in the operating system.

If PRIVILEGED_USER_FORCE_BKILL=y in `lsb.params`, only root and LSF administrators can run `bkill -r`. The -r option is ignored for other users.

Sends the same series of signals as `bkill` without -r, except that the job is removed from the system immediately, the job is marked as EXIT, and the job resources that LSF monitors are released as soon as LSF receives the first signal.

Also operates on jobs for which a `bkill` command has been issued but that cannot be reached to be acted on by sbatchd (jobs in ZOMBI state). If sbatchd recovers before the jobs are completely removed, LSF ignores the zombie jobs killed with `bkill -r`.

Use `bkill -r` only on jobs that cannot be killed in the operating system, or on jobs that cannot be otherwise removed using `bkill`.

The -r option cannot be used with the -s option.

-app *application_profile_name*

Operates only on jobs associated with the specified application profile. You must specify an existing application profile. If *job_ID* or 0 is not specified, only the most recently submitted qualifying job is operated on.

-g *job_group_name*

Operates only on jobs in the job group specified by *job_group_name*.

Use `-g with -sla` to kill jobs in job groups attached to a service class.

`bkill` does not kill jobs in lower level job groups in the path. For example, jobs are attached to job groups `/risk_group` and `/risk_group/consolidate`:

```
bsub -g /risk_group  myjob
Job <115> is submitted to default queue <normal>.
bsub -g /risk_group/consolidate myjob2
Job <116> is submitted to default queue <normal>.
```

The following `bkill` command only kills jobs in `/risk_group`, not the subgroup `/risk_group/consolidate`:

```
bkill -g /risk_group 0
Job <115> is being terminated
bkill -g /risk_group/consolidate 0
Job <116> is being terminated
```

-J *job_name*

Operates only on jobs with the specified job name. The -J option is ignored if a job ID other than 0 is specified in the *job_ID* option.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-m *host_name* | -m *host_group*

Operates only on jobs dispatched to the specified host or host group.

If *job_ID* is not specified, only the most recently submitted qualifying job is operated on. The -m option is ignored if a job ID other than 0 is specified in the *job_ID* option. See `bhosts(1)` and `bmgroup(1)` for more information about hosts and host groups.

-q *queue_name*

Operates only on jobs in the specified queue.

If *job_ID* is not specified, only the most recently submitted qualifying job is operated on.

The -q option is ignored if a job ID other than 0 is specified in the *job_ID* option.

See `bqueues`(1) for more information about queues.

-s *signal_value* | *signal_name*

Sends the specified signal to specified jobs. You can specify either a name, stripped of the SIG prefix (such as KILL), or a number (such as 9).

Eligible UNIX signal names are listed by `bkill -l`.

The -s option cannot be used with the -r option.

Use `bkill -s` to suspend and resume jobs by using the appropriate signal instead of using `bstop` or `bresume`. Sending the SIGCONT signal is the same as using `bresume`.

Sending the SIGSTOP signal to sequential jobs or the SIGTSTP to parallel jobs is the same as using `bstop`.

You cannot suspend a job that is already suspended, or resume a job that is not suspended. Using SIGSTOP or SIGTSTP on a job that is in the USUSP state has no effect and using SIGCONT on a job that is not in either the PSUSP or the USUSP state has no effect. See `bjobs(1)` for more information about job states.

Limited Windows signals are supported:

- **bkill -s 7** or **bkill SIGKILL** to terminate a job
- **bkill -s 16** or **bkill SIGSTOP** to suspend a job
- **bkill -s 15** to resume a job

-sla *service_class_name*

Operates on jobs belonging to the specified service class.

If *job_ID* is not specified, only the most recently submitted job is operated on.

Use `-sla with -g` to kill jobs in job groups attached to a service class.

The -sla option is ignored if a job ID other than 0 is specified in the *job_ID* option.

Use `bsla` to display the configuration properties of service classes configured in `lsb.serviceclasses`, the default SLA configured with ENABLE_DEFAULT_EGO_SLA in `lsb.params`, and dynamic information about the state of each service class.

-u *user_name* | -u *user_group* | -u all

Operates only on jobs submitted by the specified user or user group, or by all users if the reserved user name all is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

If job_ID is not specified, only the most recently submitted qualifying job is operated on. The -u option is ignored if a job ID other than 0 is specified in the job_ID option.

***job_ID* ... | 0 | "*job_ID*[*index*]" ...**

Operates only on jobs that are specified by *job_ID* or "*job_ID*[*index*]", where "*job_ID*[*index*]" specifies selected job array elements (see `bjobs(1)`). For job arrays, quotation marks must enclose the job ID and index, and index must be enclosed in square brackets.

Kill an entire job array by specifying the job array ID instead of the job ID.

Jobs submitted by any user can be specified here without using the -u option. If you use the reserved job ID 0, all the jobs that satisfy other options (that is, -m, -q, -u and -J) are operated on; all other job IDs are ignored.

The options -u, -q, -m and -J have no effect if a job ID other than 0 is specified. Job IDs are returned at job submission time (see `bsub(1)`) and may be obtained with the `bjobs` command (see `bjobs(1)`).

Any jobs or job arrays that are killed are logged in `lsb.acct`.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

bkill -s 17 -q night

Sends signal 17 to the last job that was submitted by the invoker to queue `night`.

bkill -q short -u all 0

Kills all the jobs that are in the queue `short`.

bkill -r 1045

Forces the removal of unkillable job 1045.

bkill -sla Tofino 0

Kill all jobs belonging to the service class named `Tofino`.

bkill -g /risk_group 0

Kills all jobs in the job group `/risk_group`.

bkill -app fluent

Kills the most recently submitted job associated with the application profile `fluent` for the current user.

bkill -app fluent 0

Kills all jobs associated with the application profile `fluent` for the current user.

See also

bsub(1), bjobs(1), bqueues(1), bhosts(1), bresume(1), bapp(1), bsla(1), bstop(1), bgadd(1), bgdel(1), bjgroup(1), bparams(5), lsb.serviceclasses(5), mbatchd(8), `kill`(1), `signal`(2)

bladmin

reconfigures the Platform LSF License Scheduler daemon (bld)

Synopsis

bladmin *subcommand*

bladmin [**-h** | **-V**]

Description

Use this command to reconfigure the License Scheduler daemon (bld).

You must be a License Scheduler administrator to use this command.

Subcommand synopsis

ckconfig [**-v**]

reconfig [*host_name* ... | **all**]

shutdown [*host_name* ... | **all**]

blddebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**]

blcdebug [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] *collector_name* ... | all

-h

-V

Usage

ckconfig [-v]

Checks LSF License Scheduler configuration in `LSF_ENVDIR/lsf.licensescheduler` and `lsf.conf`.

By default, `bladmin ckconfig` displays only the result of the configuration file check. If warning errors are found, `bladmin` prompts you to use the -v option to display detailed messages.

-v

Verbose mode. Displays detailed messages about configuration file checking to `stderr`.

reconfig [*host_name* ... | all]

Reconfigures License Scheduler.

shutdown [*host_name* ... | all]

Shuts down License Scheduler.

blddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o]

Sets the message log level for `bld` to include additional information in log files. You must be root or the LSF administrator to use this command.

If the `bladmin blddebug` is used without any options, the following default values are used:

- *class_name*=0 (no additional classes are logged)
- *debug_level*=0 (LOG_DEBUG level in parameter LS_LOG_MASK)
- *logfile_name*=current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*

-c *class_name* ...

Specifies software classes for which debug messages are to be logged.

Format of *class_name* is the name of a class, or a list of class names separated by spaces and enclosed in quotation marks. Classes are also listed in `lsf.h`.

Valid log classes:

- LC_AUTH: Log authentication messages
- LC_COMM: Log communication messages
- LC_FLEX: Log everything related to FLEX_STAT or FLEX_EXEC Macrovision APIs
- LC_LICENCE: Log license management messages
- LC_PREEMPT: Log preemption policy messages
- LC_RESREQ: Log resource requirement messages
- LC_TRACE: Log significant program walk steps
- LC_XDR: Log everything transferred by XDR

Default: 0 (no additional classes are logged)

-l *debug_level*

Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

Possible values:

0 LOG_DEBUG level in parameter LS_LOG_MASK in `lsf.conf`.

1 LOG_DEBUG1 level for extended logging.

2 LOG_DEBUG2 level for extended logging.

3 LOG_DEBUG3 level for extended logging.

Default: 0 (LOG_DEBUG level in parameter LS_LOG_MASK)

-f *logfile_name*

Specifies the name of the file where debugging messages are logged. The file name can be a full path. If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file has the following format:

logfile_name.daemon_name.`log.`*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Default: current LSF system log file in the LSF system log file directory.

-o

Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LS_LOG_MASK and classes are reset to the value of LSB_DEBUG_BLD. The log file is also reset back to the default log file.

blcdebug [-l *debug_level*] [-f *logfile_name*] [-o] *collector_name* | all

Sets the message log level for `blcollect` to include additional information in log files. You must be root or the LSF administrator to use this command.

If the `bladmin blcdebug` is used without any options, the following default values are used:

- *debug_level*=0 (LOG_DEBUG level in parameter LS_LOG_MASK)
- *logfile_name*=current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*
- *collector_name*=default

-l *debug_level*

Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

Possible values:

0 LOG_DEBUG level in parameter LS_LOG_MASK in `lsf.conf`.

1 LOG_DEBUG1 level for extended logging.

2 LOG_DEBUG2 level for extended logging.

3 LOG_DEBUG3 level for extended logging.

Default: 0 (LOG_DEBUG level in parameter LS_LOG_MASK)

-f *logfile_name*

Specifies the name of the file where debugging messages are logged. The file name can be a full path. If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file has the following format:

logfile_name.daemon_name.`log.`*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Default: current LSF system log file in the LSF system log file directory.

-o

Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LS_LOG_MASK and classes are reset to the value of LSB_DEBUG_BLD. The log file is also reset back to the default log file.

If a collector name is not specified, default value is to restore the original log mask and log file directory for the default collector.

***collector_name* ... | all**

Specifies the collector names separated by blanks. all means all the collectors.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

See also

blhosts, `lsf.licensescheduler`, `lsf.conf`

blaunch

launches parallel tasks on a set of hosts

Synopsis

blaunch [**-n**] [**-u** *host_file* | **-z** *host_name* ... | *host_name*] [**-use-login-shell** | **-no-shell**] *command* [*argument* ...]

blaunch [**-h** | **-V**]

Description

Important:

You cannot run blaunch directly from the command line.

Restriction:

The command blaunch does not work with user account mapping. Do not run blaunch on a user account mapping host.

Most MPI implementations and many distributed applications use `rsh` and `ssh` as their task launching mechanism. The `blaunch` command provides a drop-in replacement for `rsh` and `ssh` as a transparent method for launching parallel applications within LSF.

`blaunch` supports the following core command line options as `rsh` and `ssh`:

- `rsh` *host_name command*
- `ssh` *host_name command*

All other `rsh` and `ssh` options are silently ignored.

`blaunch` transparently connects directly to the RES/SBD on the remote host, and subsequently creates and tracks the remote tasks, and provides the connection back to LSF. You do not need to insert `pam`, `taskstarter` or any other wrapper.

`blaunch` only works under LSF. It can only be used to launch tasks on remote hosts that are part of a job allocation. It cannot be used as a standalone command.

When no host names are specified, LSF runs tasks on all allocated hosts, one remote task per job slot.

Windows: `blaunch` is supported on Windows 2000 or later with the following exceptions:

- Only the following signals are supported: SIGKILL, SIGSTOP, SIGCONT.
- The -n option is not supported.
- CMD.EXE /C <user command line> is used as intermediate command shell when:
  - -no-shell is not specified
- CMD.EXE /C is not used when -no-shell is specified.
- Windows Vista User Account Control must be configured correctly to run jobs.

Options

-n

Standard input is taken from `/dev/null`. (Not supported on Windows.)

-u *host_file*

Executes the task on all hosts listed in the *host_file*.

Specify the path to a file that contains a list of host names. Each host name must listed on a separator line in the host list file.

This option is exclusive of the -z option.

host_name

The name of the host where remote tasks are to be launched.

-z *host_name* ...

Executes the task on all specified hosts.

Whereas the host name value for `rsh` and `ssh` is a single host name, you can use the -z option to specify a space-delimited list of hosts where tasks are started in parallel.

Specify a list of hosts on which to execute the task. If multiple host names are specified, the host names must be enclosed by quotation marks (" or ') and separated by white space.

This option is exclusive of the -u option.

-use-login-shell

Launches commands through user's login shell.

Only applies to UNIX and Linux hosts.

-no-shell

Launches commands without any intermediate shell.

***command* [*argument* ...]**

Specify the command to execute. This must be the last argument on the command line.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Diagnostics

Exit status is 0 if all commands are executed correctly.

See also

`lsb_getalloc`, `lsb_launch`

blcollect

license information collection daemon that collects license usage information

Synopsis

blcollect -c *collector_name* **-m** *host_name [...]* **-p** *license_scheduler_port* [**-i** *lmstat_interval* | **-D** *lmstat_path*]

blcollect [**-h** | **-V**]

Description

Periodically collects license usage information from Macrovision FLEXnet. It queries FLEXnet for license usage information from the FLEXnet `lmstat` command, and passes the information to the License Scheduler daemon (bld). The `blcollect` daemon improves performance by allowing you to distribute license information queries on multiple hosts.

By default, license information is collected from FLEXnet on one host. Use `blcollect` to distribute the license collection on multiple hosts.

For each service domain configuration in `lsf.licensescheduler`, specify one name for `blcollect` to use. You can only specify one collector per service domain, but you can specify one collector to serve multiple service domains. You can choose any collector name you want, but must use that exact name when you run `blcollect`.

Options

-c

Required. Specify the collector name you set in `lsf.licensescheduler`. You must use the collector name (`LIC_COLLECT`) you define in the `ServiceDomain` section of the configuration file.

-m

Required. Specifies a space-separated list of hosts to which license information is sent. The hosts do not need to be running License Scheduler or a FLEXnet. Use fully qualified host names.

-p

Required. You must specify the License Scheduler listening port, which is set in `lsf.licensescheduler` and has a default value of 9581.

-i *lmstat_interval*

Optional. The frequency in seconds of the calls that License Scheduler makes to `lmstat` to collect license usage information from FLEXnet.

The default interval is 60 seconds.

-D *lmstat_path*

Optional. Location of the FLEXnet command `lmstat`.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

See also

lsf.licensescheduler

blhosts

displays the names of all the hosts running the License Scheduler daemon (`bld`)

Synopsis

blhosts [**-h** | **-V**]

Description

Displays a list of hosts running the License Scheduler daemon. This includes the License Scheduler master host and all the candidate License Scheduler hosts running bld.

Options

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

Output

Prints out the names of all the hosts running the License Scheduler daemon (bld).

For example, the following sample output shows the License Scheduler master host and two candidate License Scheduler hosts running bld:

```
bld is running on:
master: host1.domain1.com
slave: host2.domain1 host3.domain1
```

See also

blinfo, blstat, bladmin

blimits

displays information about resource allocation limits of running jobs

Synopsis

blimits [**-w**] [**-n** *limit_name* ...] [**-m** *host_name* | **-m** host_group | **-m** *cluster_name* ...] [**-P** *project_name* ...] [**-q** *queue_name* ...] [**-u** *user_name* | **-u** *user_group* ...]

blimits -c

blimits -h | **-V**

Description

Displays current usage of resource allocation limits configured in Limit sections in `lsb.resources`:

- Configured limit policy name
- Users (-u option)
- Queues (-q option)
- Hosts (-m option)
- Project names (-P option)
- Limits (SLOTS, MEM, TMP, SWP, JOBS)
- Limit configuration (-c option). This is the same as `bresources` with no options.

Resources that have no configured limits or no limit usage are indicated by a dash (-). Limits are displayed in a USED/LIMIT format. For example, if a limit of 10 slots is configured and 3 slots are in use, then `blimits` displays the limit for SLOTS as 3/10.

Note that if there are no jobs running against resource allocation limits, LSF indicates that there is no information to be displayed:

```
No resource usage found.
```

If limits MEM, SWP, or TMP are configured as percentages, both the limit and the amount used are displayed in MB. For example, `lshosts` displays maxmem of 249 MB, and MEM is limited to 10% of available memory. If 10 MB out of 25 MB are used, `blimits` displays the limit for MEM as 10/25 (10 MB USED from a 25 MB LIMIT).

Limits are displayed for both the vertical tabular format and the horizontal format for Limit sections. If a vertical format Limit section has no name, `blimits` displays NONAME*nnn* under the NAME column for these limits, where the unnamed limits are numbered in the order the vertical-format Limit sections appear in the `lsb.resources` file.

If a resource consumer is configured as all, the limit usage for that consumer is indicated by a dash (-)

PER_HOST slot limits are not displayed. The `bhosts` commands displays these as MXJ limits.

When LSF adds more resources to a running resizable job, `blimits` displays the added resources. When LSF removes resources from a running resizable job, `blimits` displays the updated resources.

In MultiCluster, `blimits` returns the information about all limits in the local cluster.

Limit names and policies are set up by the LSF administrator. See `lsb.resources`(5) for more information.

Options

-c

Displays all resource configurations in `lsb.resources`. This is the same as `bresources` with no options.

-w

Displays resource allocation limits information in a wide format. Fields are displayed without truncation.

-n *limit_name ...*

Displays resource allocation limits the specified named Limit sections. If a list of limit sections is specified, Limit section names must be separated by spaces and enclosed in quotation marks (") or (’).

-m *host_name* | -m *host_group* | -m *cluster_name* ...

Displays resource allocation limits for the specified hosts. Do not use quotes when specifying multiple hosts.

To see the available hosts, use `bhosts`.

For host groups:

- If the limits are configured with HOSTS, the name of the host group is displayed.
- If the limits are configured with PER_HOST, the names of the hosts belonging to the group are displayed instead of the name of the host group.

Tip:

PER_HOST slot limits are not displayed. The bhosts command displays these as MXJ limits.

For a list of host groups see `bmgroup(1)`.

In MultiCluster, if a cluster name is specified, displays resource allocation limits in the specified cluster.

-P *project_name* ...

Displays resource allocation limits for the specified projects.

If a list of projects is specified, project names must be separated by spaces and enclosed in quotation marks (") or (’).

-q *queue_name* ...

Displays resource allocation limits for the specified queues.

The command `bqueues` returns a list of queues configured in the system, and information about the configurations of these queues.

In MultiCluster, you cannot specify remote queues.

-u *user_name* | -u *user_group* ...

Displays resource allocation limits for the specified users.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or (’). You can specify both user names and user IDs in the list of users.

If a user group is specified, displays the resource allocation limits that include that group in their configuration. For a list of user groups see `bugroup(1)`).

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

Configured limits and resource usage for built-in resources (slots, mem, tmp, and swp load indices, and running and suspended job limits) are displayed as INTERNAL RESOURCE LIMITS separately from custom external resources, which are shown as EXTERNAL RESOURCE LIMITS.

Output: Resource Consumers

`blimits` displays the following fields for resource consumers:

NAME

The name of the limit policy as specified by the Limit section NAME parameter.

USERS

List of user names or user groups on which the displayed limits are enforced, as specified by the Limit section parameters USERS or PER_USER.

User group names have a slash (/) added at the end of the group name. See `bugroup (1)`.

QUEUES

The name of the queue to which the limits apply, as specified by the Limit section parameters QUEUES or PER_QUEUES.

If the queue has been removed from the configuration, the queue name is displayed as lost_and_found. Use `bhist` to get the original queue name. Jobs in the `lost_and_found` queue remain pending until they are switched with the `bswitch` command into another queue.

In a MultiCluster resource leasing environment, jobs scheduled by the consumer cluster display the remote queue name in the format *queue_name@cluster_name*. By default, this field truncates at 10 characters, so you might not see the cluster name unless you use `-w` or `-l`.

HOSTS

List of hosts and host groups on which the displayed limits are enforced, as specified by the Limit section parameters HOSTS or PER_HOSTS.

Host group names have a slash (/) added at the end of the group name. See `bmgroup (1)`.

Tip:

PER_HOST slot limits are not displayed. The bhosts command displays these as MXJ limits.

PROJECTS

List of project names on which limits are enforced., as specified by the Limit section parameters PROJECTS or PER_PROJECT.

Output: Resource Limits

`blimits` displays resource allocation limits for the following resources:

SLOTS

Number of slots currently used and maximum number of slots configured for the limit policy, as specified by the Limit section SLOTS parameter.

MEM

Amount of memory currently used and maximum configured for the limit policy, as specified by the Limit section MEM parameter.

TMP

Amount of tmp space currently used and maximum amount of tmp space configured for the limit policy, as specified by the Limit section TMP parameter.

SWP

Amount of swap space currently used and maximum amount of swap space configured for the limit policy, as specified by the Limit section SWP parameter.

JOBS

Number of currently running and suspended jobs and the maximum number of jobs configured for the limit policy, as specified by the Limit section JOBS parameter.

Example

The following command displays limit configuration and dynamic usage information for project `proj1`:

```
blimits -P proj1
INTERNAL RESOURCE LIMITS:
NAME      USERS      QUEUES     HOSTS     PROJECTS    SLOTS    MEM    TMP    SWP    JOBS
limit1    user1           -     hostA        proj1     2/6     -      -      -      -
NONAME022    -            -     hostB  proj1 proj2     1/3     -      -      -      -

EXTERNAL RESOURCE LIMITS:
NAME      USERS      QUEUES     HOSTS     PROJECTS     tmp1
limit1    user1           -     hostA proj1      1/1
```

See also

bclusters, bhosts, bhist, bmgroup, bqueues, bugroup, `lsb.resources`

blinfo

displays static License Scheduler configuration information

Synopsis

blinfo -Lp | **-p** | **-D** | **-G** | **-P**

blinfo [**-a** [**-t** *token_name* | **"***token_name ...***"**]] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

blinfo -A [**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

blinfo -C [**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

blinfo [**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

blinfo [**-h** | **-V**]

Description

Displays different license configuration information, depending on the option selected.

By default, displays information about the distribution of licenses managed by License Scheduler.

Options

-A

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, shows the feature allocation by cluster locality.

You can optionally provide license token names.

-a

Shows all information, including information about non-shared licenses (NON_SHARED_DISTRIBUTION) and workload distribution (WORKLOAD_DISTRIBUTION).

You can optionally provide license token names.

`blinfo -a` does not display NON_SHARED information for hierarchical project group scheduling policies. Use `blinfo -G` to see hierarchical group configuration.

-C

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, shows the cluster locality information for the features.

You can optionally provide license token names.

-D

Lists the License Scheduler service domains and the corresponding FLEXnet license server hosts.

-G

Lists the hierarchical configuration information.

If PRIORITY is defined in the `ProjectGroup` Section of `lsf.licensescheduler`, this option also shows the priorities of each project.

-g *feature_group* ...

When FEATURE_GROUP is configured for a group of license features in `lsf.licensescheduler`, shows only information about the features configured in the FEATURE_LIST of specified feature groups. You can specify more than one feature group at one time.

When you specify feature names with -t, features in the feature list defined by -t and feature groups are both displayed.

Feature groups listed with -g but not defined in `lsf.licensescheduler` are ignored.

-Lp

Lists the active projects managed by License Scheduler.

-Lp only displays projects associated with configured features.

If PRIORITY is defined in the `Projects` Section of `lsf.licensescheduler`, this option also lists the priorities of each project.

-o alpha | total

Sorts license feature information alphabetically, by total licenses, or by available licenses.

- alpha: Features are listed in descending alphabetical order.
- total: Features are sorted by the descending order of the sum of licenses that are allocated to LSF workload from all the service domains configured to supply licenses to the feature. Licenses borrowed by non-LSF workload are included in this amount.

-P

When LS_FEATURE_PERCENTAGE=Y, lists the license ownership in percentage.

-p

Displays values of `lsf.licensescheduler` configuration parameters and `lsf.conf` parameters related to License Scheduler. This is useful for troubleshooting.

-t *token_name* | "*token_name* ..."

Only shows information about specified license tokens. Use spaces to separate multiple names, and enclose them in quotation marks.

-h

Prints command usage to `stderr` and exits.

-V

Prints the License Scheduler release version to `stderr` and exits.

Default output

Displays the following fields:

FEATURE

The license name. This becomes the license token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

SERVICE_DOMAIN

The name of the service domain that provided the license.

TOTAL

The total number of licenses managed by FLEXnet. This number comes from FLEXnet.

DISTRIBUTION

The distribution of the licenses among license projects in the format [*project_name*, *percentage*[/*number_licenses_owned*]]. This determines how many licenses a project is entitled to use when there is competition for licenses. The percentage is calculated from the share specified in the configuration file.

Allocation output (-A)

FEATURE

The license name. This becomes the license token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

PROJECT

The License Scheduler project name.

ALLOCATION

The percentage of shares assigned to each cluster for a feature and a project.

All output (-a)

Same as Default Output with NON_SHARED_DISTRIBUTION.

NON-SHARED_DISTRIBUTION

This column is displayed directly under DISTRIBUTION with the `-a` option. If there are non-shared licenses, then the non-shared license information is output in the following format: [*project_name*, *number_licenses_non_shared*]

If there are no non-shared licenses, then the following license information is output - (dash)

Cluster locality output (-C)

NAME

The license feature token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

FLEX_NAME

The actual FLEXnet feature name—the name used by FLEXnet to identify the type of license. May be different from the License Scheduler token name if a different FLEX_NAME is specified in `lsf.licensescheduler`.

CLUSTER_NAME

The name of the cluster the feature is assigned to.

FEATURE

The license feature name. This becomes the license token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

SERVICE_DOMAIN

The service domain name.

Service Domain Output (-D)

SERVICE_DOMAIN

The service domain name.

LIC_SERVERS

Names of FLEXnet license server hosts that belong the to service domain. Each host name is enclosed in parentheses, as shown:

(*port_number@host_name*)

Redundant hosts (that share the same FLEXnet license file) are grouped together as shown:

(*port_number@host_name port_number@host_name port_number@host_name*)

Hierarchical Output (-G)

The following fields describe the values of their corresponding configuration fields in the `ProjectGroup` Section of `lsf.licensescheduler`.

GROUP

The project names in the hierarchical grouping and its relationships. Each entry specifies the name of the hierarchical group and its members. The entry is enclosed in parentheses as shown:

(*group* (*member* ...))

SHARES

The shares assigned to the hierarchical group member projects.

OWNERSHIP

The number of licenses that each project owns.

LIMITS

The maximum number of licenses that the hierarchical group member project can use at any one time.

NON_SHARED

The number of licenses that the hierarchical group member projects use exclusively.

PRIORITY

The priority of the project if it is different from the default behavior. A larger number indicates a higher priority.

DESCRIPTION

The description of the project group.

Project Output (-Lp)

List of active License Scheduler projects.

-Lp only displays projects associated with configured features.

PROJECT

The project name.

PRIORITY

The priority of the project if it is different from the default behavior. A larger number indicates a higher priority.

DESCRIPTION

The description of the project.

Parameters Output (-p)

ADMIN

The License Scheduler administrator. Defined in `lsf.licensescheduler`.

DISTRIBUTION_POLICY_VIOLATION_ACTION

This parameter includes

- The interval (a multiple of LM_STAT_INVERVAL periods) at which License Scheduler checks for distribution policy violations, and
- The directory path and command that License Scheduler runs when reporting a violation

Defined in `lsf.licensescheduler`.

HOSTS

License Scheduler candidate hosts. Defined in `lsf.licensescheduler`.

LM_REMOVE_INTERVAL

Minimum time a job must have a license checked out before `lmremove` can remove the license. Defined in `lsf.licensescheduler`.

LM_STAT_INTERVAL

Time interval between calls that License Scheduler makes to collect license usage information from FLEXnet license management. Defined in `lsf.licensescheduler`.

LS_ENABLE_MAX_PREEMPT

Enables maximum preemption time checking for taskman jobs. Defined in `lsf.licensescheduler`.

LS_MAX_TASKMAN_PREEMPT

Maximum number of times `taskman` jobs can be preempted. Enabled by LS_ENABLE_MAX_PREEMPT. Defined in `lsf.licensescheduler`.

LS_MAX_TASKMAN_SESSIONS

Maximum number of `taskman` jobs that run simultaneously. Defined in `lsf.licensescheduler`.

LSF_LIC_SCHED_HOSTS

List of hosts that are candidate LSF License Scheduler hosts. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_REQUEUE

Specifies whether to requeue or suspend a job whose license is preempted by LSF License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE

Specifies whether to release the slot of a job that is suspended when its license is preempted by LSF License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_STOP

Specifies whether to use job controls to stop a job that is preempted. Defined in `lsf.conf`.

LSF_LICENSE_FILE

Location of the LSF license file, which includes License Scheduler keys. Defined in `lsf.conf`.

PORT

TCP listening port used by License Scheduler. Defined in `lsf.licensescheduler`.

Examples

`blinfo -a` displays both NON_SHARED_DISTRIBUTION and WORKLOAD_DISTRIBUTION information:

```
blinfo -a
FEATURE      SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1           LS              3      [p1, 50.0%] [p2, 50.0% / 2] NON_SHARED_DISTRIBUTION
                                    [p2, 2]
                                    WORKLOAD_DISTRIBUTION
                                    [LSF 66.7%, NON_LSF 33.3%]
```

`blinfo -a` does not display NON_SHARED_DISTRIBUTION, if the NON_SHARED_DISTRIBUTION is not defined:

blinfo -a

```
FEATURE      SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1           LS              0      [p1, 50.0%] [p2, 50.0%]
                                    WORKLOAD_DISTRIBUTION
                                    [LSF 66.7%, NON_LSF 33.3%]
g2           LS              0      [p1, 50.0%] [p2, 50.0%]
g33          WS              0      [p1, 50.0%] [p2, 50.0%]
```

`blinfo -a` does not display WORKLOAD_DISTRIBUTION, if the WORKLOAD_DISTRIBUTION is not defined:

blinfo -a

```
FEATURE      SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1           LS              3      [p1, 50.0%] [p2, 50.0% / 2]
                                    NON_SHARED_DISTRIBUTION
                                    [p2, 2]
```

Files

Reads `lsf.licensescheduler`

See also

blstat, blusers

blkill

terminates an interactive License Scheduler task

Synopsis

blkill [**-t** *seconds*] *task_ID*

blkill [**-h** | **-V**]

Description

Terminates a running or waiting interactive task in License Scheduler.

Users can kill their own tasks. You must be a License Scheduler administrator to terminate another user's task.

By default, `blkill` notifies the user and waits 30 seconds before killing the task.

Options

task_ID

Task ID of the task you want to kill.

-t *seconds*

Specify how many seconds to delay before killing the task. A value of 0 means to kill the task immediately (do not give the user any time to save work).

-h

Prints command usage to `stderr` and exits.

-V

Prints License Scheduler release version to `stderr` and exits.

blparams

displays information about configurable License Scheduler parameters defined in the files `lsf.licensescheduler` and `lsf.conf`

Synopsis

blparams [**-h** | **-V**]

Description

Displays the following parameter values:

ADMIN

The License Scheduler administrator. Defined in `lsf.licensescheduler`.

DISTRIBUTION_POLICY_VIOLATION_ACTION

This parameter includes

- The interval (a multiple of LM_STAT_INVERVAL periods) at which License Scheduler checks for distribution policy violations, and
- The directory path and command that License Scheduler runs when reporting a violation

Defined in `lsf.licensescheduler`.

HOSTS

License Scheduler candidate hosts. Defined in `lsf.licensescheduler`.

LM_REMOVE_INTERVAL

Minimum time a job must have a license checked out before `lmremove` can remove the license. Defined in `lsf.licensescheduler`.

LM_STAT_INTERVAL

Time interval between calls that License Scheduler makes to collect license usage information from FLEXnet license management. Defined in `lsf.licensescheduler`.

LS_DEBUG_BLD

Sets the debugging log class for the LSF License Scheduler bld daemon. Defined in `lsf.licensescheduler`.

Specifies the log class filtering to be applied to bld. Messages belonging to the specified log class are recorded. Not all debug message are controlled by log class.

LS_DEBUG_BLD sets the log class and is used in combination with MASK, which sets the log level. For example:

```
LS_LOG_MASK=LOG_DEBUG
LS_DEBUG_BLD="LC_TRACE"
```

You need to restart the `bld` daemon after setting LS_DEBUG_BLD for your changes to take effect.

Valid log classes are:

- LC_AUTH: Log authentication messages
- LC_COMM: Log communication messages
- LC_FLEX: Log everything related to FLEX_STAT or FLEX_EXEC Macrovision APIs
- LC_LICENSE: Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_PREEMPT: Log license preemption policy messages
- LC_RESREQ: Log resource requirement messages
- LC_TRACE: Log significant program walk steps
- LC_XDR: Log everything transferred by XDR

If you use the command `bladmin blddebug` to temporarily change this parameter without changing `lsf.licensescheduler`, you do not need to restart the daemons.

LS_ENABLE_MAX_PREEMPT

Enables maximum preemption time checking for taskman jobs. Defined in `lsf.licensescheduler`.

LS_LOG_MASK

Specifies the logging level of error messages for LSF License Scheduler daemons. If LS_LOG_MASK is not defined in `lsf.licensescheduler`, the value of LSF_LOG_MASK in `lsf.conf` is used. If neither LS_LOG_MASK nor LSF_LOG_MASK is defined, the default is LOG_WARNING.

For example:

```
LS_LOG_MASK=LOG_DEBUG
```

The log levels in order from highest to lowest are:

- LOG_WARNING
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

The most important License Scheduler log messages are at the LOG_WARNING level. Messages at the LOG_DEBUG level are only useful for debugging.

LS_MAX_TASKMAN_PREEMPT

Maximum number of times `taskman` jobs can be preempted. Enabled by LS_ENABLE_MAX_PREEMPT. Defined in `lsf.licensescheduler`.

LS_MAX_TASKMAN_SESSIONS

Maximum number of `taskman` jobs that run simultaneously. Defined in `lsf.licensescheduler`.

LS_PREEMPT_PEER

Enables bottom-up license token preemption in hierarchical project group configuration. License Scheduler attempts to preempt tokens from the closest projects in the hierarchy first. This balances token ownership from the bottom up.

Defined in `lsf.licensescheduler`.

LSF_LIC_SCHED_HOSTS

List of hosts that are candidate LSF License Scheduler hosts. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_REQUEUE

Specifies whether to requeue or suspend a job whose license is preempted by LSF License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE

Specifies whether to release the slot of a job that is suspended when its license is preempted by LSF License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_STOP

Specifies whether to use job controls to stop a job that is preempted. Defined in `lsf.conf`.

LSF_LICENSE_FILE

Location of the LSF license file, which includes License Scheduler keys. Defined in `lsf.conf`.

PORT

TCP listening port used by License Scheduler. Defined in `lsf.licensescheduler`.

Options

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

lsf.licensescheduler, lsf.conf

blstat

displays dynamic license information

Synopsis

blstat [**-a**] [**-c** *token_name*] [**-G**] [**-s**] [**-S**] [**-D** *service_domain_name* | **"***service_domain_name ...***"**] [**-Lp** *ls_project_name* | **"***ls_project_name ...***"**] [**-P**][**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total** | **avail**] [**-g** **"***feature_group ...***"**]

blstat [**-h** | **-V**]

Description

Displays license usage statistics.

By default, shows information about all licenses and all clusters.

Options

-a

Displays each project group's accumulated value of licenses. The license token dispatching order is based on the sort order, which is based on the scaled accumulate value of each project. The lower the value, the sooner the license token is dispatched to that project.

-c *token_name*

Displays cross cluster information for tokens, sorted by the value of SCALED_ACUM. The first cluster listed receives tokens first.

Information displayed includes token usage, reserved tokens, free tokens, demand for tokens, accumulated value of tokens, and scaled accumulate value of tokens in each cluster.

-G

Displays dynamic hierarchical license information.

`blstat -G` also works with the -t option to only display hierarchical information for the specified feature names.

-S

Displays information on the license server associated with license features.

-s

Displays license usage of the LSF and non-LSF workloads. Workload distributions are defined by WORKLOAD_DISTRIBUTION in `lsf.licensescheduler`. If there are any distribution policy violations, `blstat` marks these with an asterisk (*) at the beginning of the line.

-D *service_domain_name* | "*service_domain_name* ..."

Only shows information about specified service domains. Use spaces to separate multiple names, and enclose them in quotation marks.

-g *feature_group* ...

When FEATURE_GROUP is configured for a group of license features in `lsf.licensescheduler`, shows only information about the features configured in the FEATURE_LIST of specified feature groups. You can specify more than one feature group at a time.

When you specify feature names with -t, features in the FEATURE_LIST defined by -t and feature groups are both displayed.

Feature groups listed with -g but not defined in `lsf.licensescheduler` are ignored.

-Lp *ls_project_name* | "*ls_project_name* ..."

Shows project description for specified projects (non-hierarchical). Use spaces to separate multiple names and enclose them in quotation marks.

-o alpha | total | avail

Sorts license feature information alphabetically, by total licenses, or by available licenses.

- alpha: Features are listed in descending alphabetical order.
- total: Features are sorted by the descending order of the sum of licenses that are allocated to LSF workload from all the service domains configured to supply licenses to the feature. Licenses borrowed by non-LSF workload are not included in this amount.
- avail: Features are sorted by descending order of licenses available, including free tokens.

-P

Displays percentage values for INUSE and RESERVE. The percentage value represents the number of tokens this project has used and reserved compared to total number of licenses.

-t *token_name* | "*token_name* ..."

Only shows information about specified licenses. Use spaces to separate multiple names, and enclose them in quotation marks.

-h

Prints command usage to `stderr` and exits.

-V

Prints the release version to `stderr` and exits.

Output

Information is organized first by license feature, then by service domain. For each combination of license and service domain, License Scheduler displays a line of summary information followed by rows of license project information (one row for each license project configured to use the license).

In each group of statistics, numbers and percentages refer only to licenses of the specified license feature that can be checked out from FLEXnet license server hosts in the specified service domain.

Summary output

FEATURE

The license name. (This appears only once for each feature.)

SERVICE_DOMAIN

The name of the service domain that provided the license.

TOTAL_INUSE

The number of licenses in use by License Scheduler projects. (Licenses in use have been checked out from the FLEXnet license manager.)

TOTAL_RESERVE

The number of licenses reserved for License Scheduler projects. (Licenses that are reserved and have not been checked out from the FLEXnet license manager.)

TOTAL_FREE

The number of free licenses that are available to License Scheduler projects. (Licenses that are not reserved or in use.)

OTHERS

The number of licenses checked out by users who are not submitting their jobs to License Scheduler projects.

By default, these licenses are not being managed by License Scheduler policies.

To enforce license distribution policies for these license features, configure ENABLE_DYNAMIC_RUSAGE=Y in the feature section for those features in `lsf.licensescheduler`.

Workload output

LSF_USE

The total number of licenses in use by License Scheduler projects in the LSF workload.

LSF_DESERVE

The total number of licenses assigned to License Scheduler projects in the LSF workload.

LSF_FREE

The total number of free licenses available to License Scheduler projects in the LSF workload.

NON_LSF_USE

The total number of licenses in use by projects in the non-LSF workload.

NON_LSF_DESERVE

The total number of licenses assigned to projects in the non-LSF workload.

NON_LSF_FREE

The total number of free licenses available to projects in the non-LSF workload.

Project output

For each project that is configured to use the license, `blstat` displays the following information.

PROJECT

The License Scheduler project name.

SHARE

The percentage of licenses assigned to the license project by the License Scheduler administrator. This determines how many licenses the project is entitled to when there is competition for licenses. This information is static.

The percentage is calculated to one decimal place using the share assignment in `lsf.licensescheduler`.

LIMITS

The maximum number of licenses that the hierarchical group member project can use at any one time.

OWN

Numeric value indicating the number of tokens owned by each project.

INUSE

The number of licenses in use by the license project. (Licenses in use have been checked out from the FLEXnet license manager.)

RESERVE

The number of licenses reserved for the license project. (The corresponding job has started to run, but has not yet checked out its license from the FLEXnet license manager.)

FREE

The number of licenses the license project has free. (The license tokens have been allocated to the license project by License Scheduler, but the licenses are not reserved and have not yet been checked out from the FLEXnet license manager.)

DEMAND

Numeric value indicating the number of tokens required by each project.

NON_SHARED

The number of non-shared licenses belonging to the license project. (The license tokens allocated to non-shared distribution are scheduled before the tokens allocated to shared distribution.)

DESCRIPTION

Description of the project.

ACUM_USE

The number of tokens accumulated by each consumer at runtime. It is the number of licenses assigned to a given consumer for a specific feature.

SCALED_ACUM

The number of tokens accumulated by each consumer at runtime divided by the SHARE value. License Scheduler uses this value to schedule the tokens for each project.

Hierarchical output

SHARE_INFO_FOR

The root member and name of the hierarchical group. The project information displayed after this title shows the information specific to this particular hierarchical group. If this root member is itself a member of another hierarchical group, the relationship is displayed as follows:

/*root_name*/*member_name*/...

PROJECT/GROUP

The members of the hierarchical group, listed by its group or project name.

Viewing license feature locality

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blstat` shows the cluster locality information for the license features. For example, with a group distribution configuration `blstat` shows the locality of the hspice feature configured for various sites:

blstat

```
FEATURE: hspice
 SERVICE_DOMAIN: SD3 SD4
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 22   OTHERS: 0
  PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                     50.0 %   3     1    0        0     11
  Lp2                     50.0 %   1     3    0        0     11
FEATURE: hspice@clusterA
 SERVICE_DOMAIN: SD1
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 25   OTHERS: 0
  PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                     50.0 %   4     0    0      12       3
  Lp2                     50.0 %   5     0    0      13       1
FEATURE: hspice@siteB
 SERVICE_DOMAIN: SD2
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 65   OTHERS: 0
  PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                     50.0 %   4     0    0      32       2
  Lp2                     50.0 %   5     0    0      33       6
```

See also

blhosts, blinfo

bltasks

displays License Scheduler interactive task information

Synopsis

bltasks [**-l**] [*task_ID*]

bltasks [**-l**] [**-p** | **-r** | **-w**] [**-Lp "***ls_project_name...***"**] [**-m "***host_name...***"**] [**-t "***terminal_name...***"**] [**-u "***user_name...***"**]

bltasks [| **-h** | **-V**]

Description

Displays current information about interactive tasks managed by License Scheduler (submitted using `taskman`).

By default, displays information about all tasks.

Options

task_ID

Only displays information about the specified task.

-l

Long format. Displays detailed information for each task in a multi-line format.

-p

Only displays information about tasks with `PREEMPTED` status.

Cannot be used with -r or -w.

-r

Only displays information about tasks with RUN status.

Cannot be used with -p or -w.

-w

Only displays information about tasks with WAIT status.

Cannot be used with -p or -r.

-Lp "*ls_project_name*..."

Only displays information about tasks associated with the specified projects.

-m "*host_name*..."

Only displays information about tasks submitted from the specified hosts.

-t "*terminal_name*..."

Only displays information about tasks submitted from the specified terminals.

-u "*user_name*..."

Only displays information about tasks submitted by the specified users.

-h

Prints command usage to `stderr` and exits.

-V

Prints License Scheduler release version to `stderr` and exits.

Default Output

Displays the short format with the following information:

TID

Task ID that License Scheduler assigned to the task.

USER

The user who submitted the task.

STAT

The current status of the task.

- RUN: Task is running.
- WAIT: Task has not yet started.
- PREEMPT: Task has been preempted and currently has no license token.

HOST

The name of host from which the task was submitted.

PROJECT

The name of the project to which the task belongs.

FEATURES

Name of the License Scheduler token.

CONNECT TIME

The submission time of the task.

Output for -l Option

Displays detailed information for each task in multi-line format. If the task is in WAIT status, `bltasks` displays "The application manager is waiting for a token to start" and the resource requirement. Otherwise, the current resource usage of task is displayed as follows:

TERMINAL

The terminal the task is using.

PGID

UNIX process group ID.

CPU

The total accumulated CPU time of all processes in a task, in seconds.

MEM

Total resident memory usage of all processes in a task, in KB.

SWAP

Total virtual memory usage of all processes in a task, in KB.

Keyboard idle since

Time at which the task became idle.

RES_REQ

The resource requirement of the task.

Command line

The command the License Scheduler task manager is executing.

Output for -g Option

Displays every task group and information about the tasks that belong to the group. The first 4 fields - POLICY, USER, HOST, and TERMINAL - define a unique task group. A dash (-) in one of these fields means that the group includes all tasks, no matter what the value of the field is.

POLICY

License Scheduler token name. All tasks in the group request this type of license.

USER

License Scheduler user name. All tasks in the group are submitted by this user.

HOST

Host name. All tasks in the group are submitted from this host.

TERMINAL

Terminal name. All tasks in the group are submitted using this terminal.

TASK_NAME

The task name.

RUN

If 0, the group does not have a License Scheduler token at this time.

If 1 or greater, the group has a token and all tasks in the group are running.

TOTAL

Number of tasks in the group.

CONNECT_TIME

Time and date that the group was created (and requested a License Scheduler token). If all tasks in a group finish, the group releases the License Scheduler token, and the next task submitted re-creates the group.

blusers

displays license usage information

Synopsis

blusers [**-J** | **-l** | **-P -j** *job_ID* **-u** *user_name* **-m** *host_name* | **-P -c** *cluster_name* **-j** *job_ID* **-u** *user_name* **-m** *host_name*]

blusers [**-h** | **-V**]

Description

By default, displays summarized information about usage of licenses.

Options

-J

Displays detailed license resource request information about each job.

-l

Long format. Displays additional license usage information.

-P -j *job_ID* -u *user_name* -m *host_name*

-P -c *cluster_name* -j *job_ID* -u *user_name* -m *host_name*

This string of options is designed to be used in a customized preemption script. To identify a job, specify the LSF job ID, the user name, and the name of the host where the job is running.

(If the job is an interactive task submitted using `taskman`, do not specify *-c cluster_name*.)

You see the display terminal used by the job, the licenses it has checked out, and the license servers that provided the licenses. There is one line of output for each license feature from each FLEXnet license server, in the format:

port_number@host_name token_name user_name host_name display

-h

Prints command usage to `stderr` and exits.

-V

Prints License Scheduler release version to `stderr` and exits.

Default Output

FEATURE

The license name. This becomes the license token name.

SERVICE_DOMAIN

The name of the service domain that provided the license.

USER

The name of the user who submitted the jobs.

HOST

The name of the host where jobs have started.

NLICS

The number of licenses checked out from FLEXnet.

NTASKS

The number of running tasks using these licenses.

-J Output

Displays the following summary information for each job:

JOBID

The job ID assigned by LSF.

USER

The name of the user who submitted the job.

HOST

The name of the host where the job has been started.

PROJECT

The name of the license project that the job is associated with.

CLUSTER

The name of the LSF cluster that the job is associated with. Displays “-” for an interactive job.

START_TIME

The job start time.

Displays the following information for each license in use by the job:

RESOURCE

The name of the license requested by the job.

RUSAGE

The number of licenses requested by the job.

SERVICE_DOMAIN

The name of the service domain that provided the license.

The keyword UNKNOWN means the job requested a license from License Scheduler but has not checked out the license from FLEXnet.

Long Output (-l)

Displays the default output and the following additional information for each job:

OTHERS

License usage for non-managed or non-LSF workload.

DISPLAYS

Terminal display associated with the license feature.

Viewing license feature locality

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blusers` shows the cluster locality information for the license features. For example:

blusers

```
FEATURE           SERVICE_DOMAIN  USER     HOST       NLICS    NTASKS
hspice@clusterA   SD1             user1    host1      1        1
hspice@siteB      SD2             user2    host2      1        1
```

Examples

blusers -l

```
FEATURE  SERVICE_DOMAIN  USER   HOST    NLICS   NTASKS OTHERS  DISPLAYS
feat1    LanServer       user1  hostA   1       1      0     (/dev/tty)
```

blusers -J

```
JOBID   USER      HOST      PROJECT             CLUSTER        START_TIME
553     user1     hostA     p3                  cluster1       Oct  5 15:47:14
RESOURCE        RUSAGE        SERVICE_DOMAIN
p1_f1           1             app_1
```

See also

blhosts, blinfo, blstat

bmgroup

displays information about host groups and compute units

Synopsis

bmgroup [**-r**] [**-l**] [**-w**] [**-cu**] [*group...*]

bmgroup [**-h** | **-V**]

Description

Displays compute units, host groups, host names, and administrators for each group or unit.

By default, displays information about all compute units, host partitions, and host groups including host groups created for EGO-enabled SLA scheduling.

Host groups for EGO-enabled SLA

When hosts are allocated to an EGO-enabled SLA, they are dynamically added to a host group created by the SLA. The name of the host group is _sla_*sla_name*, where *sla_name* is the name of the EGO-enabled SLA defined in `lsb.serviceclasses` or in ENABLE_DEFAULT_EGO_SLA in `lsb.params`. One of the hosts in the host group has the name _virtual.

When the host is released to EGO, the entry is removed from the host group. `bmgroup` displays the hosts allocated by EGO to the host group created by the SLA.

Options

-l

Displays static and dynamic host group members. A '+' sign before the host name indicates that the host is dynamic and is currently a member of the compute unit, host partitions, or host group. A '-' sign before the host name indicates that the host is currently not an LSF host but is a member of the dynamic group, partition, or unit.

Also identifies condensed compute units, host partitions, or host groups as defined by CONDENSE in the `HostGroup` or `ComputeUnit` section of `lsb.hosts`.

-r

Expands host groups, host partitions, and compute units recursively. The expanded list contains only host names; it does not contain the names of subgroups. Duplicate names are listed only once.

-w

Wide format. Displays names without truncating fields.

-cu

Displays compute unit information. Use with [*cu_name*] to display only the specified compute unit.

group...

Only displays information about the specified groups (host groups, or compute unit with -cu). Do not use quotes when specifying multiple groups.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

In the list of hosts, a name followed by a slash (/) indicates a subgroup.

Files

Host groups, compute units, and host partitions are configured in the configuration file `lsb.hosts`(5). Compute unit types are defined by COMPUTE_UNIT_TYPES in `lsb.params(5)`.

See also

lsb.hosts(5), lsb.params(5), bugroup(1), bhosts(1)

bmig

migrates checkpointable or rerunnable jobs

Synopsis

bmig [**-f**] [*job_ID* | "*job_ID*[*index_list*]"] ...

bmig [**-f**] [**-J** *job_name*] [**-m** "*host_name* ..." | **-m** "*host_group* ..."] [**-u** *user_name* | **-u** *user_group* | **-u all**] [**0**]

bmig [**-h** | **-V**]

Description

Migrates one or more of your checkpointable or rerunnable jobs to a different host. You can migrate only running or suspended jobs; you cannot migrate pending jobs. Members of a chunk job in the WAIT state can be migrated; LSF removes waiting jobs from the job chunk and changes their original dispatch sequence.

By default, migrates the most recently submitted job, or the most recently submitted job that also satisfies other specified options (-u and -J). Specify 0 (zero) to migrate multiple jobs. Only LSF administrators and root can migrate jobs submitted by other users. Both the original and the new hosts must:

- Be binary compatible
- Run the same dot version of the operating system for predictable results
- Have network connectivity and read/execute permissions to the checkpoint and restart executables (in LSF_SERVERDIR by default)
- Have network connectivity and read/write permissions to the checkpoint directory and the checkpoint file
- Have access to all files open during job execution so that LSF can locate them using an absolute path name

When you migrate a checkpointable job, LSF checkpoints and kills the job and then restarts the job on the next available host. If checkpoint fails, the job continues to run on the original host. If you issue the `bmig` command while a job is being checkpointed—for example, with periodic checkpointing enabled—LSF ignores the migration request.

When you migrate a rerunnable job, LSF kills the job and then restarts it from the beginning on the next available host. LSF sets the environment variable LSB_RESTART to Y when a migrating job restarts or reruns.

Note:

The user does not receive notification when LSF kills a checkpointable or rerunnable job as part of job migration.

In a MultiCluster environment, you must use `brun` rather than `bmig` to move a job to another host.

When absolute job priority scheduling (APS) is configured in the queue, LSF always schedules migrated jobs before pending jobs. For migrated jobs, LSF keeps the existing job priority. If LSB_REQUEUE_TO_BOTTOM and LSB_MIG2PEND are configured in `lsf.conf`, the migrated jobs keep their APS information, and the migrated jobs compete with other pending jobs based on the APS value. If you want to reset the APS value, you must use `brequeue` instead of `bmig`.

Options

-f

Forces a checkpointable job to be checkpointed and migrated, even if non-checkpointable conditions exist within the operating system environment.

***job_ID* | "*job_ID[index_list]*" | 0**

Migrates jobs with the specified job IDs. LSF ignores the -J and -u options.

If you specify a job ID of 0 (zero), LSF ignores all other job IDs and migrates all jobs that satisfy the -J and -u options.

If you do not specify a job ID, LSF migrates the most recently submitted job that satisfies the -J and -u options.

-J *job_name*

Migrates the job with the specified name. Ignored if a job ID other than 0 (zero) is specified.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-m "*host_name* ..." | -m "*host_group* ..."

Migrates jobs to the specified hosts.

This option cannot be used on a MultiCluster job; `bmig` can only restart or rerun the job on the original host.

-u "*user_name*" | -u "*user_group*" | -u all

Migrates only those jobs submitted by the specified users. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME\user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME\\user_name*) in a UNIX command line.

If you specify the reserved user name `all`, LSF migrates jobs submitted by all users. Ignored if a job ID other than 0 (zero) is specified.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

bsub, brestart, bchkpnt, bjobs, bqueues, bhosts, bugroup, mbatchd, `lsb.queues`, kill

bmod

modifies job submission options of a job

Synopsis

bmod [*bsub_options*] [*job_ID* | **"***job_ID***[***index***]"**]

bmod [**-g** *job_group_name* | **-gn**] [*job_ID*]

bmod [**-sla** *service_class_name* | **-slan**] [*job_ID*]

bmod [**-aps "system=***value***"** | **"admin=***value***"** | **-apsn**] [*job_ID*]

bmod [**-h** | **-V**]

Option List

[**-ar** | **-arn**]

[**-B** | **-Bn**]

[**-N** | **-Nn**]

[**-r** | **-rn**]

[**-ul** | **-uln**]

[**-x** | **-xn**]

[**-a esub_application**]

[**-app** *application_profile_name* | **-appn**]

[**-aps "system=***value***"** | **"admin=***value***"** | **-apsn**]

[**-b** *begin_time* | **-bn**]

[**-C** *core_limit* | -**Cn**]

[**-c** [*hour***:**]*minute*[**/***host_name* | **/***host_model*] | **-cn**]

[**-cwd "***current_working_directory***"** | **-cwdn**]

[**-D** *data_limit* | **-Dn**]

[**-E "***pre_exec_command* [*argument* ...]**"** | **-En**]

[**-Ep "***post_exec_command* [*argument* ...]**"** | **-Epn**]

[**-e** *err_file* | **-en**]

[**-eo** *err_file* | **-en**]

[**-ext**[**sched**] "*external_scheduler_options***"**]

[**-F** *file_limit* | **-Fn**]

[**-f "***local_file op* [*remote_file*]**"** ... | **-fn**]

[**-G** *user_group* | **-Gn**]

[**-g** *job_group_name* | **-gn**]

[**-i** *input_file* | **-in** | **-is** *input_file* | **-isn**]

[**-J** *job_name* | **-J "%***job_limit***"** | **-Jn**]

-k "*checkpoint_dir* [**init=***initial_checkpoint_period*] [*checkpoint_period*]**"** | **-kn**]

[**-L** *login_shell* | **-Ln**]

[**-Lp** *ls_project_name* | **-Lpn**]

[**-M** *mem_limit* | **-Mn**]

[**-m** *"host_name*[*@cluster_name*][[**!**] | **+**[*pref_level*]] | *host_group*[[**!**] | **+**[*pref_level*]| *compute_unit*[[**!**] | **+** [*pref_level*]] ...**"** | **-mn**]

[**-mig** *migration_threshold* | **-mign**]

[**-n** *num_processors* | **-nn**]

[**-o** *out_file* | **-on**]

[**-oo** *out_file* | **-oon**]

[**-P** *project_name* | **-Pn**]

[**-p** *process_limit* | **-pn**]

[**-Q "**[*exit_code …*] [**EXCLUDE(***exit_code …***)]"**]

[**-q "***queue_name ...***"** | **-qn**]

[**-R "***res_req***"** [**-R "***res_req***"** …] | **-Rn**]

[**-rnc** *resize_notification_cmd* | **-rncn**]

[**-S** *stack_limit* | **-Sn**]

[**-s** *signal* | **-sn**]

[**-sla** *service_class_name* | **-slan**]

[**-sp** *priority* | **-spn**]

[**-T** *thread_limit* | **-Tn**]

[**-t** *term_time* | **-tn**]

[**-U** *reservation_ID* | **-Un**]

[**-u** *mail_user* | **-un**]

[**-v** *swap_limit* | **-vn**]

[**-W** [*hour***:**]*minute*[**/***host_name* | **/***host_model*] | **-Wn**]

[**-We** [*hour***:**]*minute*[**/***host_name* | **/***host_model*] | **-Wep** [*value*] | **-We+** [*hour***:**]*minute*] | **-Wen**]

[**-w '***dependency_expression***'** | **-wn**]

[**-wa '**[*signal* | *command* | **CHKPNT**]**'** | **-wan**]

[**-wt '***job_warning_time***'** | **-wtn**]

[**-Z "***new_command***"** | **-Zs "***new_command***"** | **-Zsn**]

[*job_ID* | **"***job_ID***[***index***]"**]

Description

Modifies the options of a previously submitted job. See `bsub` for complete descriptions of job submission options you can modify with `bmod`.

Only the owner of the job, or LSF administrators, can modify the options of a job.

All options specified at submission time may be changed. The value for each option may be overridden with a new value by specifying the option as in `bsub`. To reset an option to its default value, use the option string followed by 'n'. Do not specify an option value when resetting an option.

The -i, -in, and -Z options have counterparts that support spooling of input and job command files (-is, -isn, -Zs, and -Zsn).

Options related to file names and job spooling directories support paths that contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

Options related to command names and job names can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

You can modify all options of a pending job, even if the corresponding `bsub` option was not specified.

Modifying a job that is pending in a chunk job queue (CHUNK_JOB_SIZE) removes the job from the chunk to be scheduled later.

Like `bsub`, `bmod` calls the master `esub` (`mesub`), which invokes any mandatory `esub` executables configured by an LSF administrator, and any executable named `esub` (without *.application*) if it exists in LSF_SERVERDIR. Only `esub` executables invoked by `bsub` can change the job environment on the submission host. An `esub` invoked by `bmod` cannot change the job environment.

-b modifies the job begin time. If the year field is specified and the specified time is in the past, the start time condition is considered reached and LSF dispatches the job if slots are available.

-t modifies job termination time. If the year field is specified and the specified time is in the past, the job modification request is rejected.

-cwdn sets the current working directory for the job to the directory where `bmod` is running.

-Epn cancels the setting of job-level post-execution commands. The job-level post-execution commands do not run. Application-level post-execution commands run if they exist.

For resizable jobs, `bmod -R "rusage[mem | swp]"` only affects the resize allocation request if the job has not been dispatched.

-m modifies the first execution host list. When used with a compound resource requirement, the first host allocated must satisfy the simple resource requirement string appearing first in the compound resource requirement.

-rn resets the rerunnable job setting specified by `bsub –rn` or `bsub -r.` The application profile and queue level rerunnable job setting if any is used. bmod -rn does not disable or override job rerun if the job was submitted to a rerunnable queue or application profile with job rerun configured. `bmod –rn` is different from `bsub -rn`, which does override the application profile and queue level rerunnable job setting.

-uln sets the user shell limits for pending jobs to their default values. -uln is not supported on Windows.

-Wen cancels the estimated job runtime. The runtime estimate does not take effect for the job.

-Q does not affect running jobs. For rerunnable and requeue jobs, -Q affects the next run.

Modifying running jobs

By default, you can modify resource requirements for running jobs (-R "*res_req*" except -R "cu[*cu_string*]") and the estimated running time for running or suspended jobs (`-We`, `-We+`, `-Wep`). To modify additional job options for running jobs, define LSB_MOD_ALL_JOBS=Y in `lsf.conf`.

When LSB_MOD_ALL_JOBS=Y is set, the following are the only `bmod` options that are valid for running jobs. You cannot make any other modifications after a job has been dispatched.

- CPU limit (-c [*hour*:]*minute*[/*host_name* | /*host_model*])
- Memory limit (-M *mem_limit*)
- Rerunnable jobs (-r | -rn)
- Resource requirements (-R "*res_req*" except -R "cu[*cu_string*]")
- Run limit (-W *run_limit*[/*host_name* | /*host_model*])
- Standard output (`stdout`) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-o *output_file*)
- Standard error (`stderr`) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-e *error_file*)
- Overwrite standard output (`stdout`) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-oo *output_file*)
- Overwrite standard error (`stderr`) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-eo *error_file*)

Modified resource usage limits cannot exceed limits defined in the queue.

To modify the CPU limit or the memory limit of running jobs, the parameters LSB_JOB_CPULIMIT=Y and LSB_JOB_MEMLIMIT=Y must be defined in `lsf.conf`.

If you want to specify array dependency by array name, set JOB_DEP_LAST_SUB in `lsb.params`. If you do not have this parameter set, the job is rejected if one of your previous arrays has the same name but a different index.

By default, options for the following resource usage limits are specified in KB:

- Core limit (`-C`)
- Memory limit (`-M`)
- Stack limit (`-S`)
- Swap limit (`-v`)

Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a different unit for the limit (MB, GB, TB, PB, or EB).

Modifying resource requirements

The -R option of `bmod` completely replaces any previous resource requirement specification. It does not add the modification to the existing specification. For example, if you submit a job with

```
bsub -R "rusage[res1=1]"
```

then modify it with

```
bmod -R "rusage[res2=1]"
```

the new resource usage requirement for the job is `[res2=1]`, not `[res1=1; res2=1]`.

`bmod` does not support the OR (||) operator on the -R option.

To remove all of the string input specified using the `bsub` command, use the -Rn option.

Modifying the estimated run time of jobs

The following options allow you to modify a job's estimated run time:

- `-We` [*hour*:]*minute*[*/host_name* | */host_model*]: Sets an estimated run time. Specifying a host or host model normalizes the time with the CPU factor (time/CPU factor) of the host or model.
- `-We+` [*hour*:]*minute*]: Sets an estimated run time that is the value you specify added to the accumulated run time. For example, if you specify **-We+ 30** and the job has already run for 60 minutes, the new estimated run time is now 90 minutes.

  Specifying a host or host model normalizes the time with the CPU factor (time/CPU factor) of the host or model.
- `-Wep` [*value*]: Sets an estimated run time that is the percentage of job completion that you specify added to the accumulated run time. For example, if you specify **-Wep+ 25** (meaning that the job is 25% complete) and the job has already run for 60 minutes, the new estimated run time is now 240 minutes.

  The range of valid values is greater than 0 and less than or equal to 100. Two digits after decimal are supported.

  Specifying a host or host model normalizes the time with the CPU factor of the host or model (time/CPU factor).

Modifying job groups

Use the -g option of `bmod` and specify a job group path to move a job or a job array from one job group to another. For example:

```
bmod -g /risk_group/portfolio2/monthly 105
```

moves job 105 to job group `/risk_group/portfolio2/monthly`.

Like `bsub -g`, if the job group does not exist, LSF creates it.

`bmod -g` cannot be combined with other `bmod` options. It can only operate on pending jobs. It cannot operate on running or finished jobs.

You can modify your own job groups and job groups that other users create under your job groups. LSF administrators can modify job groups of all users.

You cannot move job array elements from one job group to another, only entire job arrays. If any job array elements in a job array are running, you cannot move the job array to another group. A job array can only belong to one job group at a time.

You cannot modify the job group of a job attached to a service class.

Modifying jobs in service classes

The `-sla` option modifies a job by attaching it to the specified service class. The `-slan` option detaches the specified job from a service class. If the service class does not exist, the job is not modified. For example:

```
bmod -sla Kyuquot 2307
```

attaches job 2307 to the service class `Kyuquot`.

```
bmod -slan 2307
```

detaches job 2307 from the service class `Kyuquot`. If a default SLA is configured in `lsb.params`, the job is moved to the default service class.

You cannot

- Use -sla with other `bmod` options
- Move job array elements from one service class to another, only entire job arrays
- Modify the service class of job already attached to a job group. Use `bsla` to display the configuration properties of service classes configured in `lsb.serviceclasses`, the default SLA configured in `lsb.params`, and dynamic information about the state of each service class.

If a default SLA is configured in `lsb.params`, `bmod -slan` moves the job to the default SLA. If the job is already attached to the default SLA, `bmod -slan` has no effect on that job.

Modifying jobs associated with application profiles

The -app option modifies a job by associating it to the specified application profile. The -appn option dissociates the specified job from its application profile. If the application profile does not exist, the job is not modified.

You can only modify the application profile for pending jobs. For example:

bmod -app fluent 2308

associates job 2308 with the application profile `fluent`.

bmod -appn 2308

dissociates job 2308 from the service class `fluent`.

Use `bapp` to display the properties of application profiles configured in `LSB_CONFDIR/cluster_name/configdir/lsb.applications`.

Modifying absolute priority scheduling options

Administrators can use `bmod -aps` to adjust the APS value for pending jobs. `bmod -apsn` cancels previous `bmod -aps` settings. You cannot combing `bmod -aps` with other `bmod` options.

You can only change the APS value for pending resizable jobs.

-aps "system=*value*" *job_ID*

Set a static non-zero APS value of a pending job. Setting a system APS value overrides any calculated APS value for the job. The system APS value cannot be applied to running jobs.

-aps "admin=*value*" *job_ID*

Set a non-zero ADMIN factor value for a pending job. The ADMIN factor adjusts the calculated APS value higher or lower. A negative admin value is lowers the calculated APS value, and a positive value raises the calculated APS value relative to other pending jobs in the APS queue.

You cannot configure APS weight, limit, or grace period for the ADMIN factor. The ADMIN factor takes effect as soon as it is set.

-apsn

Run `bmod -apsn` to cancel previous `bmod -aps` settings. You cannot apply `bmod -apsn` on running jobs in an APS queue. An error is issued if the job has no system APS priority or ADMIN factor set.

Modifying resizable jobs

Use the -rnc and -ar options to modify the autoresizable attribute or resize notification command for resizable jobs. You can only modify the autoresizable attribute for pending jobs (PSUSP or PEND). You can only modify the resize notification command for unfinished jobs (not DONE or EXIT jobs).

-rnc *resize_notification_cmd*

Specify the name of an executable to be invoked on the first execution host when the job allocation has been modified (both shrink and grow). `bmod -rnc` overrides any notification command specified in the application profile.

-rncn

Remove the notification command setting from the job.

-ar

Specify that the job is autoresizable.

-arn

Remove job-level autoresizable attribute from the job.

Options

***job_ID* | "*job_ID*[*index*]"**

Modifies jobs with the specified job ID.

Modifies job array elements specified by "*job_ID*[*index*]".

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Limitations

Modifying remote running jobs in a MultiCluster environment is not supported.

If you do not specify -e or -eo before the job is dispatched, you cannot modify the name of job error file for a running job. Modifying the job output options of remote running jobs is not supported.

See also

bsub

bparams

displays information about configurable system parameters in `lsb.params`

Synopsis

bparams [**-a**] [**-l**]

bparams [**-h** | **-V**]

Description

Displays the following parameter values:

- Default Queues
- Job Dispatch Interval
- Job Checking Interval
- Job Accepting Interval

Options

-a

All format. Displays all the configurable parameters set in `lsb.params`.

-l

Long format. Displays detailed information about all the configurable parameters in `lsb.params`.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

`lsb.params`

bpeek

displays the `stdout` and `stderr` output of an unfinished job

Synopsis

bpeek [**-f**] [**-q** *queue_name* | **-m** *host_name* | **-J** *job_name* | *job_ID* | **"***job_ID***[***index_list***]"**]

bpeek [**-h** | **-V**]

Description

Displays the standard output and standard error output that have been produced by one of your unfinished jobs, up to the time that this command is invoked.

By default, displays the output using the command `cat`.

This command is useful for monitoring the progress of a job and identifying errors. If errors are observed, valuable user time and system resources can be saved by terminating an erroneous job.

Options

-f

Displays the output of the job using the command `tail -f`.

-q *queue_name*

Operates on your most recently submitted job in the specified queue.

-m *host_name*

Operates on your most recently submitted job that has been dispatched to the specified host.

-J *job_name*

Operates on your most recently submitted job that has the specified job name.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

***job_ID* | "*job_ID[index_list]*"**

Operates on the specified job.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

`cat`, `tail`, bsub, bjobs, bhist, bhosts, bqueues

bpost

sends external status messages and attaches data files to a job

Synopsis

bpost [**-i** *message_index*] [**-d** "*description*"] [**-a** *data_file*] *job_ID* | **"***job_ID***[***index***]"** | **-J** *job_name*

bpost [**-h** | **-V**]

Description

Provides external status information or sends data to a job in the system. Done or exited jobs cannot accept messages.

By default, operates on the message index 0. By default, posts the message `"no description"`.

If a you specify a job ID:

- You can only send messages and data to your own jobs.
- You cannot send messages and data to jobs submitted by other users.
- Only root and LSF administrators can send messages to jobs submitted by other users.
- Root and LSF administrators cannot attach data files to jobs submitted by other users.

Job names are not unique; if you specify -J *job_name*:

- You can only send messages and data to your own jobs.
- You cannot send messages and data to jobs submitted by other users.
- Root and the LSF administrators can only send messages and data to their own jobs.

A job can accept messages until it is cleaned from the system. If your application requires transfer of data from one job to another, use the -a option of `bpost`(1) to attach a data file to the job, then use the `bread`(1) command to copy the attachment to another file.

You can associate several messages and attached data files with the same job. As the job is processed, use `bread`(1) or `bstatus`(1) to retrieve the messages posted to the job. Use `bread`(1) to copy message attachments to external files.

For example, your application may require additional job status descriptions besides the ones that LSF provides internally (PEND, RUN, SUSP, etc.) Use the -d option to place your own status or job description text as a message to the job.

You can also use `bstatus -d` to update the external job status. The command:

```
bstatus -d "description" myjob
```

is equivalent to:

```
bpost -i 0 -d "description" myjob
```

With MultiCluster, both clusters must run LSF Version 7 or later. You cannot post a message to a MultiCluster job if the clusters are disconnected. You cannot attach files to MultiCluster jobs.

Options

-a *data_file*

Attaches the specified data file to the job external storage. This option is ignored for MultiCluster jobs; you can only attach a file if the job executes in the local cluster.

Use the JOB_ATTA_DIR parameter in `lsb.params`(5) to specify the directory where attachment data files are saved. The directory must have at least 1 MB of free space.

mbatchd checks for available space in the job attachment directory before transferring the file.

Use the MAX_JOB_ATTA_SIZE parameter in `lsb.params` to set a maximum size for job message attachments.

-d "*description*"

Places your own status text as a message to the job. The message description has a maximum length of 512 characters.

For example, your application may require additional job status descriptions besides the ones that LSF provides internally (PEND, RUN, SUSP, etc.)

Default: `"no description"`

-i *message_index*

Operates on the specified message index.

Default: 0

Use the MAX_JOB_MSG_NUM parameter in `lsb.params` to set a maximum number of messages for a job. With MultiCluster, to avoid conflicts, MAX_JOB_MSG_NUM should be the same in all clusters.

job_ID* | "*job_ID[index]*" | -J *job_name

Required. Operates on the specified job. With MultiCluster job forwarding model, you must always use the local job ID.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Example

```
bpost -i 1 -d "step 1" -a step1.out 2500
```

Puts the message text `step 1` into message index 1, and attaches the file `step1.out` to job 2500.

See also

bread(1), bstatus(1), MAX_JOB_ATTA_SIZE, MAX_JOB_MSG_NUM

bqueues

displays information about queues

Synopsis

bqueues [**-w** | **-l** | **-r**] [**-m** *host_name* | **-m** host_group | **-m** *cluster_name* | **-m all**] [**-u** *user_name* | **-u** *user_group* | **-u all**] [*queue_name* ...]

bqueues [**-h** | **-V**]

Description

Displays information about queues.

By default, returns the following information about all queues: queue name, queue priority, queue status, job slot statistics, and job state statistics.

When a resizable job has a resize allocation request, `bqueues` displays pending requests. When LSF adds more resources to a running resizable job, `bqueues` decreases job PEND counts and displays the added resources. When LSF removes resources from a running resizable job, `bqueues` displays the updated resources.

In MultiCluster, returns the information about all queues in the local cluster.

Batch queue names and characteristics are set up by the LSF administrator in `lsb.queues`.

CPU time is normalized.

CPU time output is not consistent with bacct

`bacct` displays the sum of CPU time consumed by all past jobs in event files, regardless of the execution host type and run time (unless you indicate a begin and end time.) For a specified job, `bacct` and `bhist` have the same result.

Because the value of CPU time for `bqueues` is used by `mbatchd` to calculate fairshare priority, it does not display the actual CPU time for the queue, but a CPU time normalized by CPU factor. This results in a different CPU time output in `bacct` and `bqueues`.

Options

-l

Displays queue information in a long multiline format. The -l option displays the following additional information: queue description, queue characteristics and statistics, scheduling parameters, resource usage limits, scheduling policies, users, hosts, associated commands, dispatch and run windows, and job controls.

Also displays user shares.

If you specified an administrator comment with the -C option of the queue control commands `qclose`, `qopen`, `qact`, and `qinact`, `qhist` displays the comment text.

Displays absolute priority scheduling (APS) information for queues configured with APS_PRIORITY.

-r

Displays the same information as the -l option. In addition, if fairshare is defined for the queue, displays recursively the share account tree of the fairshare queue.

-w

Displays queue information in a wide format. Fields are displayed without truncation.

-m *host_name* | -m *host_group* | -m *cluster_name* | -m all

Displays the queues that can run jobs on the specified host. If the keyword all is specified, displays the queues that can run jobs on all hosts.

If a host group is specified, displays the queues that include that group in their configuration. For a list of host groups see `bmgroup(1)`.

In MultiCluster, if the all keyword is specified, displays the queues that can run jobs on all hosts in the local cluster. If a cluster name is specified, displays all queues in the specified cluster.

-u *user_name* | -u *user_group* | -u all

Displays the queues that can accept jobs from the specified user. If the keyword all is specified, displays the queues that can accept jobs from all users.

If a user group is specified, displays the queues that include that group in their configuration. For a list of user groups see `bugroup(1)`).

***queue_name* ...**

Displays information about the specified queues.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Default Output

Displays the following fields:

QUEUE_NAME

The name of the queue. Queues are named to correspond to the type of jobs usually submitted to them, or to the type of services they provide.

lost_and_found

If the LSF administrator removes queues from the system, LSF creates a queue called `lost_and_found` and places the jobs from the removed queues into the `lost_and_found` queue. Jobs in the `lost_and_found` queue are not started unless they are switched to other queues (see `bswitch`).

PRIO

The priority of the queue. The larger the value, the higher the priority. If job priority is not configured, determines the queue search order at job dispatch, suspension and resumption time. Jobs from higher priority queues are dispatched first (this is contrary to UNIX process priority ordering), and jobs from lower priority queues are suspended first when hosts are overloaded.

STATUS

The current status of the queue. The possible values are:

Open

The queue is able to accept jobs.

Closed

The queue is not able to accept jobs.

Active

Jobs in the queue may be started.

Inactive

Jobs in the queue cannot be started for the time being.

At any moment, each queue is either `Open` or `Closed`, and is either `Active` or `Inactive`. The queue can be opened, closed, inactivated and re-activated by the LSF administrator using `badmin` (see `badmin(8)`).

Jobs submitted to a queue that is later closed are still dispatched as long as the queue is active. The queue can also become inactive when either its dispatch window is closed or its run window is closed (see DISPATCH_WINDOWS in the "Output for the -l Option" section). In this case, the queue cannot be activated using `badmin`. The queue is re-activated by LSF when one of its dispatch windows and one of its run windows are open again. The initial state of a queue at LSF boot time is set to open, and either active or inactive depending on its windows.

MAX

The maximum number of job slots that can be used by the jobs from the queue. These job slots are used by dispatched jobs that have not yet finished, and by pending jobs that have slots reserved for them.

A sequential job uses one job slot when it is dispatched to a host, while a parallel job uses as many job slots as is required by `bsub -n` when it is dispatched. See `bsub(1)` for details. If '–' is displayed, there is no limit.

JL/U

The maximum number of job slots each user can use for jobs in the queue. These job slots are used by your dispatched jobs that have not yet finished, and by pending jobs that have slots reserved for them. If '–' is displayed, there is no limit.

JL/P

The maximum number of job slots a processor can process from the queue. This includes job slots of dispatched jobs that have not yet finished, and job slots reserved for some pending jobs. The job slot limit per processor (JL/P) controls the number of jobs sent to each host. This limit is configured per processor so that multiprocessor hosts are automatically allowed to run more jobs. If '–' is displayed, there is no limit.

JL/H

The maximum number of job slots a host can allocate from this queue. This includes the job slots of dispatched jobs that have not yet finished, and those reserved for some pending jobs. The job slot limit per host (JL/H) controls the number of jobs sent to each host, regardless of whether a host is a uniprocessor host or a multiprocessor host. If '–' is displayed, there is no limit.

NJOBS

The total number of job slots held currently by jobs in the queue. This includes pending, running, suspended and reserved job slots. A parallel job that is running on n processors is counted as n job slots, since it takes n job slots in the queue. See `bjobs (1)` for an explanation of batch job states.

PEND

The number of job slots used by pending jobs in the queue.

RUN

The number of job slots used by running jobs in the queue.

SUSP

The number of job slots used by suspended jobs in the queue.

Long Output (-l)

In addition to the above fields, the -l option displays the following:

Description

A description of the typical use of the queue.

Default queue indication

Indicates that this is the default queue.

PARAMETERS/ STATISTICS

NICE

The nice value at which jobs in the queue are run. This is the UNIX nice value for reducing the process priority (see `nice(1)`).

STATUS

Inactive

The long format for the -l option gives the possible reasons for a queue to be inactive:

Inact_Win

The queue is out of its dispatch window or its run window.

Inact_Adm

The queue has been inactivated by the LSF administrator.

SSUSP

The number of job slots in the queue allocated to jobs that are suspended by LSF because of load levels or run windows.

USUSP

The number of job slots in the queue allocated to jobs that are suspended by the job submitter or by the LSF administrator.

RSV

The number of job slots in the queue that are reserved by LSF for pending jobs.

Migration threshold

The length of time in seconds that a job dispatched from the queue remains suspended by the system before LSF attempts to migrate the job to another host. See the MIG parameter in `lsb.queues` and `lsb.hosts`.

Schedule delay for a new job

The delay time in seconds for scheduling after a new job is submitted. If the schedule delay time is zero, a new scheduling session is started as soon as the job is submitted to the queue. See the NEW_JOB_SCHED_DELAY parameter in `lsb.queues`.

Interval for a host to accept two jobs

The length of time in seconds to wait after dispatching a job to a host before dispatching a second job to the same host. If the job accept interval is zero, a host may accept more than one job in each dispatching interval. See the JOB_ACCEPT_INTERVAL parameter in `lsb.queues` and `lsb.params`.

RESOURCE LIMITS

The hard resource usage limits that are imposed on the jobs in the queue (see `getrlimit`(2) and `lsb.queues`(5)). These limits are imposed on a per-job and a per-process basis.

The possible per-job limits are:

CPULIMIT

The maximum CPU time a job can use, in minutes, relative to the CPU factor of the named host. CPULIMIT is scaled by the CPU factor of the execution host so that jobs are allowed more time on slower hosts.

When the job-level CPULIMIT is reached, a SIGXCPU signal is sent to all processes belonging to the job. If the job has no signal handler for SIGXCPU, the job is killed immediately. If the SIGXCPU signal is handled, blocked, or ignored by the application, then after the grace period expires, LSF sends SIGINT, SIGTERM, and SIGKILL to the job to kill it.

PROCLIMIT

The maximum number of processors allocated to a job. Jobs that request fewer slots than the minimum PROCLIMIT or more slots than the maximum PROCLIMIT are rejected. If the job requests minimum and maximum job slots, the maximum slots requested cannot be less than the minimum PROCLIMIT, and the minimum slots requested cannot be more than the maximum PROCLIMIT.

MEMLIMIT

The maximum running set size (RSS) of a process. If a process uses more memory than the limit allows, its priority is reduced so that other processes are more likely to be paged in to available memory. This limit is enforced by the `setrlimit` system call if it supports the RLIMIT_RSS option.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

SWAPLIMIT

The swap space limit that a job may use. If SWAPLIMIT is reached, the system sends the following signals in sequence to all processes in the job: `SIGINT`, `SIGTERM`, and `SIGKILL`.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

PROCESSLIMIT

The maximum number of concurrent processes allocated to a job. If PROCESSLIMIT is reached, the system sends the following signals in sequence to all processes belonging to the job: `SIGINT`, `SIGTERM`, and `SIGKILL`.

THREADLIMIT

The maximum number of concurrent threads allocated to a job. If THREADLIMIT is reached, the system sends the following signals in sequence to all processes belonging to the job: `SIGINT`, `SIGTERM`, and `SIGKILL`.

The possible UNIX per-process resource limits are:

RUNLIMIT

The maximum wall clock time a process can use, in minutes. RUNLIMIT is scaled by the CPU factor of the execution host. When a job has been in the RUN state for a total of RUNLIMIT minutes, LSF sends a `SIGUSR2` signal to the job. If the job does not exit within 5 minutes, LSF sends a `SIGKILL` signal to kill the job.

FILELIMIT

The maximum file size a process can create, in KB. This limit is enforced by the UNIX `setrlimit` system call if it supports the RLIMIT_FSIZE option, or the `ulimit` system call if it supports the UL_SETFSIZE option.

DATALIMIT

The maximum size of the data segment of a process, in KB. This restricts the amount of memory a process can allocate. DATALIMIT is enforced by the `setrlimit` system call if it supports the RLIMIT_DATA option, and unsupported otherwise.

STACKLIMIT

The maximum size of the stack segment of a process. This limit restricts the amount of memory a process can use for local variables or recursive function calls. STACKLIMIT is enforced by the `setrlimit` system call if it supports the RLIMIT_STACK option.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

CORELIMIT

The maximum size of a core file. This limit is enforced by the `setrlimit` system call if it supports the RLIMIT_CORE option.

If a job submitted to the queue has any of these limits specified (see `bsub(1)`), then the lower of the corresponding job limits and queue limits are used for the job.

If no resource limit is specified, the resource is assumed to be unlimited.

By default, the limit is shown in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

SCHEDULING PARAMETERS

The scheduling and suspending thresholds for the queue.

The scheduling threshold `loadSched` and the suspending threshold `loadStop` are used to control batch job dispatch, suspension, and resumption. The queue thresholds are used in combination with the thresholds defined for hosts (see `bhosts(1)` and `lsb.hosts`(5)). If both queue level and host level thresholds are configured, the most restrictive thresholds are applied.

The `loadSched` and `loadStop` thresholds have the following fields:

r15s

The 15-second exponentially averaged effective CPU run queue length.

r1m

The 1-minute exponentially averaged effective CPU run queue length.

r15m

The 15-minute exponentially averaged effective CPU run queue length.

ut

The CPU utilization exponentially averaged over the last minute, expressed as a percentage between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

io

The disk I/O rate exponentially averaged over the last minute, in KB per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in /tmp, in MB.

swp

The amount of currently available swap space. By default, swap space is shown in MB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

mem

The amount of currently available memory. By default, memory is shown in MB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for display (MB, GB, TB, PB, or EB).

cpuspeed

The speed of each individual cpu, in megahertz (MHz).

bandwidth

The maximum bandwidth requirement, in megabits per second (Mbps).

In addition to these internal indices, external indices are also displayed if they are defined in `lsb.queues` (see `lsb.queues`(5)).

The `loadSched` threshold values specify the job dispatching thresholds for the corresponding load indices. If '–' is displayed as the value, it means the threshold is not applicable. Jobs in the queue may be dispatched to a host if the values of all the load indices of the host are within (below or above, depending on the meaning of the load index) the corresponding thresholds of the queue and the host. The same conditions are used to resume jobs dispatched from the queue that have been suspended on this host.

Similarly, the `loadStop` threshold values specify the thresholds for job suspension. If any of the load index values on a host go beyond the corresponding threshold of the queue, jobs in the queue are suspended.

JOB EXCEPTION PARAMETERS

Configured job exception thresholds and number of jobs in each exception state for the queue.

`Threshold` and `NumOfJobs` have the following fields:

overrun

Configured threshold in minutes for overrun jobs, and the number of jobs in the queue that have triggered an overrun job exception by running longer than the overrun threshold

underrun

Configured threshold in minutes for underrun jobs, and the number of jobs in the queue that have triggered an underrun job exception by finishing sooner than the underrun threshold

idle

Configured threshold (CPU time/runtime) for idle jobs, and the number of jobs in the queue that have triggered an overrun job exception by having a job idle factor less than the threshold

SCHEDULING POLICIES

Scheduling policies of the queue. Optionally, one or more of the following policies may be configured:

APS_PRIORITY

Absolute Priority Scheduling is enabled. Pending jobs in the queue are ordered according to the calculated APS value.

FAIRSHARE

Queue-level fairshare scheduling is enabled. Jobs in this queue are scheduled based on a fairshare policy instead of the first-come, first-served (FCFS) policy.

BACKFILL

A job in a backfill queue can use the slots reserved by other jobs if the job can run to completion before the slot-reserving jobs start.

Backfilling does not occur on queue limits and user limit but only on host based limits. That is, backfilling is only supported when MXJ, JL/U, JL/P, PJOB_LIMIT, and HJOB_LIMIT are reached. Backfilling is not supported when MAX_JOBS, QJOB_LIMIT, and UJOB_LIMIT are reached.

IGNORE_DEADLINE

If IGNORE_DEADLINE is set to Y, starts all jobs regardless of the run limit.

EXCLUSIVE

Jobs dispatched from an exclusive queue can run exclusively on a host if the user so specifies at job submission time (see `bsub(1)`). Exclusive execution means that the job is sent to a host with no other batch job running there, and no further job, batch or interactive, is dispatched to that host while the job is running. The default is not to allow exclusive jobs.

NO_INTERACTIVE

This queue does not accept batch interactive jobs. (see the -I, -Is, and -Ip options of `bsub(1)`). The default is to accept both interactive and non-interactive jobs.

ONLY_INTERACTIVE

This queue only accepts batch interactive jobs. Jobs must be submitted using the -I, -Is, and -Ip options of `bsub(1)`. The default is to accept both interactive and non-interactive jobs.

FAIRSHARE_QUEUES

Lists queues participating in cross-queue fairshare. The first queue listed is the master queue—the queue in which fairshare is configured; all other queues listed inherit the fairshare policy from the master queue. Fairshare information applies to all the jobs running in all the queues in the master-slave set.

QUEUE_GROUP

Lists queues participating in an absolute priority scheduling (APS) queue group.

If both FAIRSHARE and APS_PRIORITY are enabled in the same queue, the FAIRSHARE_QUEUES are not displayed. These queues are instead displayed as QUEUE_GROUP.

DISPATCH_ORDER

DISPATCH_ORDER=QUEUE is set in the master queue. Jobs from this queue are dispatched according to the order of queue priorities first, then user fairshare priority. Within the queue, dispatch order is based on user share quota. This avoids having users with higher fairshare priority getting jobs dispatched from low-priority queues.

USER_SHARES

A list of [*user_name*, *share*] pairs. *user_name* is either a user name or a user group name. *share* is the number of shares of resources assigned to the user or user group. A party receives a portion of the resources proportional to that party's share divided by the sum of the shares of all parties specified in this queue.

DEFAULT HOST SPECIFICATION

The default host or host model that is used to normalize the CPU time limit of all jobs.

If you want to view a list of the CPU factors defined for the hosts in your cluster, see `lsinfo(1)`. The CPU factors are configured in `lsf.shared(5)`.

The appropriate CPU scaling factor of the host or host model is used to adjust the actual CPU time limit at the execution host (see CPULIMIT in `lsb.queues(5)`). The DEFAULT_HOST_SPEC parameter in `lsb.queues` overrides the system DEFAULT_HOST_SPEC parameter in `lsb.params` (see `lsb.params(5)`). If a user explicitly gives a host specification when submitting a job using `bsub -c` *cpu_limit* [/*host_name* | /*host_model*], the user specification overrides the values defined in both `lsb.params` and `lsb.queues`.

RUN_WINDOWS

The time windows in a week during which jobs in the queue may run.

When a queue is out of its window or windows, no job in this queue is dispatched. In addition, when the end of a run window is reached, any running jobs from this queue are suspended until the beginning of the next run window, when they are resumed. The default is no restriction, or always open.

DISPATCH_WINDOWS

Dispatch windows are the time windows in a week during which jobs in the queue may be dispatched.

When a queue is out of its dispatch window or windows, no job in this queue is dispatched. Jobs already dispatched are not affected by the dispatch windows. The default is no restriction, or always open (that is, twenty-four hours a day, seven days a week). Note that such windows are only applicable to batch jobs. Interactive jobs scheduled by LIM are controlled by another set of dispatch windows (see `lshosts (1)`). Similar dispatch windows may be configured for individual hosts (see `bhosts (1)`).

A window is displayed in the format *begin_time–end_time*. Time is specified in the format [*day*:]*hour*[:*minute*], where all fields are numbers in their respective legal ranges: 0(Sunday)-6 for *day*, 0-23 for *hour*, and 0-59 for *minute*. The default value for *minute* is 0 (on the hour). The default value for *day* is every day of the week. The *begin_time* and *end_time* of a window are separated by '–', with no blank characters (SPACE and TAB) in between. Both *begin_time* and *end_time* must be present for a window. Windows are separated by blank characters.

USERS

A list of users allowed to submit jobs to this queue. LSF administrators can submit jobs to the queue even if they are not listed here.

User group names have a slash (/) added at the end of the group name. See `bugroup (1)`.

If the fairshare scheduling policy is enabled, users cannot submit jobs to the queue unless they also have a share assignment. This also applies to LSF administrators.

HOSTS

A list of hosts where jobs in the queue can be dispatched.

Host group names have a slash (/) added at the end of the group name. See `bmgroup (1)`.

NQS DESTINATION QUEUES

A list of NQS destination queues to which this queue can dispatch jobs.

When you submit a job using `bsub -q` *queue_name*, and the specified queue is configured to forward jobs to the NQS system, LSF routes your job to one of the NQS destination queues. The job runs on an NQS batch server host, which is not a member of the LSF cluster. Although running on an NQS system outside the LSF cluster, the job is still managed by LSF in almost the same way as jobs running inside the LSF cluster. Thus, you may have your batch jobs transparently sent to an NQS system to run and then get the results of your jobs back. You may use any supported user interface, including LSF commands and NQS commands (see `lsnqs(1)`) to submit, monitor, signal and delete your batch jobs that are running in an NQS system. See `lsb.queues(5)` and `bsub(1)` for more information.

ADMINISTRATORS

A list of queue administrators. The users whose names are specified here are allowed to operate on the jobs in the queue and on the queue itself. See `lsb.queues(5)` for more information.

PRE_EXEC

The PRE_EXEC command runs on the execution host before the job associated with the queue is dispatched to the execution host (or to the first host selected for a parallel batch job).

POST_EXEC

The post-execution command for the queue. The POST_EXEC command runs on the execution host after the job finishes.

REQUEUE_EXIT_VALUES

Jobs that exit with these values are automatically requeued. See `lsb.queues(5)` for more information.

RES_REQ

Resource requirements of the queue. Only the hosts that satisfy these resource requirements can be used by the queue.

Maximum slot reservation time

The maximum time in seconds a slot is reserved for a pending job in the queue. See the SLOT_RESERVE=MAX_RESERVE_TIME[n] parameter in `lsb.queues`.

RESUME_COND

The conditions that must be satisfied to resume a suspended job on a host. See `lsb.queues(5)` for more information.

STOP_COND

The conditions that determine whether a job running on a host should be suspended. See `lsb.queues(5)` for more information.

JOB_STARTER

An executable file that runs immediately prior to the batch job, taking the batch job file as an input argument. All jobs submitted to the queue are run via the job starter, which is generally used to create a specific execution environment before processing the jobs themselves. See `lsb.queues(5)` for more information.

CHUNK_JOB_SIZE

Chunk jobs only. Specifies the maximum number of jobs allowed to be dispatched together in a chunk job. All of the jobs in the chunk are scheduled and dispatched as a unit rather than individually. The ideal candidates for job chunking are jobs that typically takes 1 to 2 minutes to run.

SEND_JOBS_TO

MultiCluster. List of remote queue names to which the queue forwards jobs.

RECEIVE_JOBS_FROM

MultiCluster. List of remote cluster names from which the queue receives jobs.

PREEMPTION

PREEMPTIVE

The queue is preemptive. Jobs in this queue may preempt running jobs from lower-priority queues, even if the lower-priority queues are not specified as preemptive.

PREEMPTABLE

The queue is preemptable. Running jobs in this queue may be preempted by jobs in higher-priority queues, even if the higher-priority queues are not specified as preemptive.

RERUNNABLE

If the RERUNNABLE field displays `yes`, jobs in the queue are rerunnable. That is, jobs in the queue are automatically restarted or rerun if the execution host becomes unavailable. However, a job in the queue is not restarted if you remove the rerunnable option from the job.

CHECKPOINT

If the CHKPNTDIR field is displayed, jobs in the queue are checkpointable. Jobs use the default checkpoint directory and period unless you specify other values. Note that a job in the queue is not checkpointed if you remove the checkpoint option from the job.

CHKPNTDIR

Specifies the checkpoint directory using an absolute or relative path name.

CHKPNTPERIOD

Specifies the checkpoint period in seconds.

Although the output of `bqueues` reports the checkpoint period in seconds, the checkpoint period is defined in minutes (the checkpoint period is defined through the `bsub -k` "*checkpoint_dir* []" option, or in `lsb.queues`).

JOB CONTROLS

The configured actions for job control. See JOB_CONTROLS parameter in `lsb.queues`.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is either SUSPEND, RESUME, or TERMINATE.

ADMIN ACTION COMMENT

If the LSF administrator specified an administrator comment with the -C option of the queue control commands `qclose`, `qopen`, `qact`, and `qinact`, `qhist` the comment text is displayed.

SLOT_SHARE

Share of job slots for queue-based fairshare. Represents the percentage of running jobs (job slots) in use from the queue. SLOT_SHARE must be greater than zero.

The sum of SLOT_SHARE for all queues in the pool does not need to be 100%. It can be more or less, depending on your needs.

SLOT_POOL

Name of the pool of job slots the queue belongs to for queue-based fairshare. A queue can only belong to one pool. All queues in the pool must share the same set of hosts.

SHARE_INFO_FOR

User shares and dynamic priority information based on the scheduling policy in place for the queue.

USER/GROUP

Name of users or user groups who have access to the queue.

SHARES

Number of shares of resources assigned to each user or user group in this queue, as configured in the file `lsb.queues`. The shares affect dynamic user priority for when fairshare scheduling is configured at the queue level.

PRIORITY

Dynamic user priority for the user or user group. Larger values represent higher priorities. Jobs belonging to the user or user group with the highest priority are considered first for dispatch.

In general, users or user groups with larger SHARES, fewer STARTED and RESERVED, and a lower CPU_TIME and RUN_TIME have higher PRIORITY.

STARTED

Number of job slots used by running or suspended jobs owned by users or user groups in the queue.

RESERVED

Number of job slots reserved by the jobs owned by users or user groups in the queue.

CPU_TIME

Cumulative CPU time used by jobs of users or user groups executed in the queue. Measured in seconds, to one decimal place.

LSF calculates the cumulative CPU time using the actual (not normalized) CPU time and a decay factor such that 1 hour of recently-used CPU time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default).

RUN_TIME

Wall-clock run time plus historical run time of jobs of users or user groups that are executed in the queue. Measured in seconds.

LSF calculates the historical run time using the actual run time of finished jobs and a decay factor such that 1 hour of recently-used run time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default). Wall-clock run time is the run time of running jobs.

ADJUST

Dynamic priority calculation adjustment made by the user-defined fairshare plugin (`libfairshareadjust.*`).

The fairshare adjustment is enabled and weighted by the parameter FAIRSHARE_ADJUSTMENT_FACTOR in `lsb.params`.

Recursive Share Tree Output (-r)

In addition to the fields displayed for the -l option, the -r option displays the following:

SCHEDULING POLICIES

FAIRSHARE

The -r option causes `bqueues` to recursively display the entire share information tree associated with the queue.

See also

bugroup(1), `nice`(1), `getrlimit`(2), lsb.queues(5), bsub(1), bjobs(1), bhosts(1), badmin(8), `mbatchd`(8)

bread

reads messages and attached data files from a job

Synopsis

bread [**-i** *message_index*] [**-a** *file_name*] *job_ID* | **"***job_ID***[***index***]"** | **-J** *job_name*

bread [**-h** | **-V**]

Description

Reads messages and data posted to an unfinished job with `bpost`.

By default, displays the message description text of the job. By default, operates on the message with index 0.

You can read messages and data from a job until it is cleaned from the system. You cannot read messages and data from done or exited jobs.

If a you specify a job ID:

- You can get read messages of jobs submitted by other users, but you cannot read data files attached to jobs submitted by other users.
- You can only read data files attached to your own jobs.
- Root and LSF administrators can read messages of jobs submitted by other users.
- Root and LSF administrators cannot read data files attached to jobs submitted by other users.

Job names are not unique; if you specify -J *job_name*:

- You only can read messages and data from your own jobs.
- You cannot read messages and data from jobs submitted by other users.
- Root and the LSF administrators can only read messages and data from their own jobs.

The command:

```
bstatus
```

is equivalent to:

```
bread -i 0
```

Options

-a *file_name*

Gets the text message and copies the data file attached to the specified message index of the job to the file specified by *file_name*. Data files cannot be attached to MultiCluster jobs.

If you do not specify a message index, copies the attachment of message index 0 to the file. The job must have an attachment, and you must specify a name for the file you are copying the attachment to. If the file already exists, `-a` overwrites it with the new file.

By default, -a gets the attachment file from the directory specified by the JOB_ATTA_DIR parameter. If JOB_ATTA_DIR is not specified, job message attachments are saved in `LSB_SHAREDIR/info/`.

-i *message_index*

Specifies the message index to be retrieved.

Default: 0

job_ID* | "*job_ID[index]*" | -J *job_name

Required. Specify the job to operate on.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Example

```
bpost -i 1 -d "step 1" -a step1.out 2500
bread -i 1 -a step2.in 2500
JOBID       MSG_ID FROM       POST_TIME      DESCRIPTION
2500        1      user1      May 19 13:59   step 1
```

Displays the message description text `step 1` for message index 1 of job 2500 and copies the data in the file `step1.out` attached to message 1 to the file `step2.in`.

See also

bpost(1), bstatus(1), bsub(1), JOB_ATTA_DIR

brequeue

kills and requeues a job

Synopsis

brequeue [**-J** *job_name* | **-J "***job_name***[***index_list***]"**] [**-u** *user_name* | **-u all**] [*job_ID* | **"***job_ID***[***index_list***]"**] [**-d**] [**-e**] [**-r**] [**-a**] [**-H**]

brequeue [**-h** | **-V**]

Description

You can only use `brequeue` on a job you own, unless you are root or the LSF administrator.

Kills a running (RUN), user-suspended (USUSP), or system-suspended (SSUSP) job and returns it to the queue. A job that is killed and requeued retains its submit time but is dispatched according to its requeue time. When the job is requeued, it is assigned the PEND status or PSUSP if the -H option is used. Once dispatched, the job starts over from the beginning. The requeued job keeps the same job ID.

When JOB_INCLUDE_POSTPROC=Y is set in `lsb.params` or in an application profile in `lsb.applications`, job requeue occurs only after post-execution processing, not when the job finishes.

Use `brequeue` to requeue job arrays or job array elements.

By default, kills and requeues your most recently submitted job when no job ID is specified.

With MultiCluster, you can only use `brequeue` on jobs in local queues. A job that is killed and requeued is assigned a new job ID on the cluster in which it is executed, but it retains the same job ID on the cluster from which it was submitted. For example, a job from cluster A that is killed and requeued and then run on cluster B is assigned a new job ID on cluster B. However, when the `bjobs` command is used from cluster A, the submitting cluster, the job is displayed with the original job ID. When the `bjobs` command is used from cluster B, the execution cluster, the job is displayed with the new job ID.

When absolute job priority scheduling (APS) is configured in the queue, all requeued jobs are treated as newly submitted jobs for APS calculation. The job priority, system, and ADMIN APS factors are reset on requeue.

Options

-a

Requeues all jobs including running jobs, suspending jobs, and jobs with EXIT or DONE status.

-d

Requeues jobs that have finished running with DONE job status.

-e

Requeues jobs that have terminated abnormally with EXIT job status.

-H

Requeues jobs to PSUSP job status.

-r

Requeues jobs that are running.

-J *job_name* | -J "*job_name[index_list]*"

Operates on the specified job.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-u *user_name* | -u all

Operates on the specified user's jobs or all jobs. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

Only `root` and LSF administrators can requeue jobs submitted by other users.

***job_ID* |"*job_ID[index_list]*"**

Operates on the specified job or job array elements.

The value of 0 for *job_ID* is ignored.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Limitations

`brequeue` cannot be used on interactive batch jobs; `brequeue` only kills interactive batch jobs, it does not restart them.

bresize

release slots from a running resizable job, and cancel pending job resize allocation requests

Synopsis

bresize *subcommand*

bresize [**-h** | **-V**]

Description

Use `bresize release` to explicitly release slots from a running job. When releasing slots from an allocation, a minimum of 1 slot on the first execution host must be retained.

Use `bresize cancel` to cancel a pending allocation request for the specified job ID. The active pending allocation request is generated by LSF automatically for autoresizable jobs. If job does not have active pending request, the command fails with an error message.

By default, only cluster administrators, queue administrators, root and the job owner are allowed to run `bresize` to change job allocations.

User group administrators are allowed to run `bresize` to change the allocation of jobs within their user groups.

Subcommand List

release [**-c**] [**-rnc** *resize_notification_cmd* | **-rncn**] *released_host_specification job_ID***cancel** *job_ID***-h-V**

Options

-c

Optional. Cancel the active pending resource request when releasing slots from existing allocation. By default, the command only releases slots for jobs with pending requests.

-rnc *resize_notification_cmd*

Optional. Specify the name of an executable to be invoked on the first execution host when the job allocation has been modified. This setting only applies to this release request, which overrides any notification command specified in `bsub` or an application profile. The resize notification command runs under the user account of job.

-rncn

Cancels the resize notification command at both job-level and application-level. This setting only applies to this request.

released_host_specification

Required. Defines the list of hosts to be released. The following is the EBNF definition of the released host specification:

```
<released_host_spec> ::= all | all <exclude_host_list_spec>|<host_list_spec>

<host_list_spec> ::= <host_spec>|<host_list_spec><host_spec>

<exclude_host_list_spec> ::= <exclude_host_spec> | <exclude_host_list_spec>
<exclude_host_spec>

<exclude_host_spec> ::= ~<host_spec>

<host_spec> ::= [<positive_integer>*]<host_name>
```

all

Specifies all the slots currently being used by the job. If all is used alone, it means release every slot except one slot from the first execution node. all can also be used with a list of hosts to exclude with the tilde (not) operator (~).

host_spec

Release the number of slots specified by *positive_integer* on the host specified by *host_name*. If the number of slots is not specified, all slots on specified host are released.

~

Specifies hosts to exclude when releasing slots. Slots on the specified hosts are not released. The tilde (not) operator (~) must be used together with all keyword.

job_ID

Required. The job ID of the job to be resized.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

Examples

For a job that uses 8 slots across 4 nodes: 2 on hostA 2 on hostB, 2 on hostC, and 2 on hostD, the following command releases all slots except hostA. After releasing, the job allocation becomes 2 on hostA:

```
bresize release "all ~hostA" 100
```

The following command releases all slots except 1 slot from hostA. After releasing, the job allocation becomes 1 on hostA:

```
bresize release all 100 or bresize release "all ~1*hostA" 100
```

The following command releases one slot from each of four hosts. After releasing, the job allocation becomes 1 on hostA, 1 on hostB, 1 on hostC, and 1 on hostD:

```
bresize release "1*hostA 1*hostB 1*hostC 1*hostD"  100
```

See also

bsub, `lsb.applications`

bresources

displays information about resource reservation and resource limits configuration

Synopsis

bresources [**-s**] [*resource_name* ...]

bresources [**-h** | **-V**]

Description

By default, `bresources` displays all resource configurations in `lsb.resources`. This is the same as `blimits -c`.

Options

-s

Displays per-resource reservation configurations from the ReservationUsage section of `lsb.resources`. For example:

```
bresources -s
Begin ReservationUsage
RESOURCE            METHOD    RESERVElicense1            PER_HOST     Y
license2            PER_SLOT     Y
license3            PER_JOB      N
license5            PER_HOST     Y
license6            PER_JOB      Y
End ReservationUsage
```

***resource_name* ...**

Only displays information about the specified resource. For example:

```
bresources -s  license1
RESOURCE            METHOD    RESERVElicense1            PER_HOST     Y
```

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

brestart

restarts checkpointed jobs

Synopsis

brestart [*bsub_options*] [**-f**] *checkpoint_dir* [*job_ID* | "*job_ID***[***index***]"**]

brestart [**-h** | **-V**]

Option List

-B

-f

-N

-x

-b *begin_time*

-C *core_limit*

-c [*hour***:]***minute*[**/***host_name* | **/***host_model*]

-D *data_limit*

-E "*pre_exec_command* [*argument* ...]"

-F *file_limit*

-m "*host_name*[**+**[*pref_level*]] | *host_group*[**+**[*pref_level*]] ..."

-G *user_group*

-M *mem_limit*

-q "*queue_name* ..."

-S *stack_limit*

-t *term_time*

-w '*dependency_expression***'**

-W *run_limit*[**/***host_name*| **/***host_model*]

checkpoint_dir [*job_ID* | "*job_ID***[***index***]"**]

Description

Restarts a checkpointed job using the checkpoint files saved in *checkpoint_dir/last_job_ID/*. Only jobs that have been successfully checkpointed can be restarted.

Jobs are re-submitted and assigned a new job ID. The checkpoint directory is renamed using the new job ID, *checkpoint_dir/new_job_ID/*.

The file path of the checkpoint directory can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

By default, jobs are restarted with the same output file and file transfer specifications, job name, window signal value, checkpoint directory and period, and rerun options as the original job.

To restart a job on another host, both hosts must be binary compatible, run the same OS version, have access to the executable, have access to all open files (LSF must locate them with an absolute path name), and have access to the checkpoint directory.

The environment variable LSB_RESTART is set to Y when a job is restarted.

LSF invokes the erestart(8) executable found in LSF_SERVERDIR to perform the restart.

Only the `bsub` options listed here can be used with `brestart`.

Like `bsub`, `brestart` calls the master `esub` (`mesub`), which invokes any mandatory `esub` executables configured by an LSF administrator, and any executable named `esub` (without *.application*) if it exists in LSF_SERVERDIR. Only `esub` executables invoked by `bsub` can change the job environment on the submission host. An `esub` invoked by `brestart` cannot change the job environment.

Options

The following option applies only to `brestart`.

-f

Forces the job to be restarted even if non-restartable conditions exist (these conditions are operating system specific).

See `bsub(1)` for a description of all other options.

Limitations

In kernel-level checkpointing, you cannot change the value of core limit, CPU limit, stack limit or memory limit with `brestart`.

See also

bsub(1), bjobs(1), bmod(1), bqueues(1), bhosts(1), bchkpnt(1), lsbqueues(5), `echkpnt`(8), `erestart`(8), `mbatchd` (8)

bresume

resumes one or more suspended jobs

Synopsis

bresume [**-app** *application_profile_name*] [**-g** *job_group_name*] [**-J** *job_name*] [**-m** *host_name*] [**-q** *queue_name*] [**-sla** *service_class_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [**0**]

bresume [*job_ID* | "*job_ID*[*index_list*]"] ...

bresume [**-h** | **-V**]

Description

Sends the SIGCONT signal to resume one or more of your suspended jobs.

Only root and LSF administrators can operate on jobs submitted by other users. You cannot resume a job that is not suspended. Using `bresume` on a job that is not in either the PSUSP or the USUSP state has no effect.

You must specify a job ID or -g, -J, -m, -u, or -q. You cannot resume a job that is not suspended. Specify 0 (zero) to resume multiple jobs.

You can also use `bkill -s CONT` to send the resume signal to a job.

If a signal request fails to reach the job execution host, LSF retries the operation later when the host becomes reachable. LSF retries the most recent signal request.

Jobs that are suspended by the administrator can only be resumed by the administrator or root; users do not have permission to resume a job suspended by another user or the administrator. Administrators or root can resume jobs suspended by users or administrators.

ENABLE_USER_RESUME parameter (lsb.params)

If ENABLE_USER_RESUME=Y in `lsb.params`, users can resume their own jobs that have been suspended by the administrator.

Options

0

Resumes all the jobs that satisfy other options (-g, -m, -q, -u, and -J).

-app *application_profile_name*

Resumes only jobs associated with the specified application profile. You must specify an existing application profile.

-g *job_group_name*

Resumes only jobs in the specified job group.

-J *job_name*

Resumes only jobs with the specified name.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-m *host_name*

Resumes only jobs dispatched to the specified host.

-q *queue_name*

Resumes only jobs in the specified queue.

-sla *service_class_name*

Resume jobs belonging to the specified service class.

Use `bsla` to display the properties of service classes configured in `LSB_CONFDIR/`*cluster_name*`/configdir/lsb.serviceclasses` (see `lsb.serviceclasses`(5)) and dynamic information about the state of each configured service class.

-u *user_name* | -u *user_group* | -u all

Resumes only jobs owned by the specified user or group, or all users if the reserved user name all is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

***job_ID* ... | "*job_ID*[*index_list*]" ...**

Resumes only the specified jobs. Jobs submitted by any user can be specified here without using the -u option.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

bresume -q night 0

Resumes all of the user's suspended jobs that are in the `night` queue. If the user is the LSF administrator, resumes all suspended jobs in the `night` queue.

bresume -g /risk_group 0

Resumes all suspended jobs in the job group `/risk_group`.

See also

bsub(1), bjobs(1), bqueues(1), bhosts(1), bstop(1), bkill(1), bgadd(1), bgdel(1), bjgroup(1), bparams(1), bapp(1), mbatchd(8), `kill`(1), `signal`(2) lsb.params(5), lsb.applications(5)

brlainfo

displays host topology information

Synopsis

brlainfo [**-l**] [*host_name* ...]

brlainfo [**-h** | **-V**]

Description

`brlainfo` contacts the Platform LSF HPC topology adapter (RLA) on the specified host and presents topology information to the user. By default, displays information about all hosts running RLA.

Options

-l

Long format. Displays additional host topology information.

***host_name* ...**

Only displays information about the specified host.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Default output

Displays the following fields:

HOSTNAME

Name of the host running RLA

CPUSET_OS

RLA host operating system

NCPUS

Total number of CPUs

FREECPUS

Number of free CPUS

NNODES

Number of nodes allocated

NCPU/NODE

Number of CPUs per node

NSTATIC_CPUSETS

Number of static cpusets allocated

Long output (-l)

The `-l` option displays a long format listing with the following additional fields:

FREE CPU LIST

List of free CPUs in the cpuset

For example:

```
0-2
```

NFREECPUS ON EACH NODE

Number of free CPUs on each node

For example:

```
2/0,1/1
```

STATIC CPUSETS

List of static cpuset names

For example:

```
NO STATIC CPUSETS
```

CPU_RADIUS

For example:

```
2,3,3,3,3,3,3,3
```

- 2 CPUs are available within radius 0
- 3 CPUs are available within radius 1, 2, 3, 4, 5, 6, and 7.

CPUs grouped within a smaller radius can be thought of as being closer together and therefore have better communications performance.

Examples

brlainfo hostA hostB hostC

```
HOSTNAME          CPUSET_OS  NCPUS  NFREECPUS NNODES  NCPU/NODE NSTATIC_CPUSETS
hostA             SGI_IRIX   2      2         1       2         0
hostB             PROPACK_4  4      4         2       2         0
hostC             PROPACK_4  4      3         2       2         0
```

brlainfo -l

```
HOST: hostC
CPUSET_OS   NCPUS  NFREECPUS NNODES  NCPU/NODE NSTATIC_CPUSETS
PROPACK_4   4      3         2       2         0
FREE CPU LIST: 0-2
NFREECPUS ON EACH NODE: 2/0,1/1
STATIC CPUSETS: NO STATIC CPUSETS
CPU_RADIUS: 2,3,3,3,3,3,3,3
```

brsvadd

adds an advance reservation

Synopsis

brsvadd [**-o**] [**-d "***description***"**] [**-N** *reservation_name*]

{**-s** | **-n** *job_slots* {**-u** *user_name* | **-g** *group_name*}}

{**-m "***host_name* | *host_group* ...**"** [**-R "***res_req***"**] |

[**-m "***host_name* | *host_group* ...**"**] **-R "***res_req***"**}

{**-b** *begin_time* **-e** *end_time* | **-t** *time_window*}

brsvadd {**-h** | **-V**}

Description

Caution:

By default, this command can only be used by LSF administrators or root.

Reserves job slots in advance for a specified period of time for a user or user group, or for system maintenance purposes. Use -b and -e for one-time reservations, and -t for recurring reservations.

To allow users to create their own advance reservations without administrator intervention, configure advance reservation policies in the ResourceReservation section of `lsb.resources`.

Only administrators, root, or the users listed in the ResourceReservation section can add reservations for themselves or any other user or user group.

Options

-o

Creates an open advance reservation. A job with an open advance reservation only has the advance reservation property during the reservation window, after which the job becomes a normal job, not subject to termination when the reservation window closes.

This prevents jobs from being killed if the reservation window is too small. Instead, the job is suspended and normal scheduling policies apply after the reservation window.

-s

Creates a reservation for system use. LSF does not dispatch jobs to the specified hosts while the reservation is active.

When specifying a system reservation with -s, you do not need to specify the number of job slots to reserve with the -n option.

-b *begin_time*

Begin time for a one-time reservation. The begin time is in the form

`[[[`*`year`*`:]`*`month`*`:]`*`day`*`:]hour:`*`minute`*

with the following ranges:

- *year*: any year after 1900 (YYYY)
- *month*: 1-12 (MM)
- *day of the month*: 1-31 (dd)
- *hour*: 0-23 (hh)
- *minute*: 0-59 (mm)

You must specify at least hour:minute. Year, month, and day are optional. Three fields are assumed to be day:hour:minute, four fields are assumed to be month:day:hour:minute, and five fields are year:month:day:hour:minute.

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The time value for -b must use the same syntax as the time value for -e. It must be earlier than the time value for -e, and cannot be earlier than the current time.

-d "*description*"

Specifies a description for the reservation to be created. The description must be provided as a double quoted text string. The maximum length is 512 characters.

-e *end_time*

End time for a one-time reservation. The end time is in the form

```
[[[year:]month:]day:]hour:minute
```

with the following ranges:

- *year*: any year after 1900 (YYYY)
- *month*: 1-12 (MM)
- *day of the month*: 1-31 (dd)
- *hour*: 0-23 (hh)
- *minute*: 0-59 (mm)

You must specify at least hour:minute. Year, month, and day are optional. Three fields are assumed to be day:hour:minute, four fields are assumed to be month:day:hour:minute, and five fields are year:month:day:hour:minute.

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The time value for -e must use the same syntax as the time value for -b. It must be later than the time value for -b.

-g *group_name*

Creates a reservation for a user group.

The -g *group_name* option does not support the @cluster notation for advance reservations on remote clusters.

-m "*host_name* | *host_group* ..."

Lists hosts for which job slots specified with -n are reserved. At job submission, LSF considers the hosts in the specified order.

If you also specify a resource requirement string with the -R option, -m is optional.

The hosts can be local to the cluster or hosts leased from remote clusters.

-N *reservation_name*

Specifies a user-defined advance reservation name unique in an LSF cluster. The name is a string of letters, numeric characters, underscores, and dashes beginning with a letter. The maximum length of the name is 40 characters.

If no user-defined advance reservation name is specified, LSF creates the reservation with a system assigned name with the form

```
user_name#sequence
```

For example:

```
brsvadd -n 3 -M "hostA hostB" -u user2 -b 16:0 -e 17:0 -d "Production AR test"
Reservation user2#0 (Production AR test) is created
brsvadd -n 2 -N Production_AR -M hostA -u user2 -b 16:0 -e 17:0 -d "Production AR test"
Reservation Production_AR (Production AR test) is created
```

If a job already exists that references a reservation with the specified name, an error message is returned: The specified reservation name is referenced by a job.

-n *job_slots*

Number of job slots to reserve. *job_slots* must be less than or equal to the actual number of job slots for the hosts selected by -m or -R for the reservation.

If you also specify the reservation for system use with the -s option, -n is optional.

-R "*res_req*"

Selects hosts for the reservation according to the specified resource requirements. Only hosts that satisfy the resource requirement expression are reserved. -R accepts any valid resource requirement string, but only the select string takes effect.

If you also specify a host list with the -m option, -R is optional.

For more information about specifying resource requirement strings, see *Administering Platform LSF*.

The size of the resource requirement string is limited to 512 bytes.

-t *time_window*

Time window for a recurring reservation.

To specify a time window, specify two time values separated by a hyphen (-), with no space in between:

```
time_window = begin_time-end_time
```

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23

- *minute*: 0-59

Specify a time window one of the following ways:

- *hour-hour*
- *hour*:*minute-hour*:*minute*
- *day*:*hour*:*minute-day*:*hour*:*minute*

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (`:00`). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

LSF administrators can prevent running jobs from being killed when the reservation expires by changing the termination time of the job using the reservation (`bmod -t`) before the reservation window closes.

When the job starts running, the run limit of the reservation is set to the minimum of the job run limit (if specified), the queue run limit (if specified), or the duration of the time window.

-u *user_name*

Creates a reservation for an individual user.

The -u *user_name* option does not support the @cluster notation for advance reservations on remote clusters.

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

Examples

The following command creates a one-time advance reservation for 1024 job slots on host `hostA` for user `user1` between 6:00 a.m. and 8:00 a.m. today:

```
brsvadd -n 1024 -m hostA -u user1 -b 6:0 -e 8:0
Reservation "user1#0" is created
```

The hosts specified by -m can be local to the cluster or hosts leased from remote clusters.

The following command creates an advance reservation for 1024 job slots on two hosts `hostA` and `hostB` for user group `groupA` every Wednesday from 12:00 midnight to 3:00 a.m.:

```
brsvadd -n 1024 -m "hostA hostB" -g groupA -t "3:0:0-3:3:0"
Reservation "groupA#0" is created
```

The following command creates an open advance reservation for 1024 job slots on host hostA for user user1 between 6:00 a.m. and 8:00 a.m. today.

```
brsvadd -o -n 1024 -m hostA -u user1 -b 6:0 -e 8:0
Reservation "user1#0" is created
```

See also

brsvdel, brsvmod, brsvs, `lsb.resources`

brsvdel

deletes an advance reservation

Synopsis

brsvdel *reservation_ID ...*

brsvdel {**-h** | **-V**}

Description

Caution:

By default, this command can only be used by LSF administrators or root.

Deletes advance reservations for the specified reservation IDs.

For example, if the following command was used to create the reservation `user1#0`,

```
brsvadd -n 1024 -m hostA -u user1 -b 13:0 -e 18:0
Reservation "user1#0" is created
```

the following command deletes the reservation:

```
brsvdel user1#0
Reservation user1#0 is being deleted
```

You can delete multiple reservations at a time.

To allow users to delete their own advance reservations without administrator intervention, configure advance reservation policies in the ResourceReservation section of `lsb.resources`.

Administrators and root can delete any reservations. Users listed in the ResourceReservation section can only delete reservations they created themselves.

Options

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

See also

brsvadd, brsvmod, brsvs, `lsb.resources`

brsvmod

modifies an advance reservation

Synopsis

brsvmod [**-o** | **-on**] [**-d "***description***"**] [**-u** *user_name* | **-g** *group_name*] [[**-b** *begin_time* | [**+**|**-**]*minutes*] [**-e** *end_time* | [**+**|**-**]*minutes*]] | [**-t** *time_window*] *reservation_ID*

brsvmod disable {**-td "***begin_date***-***end_date***"** | **-tn**} [**-f**] *reservation_ID*

brsvmod addhost {**-n** *job_slots* **-R "***res_req***"** [**-m "***host_name ...* | *host_group ...***"**]} | {[**-n** *job_slots*] **-m "***host_name ...* | *host_group ...***"**} *reservation_ID*

brsvmod rmhost {**-n** *job_slots* [**-m "***host_name ...* | *host_group ...***"**]} | {[**-n** *job_slots*] **-m "***host_name ...* | *host_group ...***"**} *reservation_ID*

brsvmod {**-h** | **-V**}

Description

Caution:

By default, this command can only be used by LSF administrators or root.

Replaces advance reservation option values previously created, extends or reduces the reservation time window, or adds or removes reserved hosts of the advance reservation specified by *reservation_ID*. For a recurring reservation, can disable specified occurrences of the reservation.

Administrators and root can modify any reservations. Users listed in the ResourceReservation section of `lsb.resources`, can only modify reservations they created themselves.

The original value for user, user group, or time window, can be overridden with a new value by specifying the option as in `brsvadd`. Change a reservation from closed (the default) to open with the -o option, or from open to closed with the -on option.

Options -n, -m, and -R must be used with the subcommands `addhost` or `rmhost`. These options allow adding or removing from the original values.

The -td and -tn options are only allowed in `disable` subcommand.

All three subcommands are mutually exclusive. The time window options -b, -e and -t are not valid in any of the subcommands.

You cannot modify the start time of an active reservation.

`brsvmod` does not support the ***reservation_ID@cluster_name*** notation for advance reservations on remote clusters, or the ***user_name@cluster_name*** notation for reservations with remote users.

The job slot requirements of the -n option must be satisfied, but -m or -R provides a candidate list for processing, which does not trigger error unless no valid hosts are the list. For instance, if you specify

```
-n 3 -m "host1 host2"
```

3 slots are required. LSF tries to find as many slots as possible from host1. If 3 slots are not available on host1, then LSF tries to find the rest from host2. Hosts with no slots available are removed from the list when the request is handled.

Subcommands

addhost [-n *job_slots* [-R "*res_req*"]] [-m "*host_name ...* | *host_group ...*"] *reservation_ID*

Adds hosts and slots on hosts into the original reservation allocation. The hosts can be local to the cluster or hosts leased from remote clusters.

Adding a host without -n reserves all available slots on the host that are not already reserved by other reservations. You can specify the number of slots to be added from the host list specified with -n, but -n cannot be used alone. -m can be used alone if no host group is specified in the list. You cannot specify -R without with -n.

The specified slot number must be less than or equal to the available number of slots for the hosts.

Only hosts can be added (-m) to a system reservation. Slots cannot be added (-n) to a system reservation.

disable -td "*begin_date-end_date*" | -tn [-f] *reservation_ID*

Disables specified periods, or instances, of a recurring advance reservation. The *start_date* and *end_date* represent the start and end date of a period in which the reservation should be disabled. These periods must take one of the following forms:

- yyyy:mm:dd-yyyy:mm:dd
- mm:dd-mm:dd - current year is assumed
- dd-dd - current month and year are assumed

The start date must be the same as or earlier than the end date.

If a reservation is disabled for a given day, then it does not become active on that day, and remains inactive for the duration of the reservation time window. Non-recurring reservations are able to use slots of the recurring reservation for the duration of the time window.The -tn option is a shortcut that disables a reservation on the starting day of the next instance of the reservation time window; that is, the instance that starts nearest in the future. If the reservation has already been disabled for this day, the modification request is rejected.

For example, for a weekly reservation with time window from Wednesday 9 a.m. to Friday 10 p.m, if the current day is Monday, then running the command with the -tn option disables the reservation from Wednesday to Friday of the current week. However, if the current day is Thursday, then the reservation is disabled from Wednesday to Friday of the following week. If it is Wednesday, then whether to disable in the current week or following week depends on whether or not the start time of the instance has passed: if not then the reservation is disabled in the current week, otherwise the following week's reservation is disabled.

Running the disable command with the -tn option twice on Monday tries to disable twice in the current week. The second run has no effect, but is rejected because the specified reservation instance is already disabled.

Once a reservation is disabled for a period, it cannot be enabled; that is, the disabled periods remain fixed. Before a reservation is disabled, you are prompted to confirm whether to continue disabling the reservation. Use the -f option to silently force the command to run without prompting for confirmation; for example, to allow for automating disabling reservations from a script.

rmhost [-n *job_slots*] [-m "*host_name* ... | *host_group* ..."] *reservation_ID*

Removes hosts or slots on hosts from the original reservation allocation. You must specify either -n or -m. Use -n to specify the number of slots to be released from reserved hosts. Removing a host without -n releases all reserved free slots on the host. The new slot specification must be less than or equal to the actual reserved slot number of the host.

You can only remove a whole host from a system AR.

How many slots or hosts can be removed depends on the number of slots that are free as long as the reservation is active. `rmhost` cannot remove more slots than are free on the host. This applies to removing hosts on both one-time and recurring reservations that are active. If you want to remove more slots from the reservation, you must wait until running jobs finish or the reservation is inactive.

Options

-o

Changes a closed advance reservation to open, or cancels an open reservation.

Changes the type of a reservation to be open or closed. If the reservation is open, all jobs in the reservation become normal jobs, not subject to termination when the reservation window closes. -on closes the reservation when it expires.The running jobs of an open reservation are terminated when the reservation is changed into closed. The termination times of running jobs of a closed reservation are removed if the reservation is changed to open. The termination time of running jobs is set by `mbatchd` but checked by `sbatchd`. Termination time is an absolute time based on master host, so all hosts in the cluster should be synchronized with the local time on the master host. If sbatchd and mbatchd are not synchronized, termination may not occur at the correct time.

-b *begin_time* | [+ | -]*minutes*

Replaces the begin time for a one-time reservation, or gives an offset in minutes to the current begin time.

Restriction:

You cannot modify the begin time of an active reservation.

The begin time is in the form

```
[[[year:]month:]day:]hour:minute
```

with the following ranges:

- *year*: any year after 1900 (YYYY)
- *month*: 1-12 (MM)
- *day of the month*: 1-31 (dd)
- *hour*: 0-23 (hh)
- *minute*: 0-59 (mm)

You must specify at least *hour*:*minute*. Year, month, and day are optional. Three fields are assumed to be *day*:*hour*:*minute*, four fields are assumed to be *month*:*day*:*hour*:*minute*, and five fields are *year*:*month*:*day*:*hour*:*minute*.

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The offset is in minutes, an integer with a prefix+ or -. For example, `-b+5` moves the begin time 5 minutes later, and `-b-5` moves the begin time 5 minutes earlier.

The modified time value for -b must use the same syntax as the time value for -e. It must be earlier than the time value for -e, and cannot be earlier than the current time.

-d "*description*"

Replaces or sets a description for the reservation. The description must be provided as a double quoted text string. The maximum length is 512 characters.

-e *end_time* | [+ | -]*minutes*

Replaces the end time for a one-time reservation, or gives an offset in minutes to the current end time.

By giving a positive offset to the end time, you extend the duration of a reservation so that the jobs in the reservation can run longer. Shrinking the reservation with a negative value terminates running jobs earlier.

The end time is in the form

```
[[[year:]month:]day:]hour:minute
```

with the following ranges:

- *year*: any year after 1900 (YYYY)
- *month*: 1-12 (MM)
- *day of the month*: 1-31 (dd)
- *hour*: 0-23 (hh)
- *minute*: 0-59 (mm)

You must specify at least *hour*:*minute*. Year, month, and day are optional. Three fields are assumed to be *day*:*hour*:*minute*, four fields are assumed to be *month*:*day*:*hour*:*minute*, and five fields are *year*:*month*:*day*:*hour*:*minute*.

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The time value for -e must use the same syntax as the time value for -b. It must be later than the time value for -b.

-g *group_name*

Changes the user group that is able to submit jobs to the reservation. Changing the user group does not affect the currently running jobs.

Jobs submitted by the original user group to the reservation still belong to the reservation and scheduled as advance reservation jobs, but newly submitted jobs from a user group that has been removed from the reservation cannot use the reservation any longer.

The -g *group_name* option does not support the @cluster notation for advance reservations on remote clusters.

-m "*host_name* | *host_group* ..."

Changes the list of hosts for which job slots specified with -n are reserved. At job submission, LSF considers the hosts in the specified order.

If you also specify a resource requirement string with the -R option, -m is optional.

The hosts can be local to the cluster or hosts leased from remote clusters.

-n *job_slots*

Changes the number of job slots to reserve. *job_slots* must be less than or equal to the actual number of slots for the hosts selected by -m or -R for the reservation.

If you also specify the reservation for system use with the -s option, -n is optional.

-R "*res_req*"

Changes the host selection for the reservation according to the specified resource requirements. Only hosts that satisfy the resource requirement expression are reserved. -R accepts any valid resource requirement string, but only the select string takes effect.

If you also specify a host list with the -m option, -R is optional.

For more information about resource requirements, see *Administering Platform LSF*.

The size of the resource requirement string is limited to 512 bytes.

-t *time_window*

Replaces the time window with a new one to shift a recurring reservation. You cannot modify the start time of a recurring reservation that has current active instance.

To specify a time window, specify two time values separated by a hyphen (-), with no space in between:

```
time_window = begin_time-end_time
```

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23
- *minute*: 0-59

Specify a time window one of the following ways:

- *hour-hour*
- *hour*:*minute-hour*:*minute*
- *day*:*hour*:*minute-day*:*hour*:*minute*

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute,

LSF assumes the first minute of the hour (`:00`). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

LSF administrators can prevent running jobs from being killed when the reservation expires by changing the termination time of the job using the reservation (`bmod -t`) before the reservation window closes.

When the job starts running, the run limit of the reservation is set to the minimum of the job run limit (if specified), the queue run limit (if specified), or the duration of the time window.

-u *user_name*

Changes the user who is able to submit jobs to the reservation. Changing user does not affect the currently running jobs.

Jobs submitted by the original user to the reservation still belong to the reservation and scheduled as advance reservation jobs, but newly submitted jobs from users that have been removed from the reservation cannot use the reservation any longer.

The -u *user_name* option does not support the @cluster notation for advance reservations on remote clusters.

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

Examples

The following command adds a host to an existing reservation.

```
brsvmod -addhost hostB user1#0
HostB is added to reservation "user1#0".
```

The following example disables the advanced reservation between January 1 and January 6, 2008, inclusive.

```
brsvmod disable {-td "2008:01:01-2008:01:06"}
```

See also

brsvadd, brsvdel, brsvs, `lsb.resources`

brsvs

displays advance reservations

Synopsis

brsvs [**-l** | **-w**] [**-p all** | **"***host_name* ...**"**]

brsvs [**-l** | **-w**] [**-z all** | **"***host_name* ...**"**]

brsvs [**-c all** | **"***policy_name***"**]

brsvs [**-h** | **-V**]

Description

By default, displays the current advance reservations for all hosts, users, and groups.

For advance reservations across clusters:

- -p all shows local and all remote reservations
- The default all includes both local and remote
- *host_name* does NOT take *host_name@cluster_name*

By default, `brsvs` truncates the reservation ID (RSVID) at 11 characters. Use -w to see the full reservation ID.

Options

-l

Displays advance reservations in a long multiline format. In addition to the standard output, the -l option displays the reservation type (open or closed) and the job IDs of any jobs associated with the specified advance reservation, sorted by status.

-w

Wide format. Displays reservation information without truncating fields.

-c all | "*policy_name* ..."

Shows advance reservation policies defined in `lsb.resources`. By default, displays all policy names.

The all keyword shows detailed information for all policies.

-p all | "*host_name* ..."

Shows a weekly planner for specified hosts using advance reservations.

The all keyword shows a weekly planner for all hosts with reservations.

-z all | "*host_name*"

Show a planner with only the weekly items that have reservation configurations displayed. Empty lines are omitted.

The all keyword shows a weekly planner for all hosts with reservations.

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

Output

TIME_WINDOW

Time window for a recurring reservation.

Values are separated by a hyphen (-), with no space in between:

```
time_window = begin_time-end_time
```

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23
- *minute*: 0-59

A time window can be specified in any of the following ways:

- *hour-hour*
- *hour*:*minute-hour*:*minute*
- *day*:*hour*:*minute-day*:*hour*:*minute*

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

Example

```
brsvs -c reservation1
Policy Name: reservation1
Users: ugroup1 ~user1
Hosts: hostA hostB
Time Window: 8:00-13:00
```

See also

brsvadd, brsvdel, brsvmod, `lsb.resources`

brun

forces a job to run immediately

Synopsis

brun [**-b**] [**-c**] [**-f**] **-m "***host_name*[**#***num_cpus*] ... **"** *job_ID*

brun [**-b**] [**-c**] [**-f**] **-m "***host_name*[**#***num_cpus*] ... **" "***job_ID***[***index_list***]"**

brun [**-h** | **-V**]

Description

Caution:

This command can only be used by LSF administrators.

Forces a pending job to run immediately on specified hosts.

A job that has been forced to run is counted as a running job, this may violate the user, queue, or host job limits, and fairshare priorities. The forced job can run on hosts with an exclusive resource definition.

A job that has been forced to run cannot be preempted by other jobs even if it is submitted to a preemptable queue and other jobs are submitted to a preemptive queue.

By default, after the job is started, it is still subject to run windows and suspending conditions.

LSF administrators can use `brun` to force jobs with an advance reservation to run before the reservation is active, but the job must finish running before the time window of the reservation expires.

For example, if the administrator forces a job with a reservation to run one hour before the reservation is active, and the reservation period is 3 hours, a 4 hour run limit takes effect.

Options

-b

Causes a checkpointable job to start over from the beginning, as if it had never been checkpointed.

-c

Distribute job slots for a multihost parallel job according to free CPUs.

By default, if a parallel job spans for more than one host, LSF distributes the slots based on the static CPU counts of each host listed in the -m option. Use -c to distribute the slots based on the free CPUs of each host instead of the static CPUs.

The -c option can be only applied to hosts whose total slot counts equal to their total CPU counts. MXJ in `lsb.hosts` must be less than or equal to the number of CPUs and PJOB_LIMIT=1 must be specified in the queue (`lsb.queues`).

For example, a 6-CPU job is submitted to `hostA` and `hostB` with 4 CPUs each. Without -c, LSF would let the job take 4 slots from `hostA` first and then take 2 slots from `hostB` regardless to the status or the slots usage on `hostA` and `hostB`. If any slots on `hostA` are used, the job remains pending. With -c, LSF takes into consideration that

`hostA` has 2 slots in use and `hostB` is completely free, so LSF is able to dispatch the job using the 2 free slots on `hostA` and all 4 slots on `hostB`.

-f

Allows the job to run without being suspended due to run windows or suspending conditions.

-m "*host_name*[*#num_cpus*] ... "

Required. Specify one or more hosts on which to run the job.

You can optionally specify the number of CPUs required per host for multihost parallel jobs. The *#num_cpus* option distributes job slots according the number of CPUs on the host. If *#num_cpus* is not defined, or if *#num_cpus* is greater than the number of static CPUs on the host (or the number of free CPUs if -c is specified), LSF distributes job slots according to the number of static CPUs on the host, or the number of free CPUs on the host if -c is specified. The number sign (#) is required as a prefix to the number of CPUs. The square brackets ([ ])indicate that *#num_cpus* is optional. Do not include them in the command.

For example, the following command forces job 123 to run and specifies 1 CPU on hostA and 1 CPU on hostB:

```
brun -m "hostA#1 hostB#1" 123
```

***job_ID* | "*job_ID*[*index_list*]"**

Required. Specify the job to run, or specify one element of a job array.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Limitations

You cannot force a job in SSUSP or USUSP state.

`brun` does not guarantee a job runs; it just forces LSF to dispatch the job.

In the MultiCluster job forwarding model, you can only force a job by running the command in the execution cluster.

bsla

displays information about service class configuration for goal-oriented service-level agreement (SLA) scheduling

Synopsis

bsla [*service_class_name*]

bsla [**-h** | **-V**]

Description

`bsla` displays the properties of service classes configured in `lsb.serviceclasses` and dynamic information about the state of each configured service class.

If a default system service class is configured with ENABLE_DEFAULT_EGO_SLA in `lsb.params` but no other service classes are explicitly configured in `lsb.serviceclasses`, `bsla` only displays information for the default SLA.

Options

service_class_name

The name of a service class configured in `lsb.serviceclasses`.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

A list of job groups is displayed with the following fields:

SERVICE CLASS NAME

The name of the service class, followed by its description, if any.

PRIORITY

The service class priority. A higher value indicates a higher priority, relative to other service classes. Similar to queue priority, service classes access the cluster resources in priority order.

CONSUMER

For EGO-enabled SLA service classes, the name of the EGO consumer from which hosts are allocated to the SLA.

EGO_RES_REQ

For EGO-enabled SLA service classes, the EGO resource requirement defined in the SLA.

MAX_HOST_IDLE_TIME

For EGO-enabled SLA service classes, how long the SLA holds its idle hosts before LSF releases them to EGO.

NUM_RECALLED_HOSTS

For EGO-enabled SLA service classes, the number of hosts allocated to the SLA that EGO has reclaimed.

RECALLED_HOSTS_TIMEOUT

For EGO-enabled SLA service classes, the amount of time EGO gives to LSF to clean up its workload before EGO reclaims the host.

USER GROUP

User names or user groups who can submit jobs to the service class.

GOAL

The type of service class goal and its configured value:

- THROUGHPUT
- VELOCITY
- DEADLINE

ACTIVE WINDOW

The configured time window when the service class goal is active. If a throughput or velocity goal has no time window configured, ACTIVE WINDOW is Always Open.

STATUS

Current status of the service class goal:

- Active:On time — the goal is active and meeting its target.
- Active:Delayed — the goal is active but is missing its target.
- Inactive — the goal is not active; its time window is closed. Jobs are scheduled as if no service class is defined. LSF does not enforce any service-level goal for an inactive SLA.

THROUGHPUT

For throughput goals, the configured job throughput (finished jobs per hour) for the service class.

SLA THROUGHPUT

The current throughput for the SLA finished jobs per clean period.

ESTIMATED FINISH TIME

For goals with a time window, estimated finish time of the SLA. If the service class status is on time, the finish time is before the configured deadline. If the service class status is delayed, the service class is missing its goal and `bsla` shows a finish time later than the deadline.

OPTIMUM NUMBER OF RUNNING JOBS

For goals with a time window, the optimum number of jobs that should be running in the service class for the SLA to meet its goal.

NJOBS

The current number of jobs in the specified service class. A parallel job is counted as 1 job, regardless of the number of job slots it uses.

PEND

The number of pending jobs in the specified service class.

RUN

The number of running jobs in the specified service class.

SSUSP

The number of system-suspended jobs in the service class.

USUSP

The number of user-suspended jobs in the specified service class.

FINISH

The number of jobs in the specified service class in EXITED or DONE state.

Example

The following service class named `Kyuquot` is configured in `lsb.serviceclasses`:

```
Begin ServiceClass
NAME = Kyuquot
CONSUMER = Kyuquot
PRIORITY = 23
USER_GROUP = user1 user2
GOALS = [VELOCITY 8 timeWindow (9:00-17:30)] \
[DEADLINE timeWindow (17:30-9:00)]
DESCRIPTION = Daytime/Nighttime SLA
End ServiceClass
```

`bsla` shows the following properties and current status:

bsla Kyuquot

```
SERVICE CLASS NAME:  Kyuquot
 -- Daytime/Nighttime SLA
PRIORITY:  23
CONSUMER: Kyuquot
EGO_RES_REQ: any host
MAX_HOST_IDLE_TIME: 120
USER_GROUP:  user1 user2

GOAL:  VELOCITY 8
ACTIVE WINDOW: (9:00-17:30)
STATUS:  Active:On time
SLA THROUGHPUT:  0.00 JOBS/CLEAN_PERIOD
GOAL:  DEADLINE
ACTIVE WINDOW: (17:30-9:00)
STATUS:  Inactive
SLA THROUGHPUT:  0.00 JOBS/CLEAN_PERIOD

 NJOBS   PEND    RUN     SSUSP   USUSP   FINISH
      0      0       0         0       0       0
```

See also

bsla(1), bhist(1), bjobs(1), bkill(1), bmod(1), bsub(1), lsb.acct(5), lsb.serviceclasses(5)

bslots

displays slots available and backfill windows available for backfill jobs

Synopsis

bslots [**-l**] [**-n** *slots*] [**-R "***res_req***"**] [**-W** [*hour*:]*minutes*]

bslots [**-h** | **-V**]

Description

The available slots displayed by `bslots` are not currently used for running jobs and can be used for backfill jobs. `bslots` displays a snapshot of the slots currently not in use by parallel jobs or advance reservations. They are not guaranteed to be available at job submission.

By default, displays all available slots, and the available run times (backfill windows) for those slots. When no slots are available for backfill, `bslots` displays

```
No backfill window exists at this time
```

bslots calculates the backfill window based on the estimated start time of potential backfill jobs. Estimated start time is only relatively accurate according to current running job information. If running jobs finish earlier or later, estimated start time may be moved to earlier or later time. There may be a small delay of a few minutes between the job finish time on which the estimate was based and the actual start time of the allocated job.

If the available backfill window has no run time limit, its length is displayed as UNLIMITED.

Options

-l

Displays backfill windows in a long multi-line format. The `-l` option displays host names and the number of slots on each host available for backfill.

-n *slots*

Specifies required slots (processors). Backfill windows whose widths are equal or larger than specified value are returned.

When no slots are available for backfill, `bslots -n` displays

```
No backfill window meets these requirements at this time
```

-R "*res_req*"

Selects hosts for calculating the backfill windows according to the specified resource requirement. By default, selects hosts of any type. The `-R` option only supports the select resource requirement string. Other resource requirement sections are not supported.

If LSF_STRICT_RESREQ=y in `lsf.conf`, the selection string must conform to the stricter resource requirement string syntax described in *Administering Platform LSF*. The strict resource requirement syntax only applies to the select section.

When no slots are available for backfill, `bslots -R` displays

```
No backfill window meets these requirements at this time
```

-W [*hour*:]*minutes*

Specifies expected runtime limit. Backfill windows whose lengths are equal or larger than specified value are returned.

When no slots are available for backfill, `bslots -W` displays

```
No backfill window meets these requirements at this time
```

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

bstatus

gets current external job status or sets new job status

Synopsis

bstatus [**-d "***description***"**] *job_ID* | **"***job_ID***[***index***]"** | **-J** *job_name*

bstatus [**-h** | **-V**]

Description

Gets and displays the message description text of a job, or changes the contents of the message description text with the -d option. Always operates on the message with index 0.

You can set the external status of a job until it completes. You cannot change the status of done or exited jobs. You can display the status of a job until it is cleaned from the system.

If a you specify a job ID:

- You can get the external job status of jobs submitted by other users, but you cannot set job status of jobs submitted by other users.
- You can only set external status on your own jobs.
- Only root and LSF administrators can set external job status on jobs submitted by other users.

Job names are not unique; if you specify `-J` *job_name*:

- You can only get or set the external status on your own jobs.
- You cannot get or set external job status on jobs submitted by other users.
- Root and the LSF administrators can only get or set the external status on their own jobs.

Options

-d "*description*"

Updates the job status with specified message description text.

job_ID* | "*job_ID*[*index*]" | -J *job_name

Required. Operates on the specified job.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

bstatus 2500

```
JOBID     FROM    UPDATE_TIME   STATUS
2500      user1   Sep 14 16:54  step 1
```

Displays the message description text of message index 0 of job 2500.

bstatus -d "step 2" 2500

Changes the message description text of message index 0 of job 2500 to `step 2`.

See also

bpost(1), bread(1)

bstop

suspends unfinished jobs

Synopsis

bstop [**-a**] [**-app** *application_profile_name*] [**-g** *job_group_name*] [**-sla** *service_class_name*] [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [**0**] [*job_ID* ... | **"***job_ID* **[***index***]"**] ...

bstop [**-h** | **-V**]

Description

Suspends unfinished jobs.

Sends the SIGSTOP signal to sequential jobs and the SIGTSTP signal to parallel jobs to suspend them.

You must specify a job ID or -g, -J, -m, -u, or -q. You cannot suspend a job that is already suspended. Specify job ID 0 (zero) to stop multiple jobs.

Only root and LSF administrators can operate on jobs submitted by other users.

Use `bresume` to resume suspended jobs.

An administrator can use `bstop` on a job stopped by the user (in the state USUSP) to prevent the user from resuming the job.

You can also use `bkill -s STOP` to send the suspend signal to a job or use `bkill -s TSTP` to suspend one or more parallel jobs. Use `bkill -s CONT` to send a resume signal to a job.

If a signal request fails to reach the job execution host, LSF retries the operation later when the host becomes reachable. LSF retries the most recent signal request.

Options

0

Suspends all the jobs that satisfy other options (-g, -m, -q, -u, and -J).

-a

Suspends all jobs.

-app *application_profile_name*

Suspends only jobs associated with the specified application profile. You must specify an existing application profile.

-g *job_group_name*

Suspends only jobs in the specified job group.

-J *job_name*

Suspends only jobs with the specified name.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]` returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-m *host_name* | -m *host_group*

Suspends only jobs dispatched to the specified host or host group.

-q *queue_name*

Suspends only jobs in the specified queue.

-sla *service_class_name*

Suspends jobs belonging to the specified service class.

Use `bsla` to display the properties of service classes configured in `LSB_CONFDIR/`*cluster_name*`/configdir/lsb.serviceclasses` (see `lsb.serviceclasses`(5)) and dynamic information about the state of each configured service class.

-u *user_name* | -u *user_group* | -u all

Suspends only jobs owned by the specified user or user group, or all users if the keyword all is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

***job_ID* ... | "*job_ID*[*index*]" ...**

Suspends only the specified jobs. Jobs submitted by any user can be specified here without using the `-u` option.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

bstop 314

Suspends job number 314.

bstop -m hostA

Suspends the invoker's last job that was dispatched to host `hostA`.

bstop -u jsmith 0

Suspends all the jobs submitted by user `jsmith`.

bstop -u all

Suspends the last submitted job in the LSF system.

bstop -u all 0

Suspends all jobs for all users in the LSF system.

bstop -g /risk_group/consolidate 0

Suspends all jobs in the job group `/risk_group/consolidate`.

bstop -app fluent 0

Suspends all jobs associated with the application profile `fluent`.

See also

bsub(1), bjobs(1), bqueues(1), bhosts(1), bresume(1), bkill(1), bapp(1), bgadd(1), bgdel(1), bjgroup(1), bparams(5), mbatchd(8), `kill`(1), `signal`(2) lsb.params(5)

bsub

submits a batch job to LSF

Synopsis

bsub [*options*] *command* [*arguments*]

bsub [**-h** | **-V**]

Option List

-**ar**

-**B**

-H

-I | **-Ip** | **-Is** [**-tty**]

-IS | **-ISp** | **-ISs** | **-IX** [**-tty**]

-K

-N

-r | **-rn**

-ul

-x

-a "*esub_application ...***"**

-app *application_profile_name*

-b [[*year*:][*month*:]*day*:]*hour*:*minute*

-C *core_limit*

-c [*hour*:]*minute*[**/***host_name* | **/***host_model*]

-cwd "*current_working_directory***"**

-D *data_limit*

-E "*pre_exec_command* [*arguments ...*]**"**

-Ep "*post_exec_command* [*arguments …*]**"**

-e *error_file*

-eo *error_file*

-ext[**sched**] **"***external_scheduler_options***"**

-F *file_limit*

-f "*local_file operator* [*remote_file*]**"** ...

-G *user_group*

-g *job_group_name*

-i *input_file* | **-is** *input_file*

-J *job_name* | **-J "***job_name***[***index_list***]%***job_slot_limit***"**

-jsdl *file_name* | **-jsdl_strict** *file_name*

-k "*checkpoint_dir* [**init=***initial_checkpoint_period*]

[*checkpoint_period*] [**method=***method_name*]**"**

-L *login_shell*

-Lp *ls_project_name*

-M *mem_limit*

-m *"host_name*[*@cluster_name*][[**!**] | **+**[*pref_level*]] | *host_group*[[**!**] | **+**[*pref_level* | *compute_unit*[[**!**] | **+**[*pref_level*]] ...**"**

-mig *migration_threshold*

-n *min_proc*[**,***max_proc*]

-o *output_file*

-oo *output_file*

-P *project_name*

-p *process_limit*

-Q "[*exit_code …*] [**EXCLUDE(***exit_code …***)**]**"**

-q "*queue_name ...***"**

-R "*res_req***"** [**-R** "*res_req*" *…*]

-rnc *resize_notification_cmd*

-S *stack_limit*

-s *signal*

-sla *service_class_name*

-sp *priority*

-T *thread_limit*

-t [[[year:]*month***:**]*day***:**]*hour***:***minute*

-U *reservation_ID*

-u *mail_user*

-v *swap_limit*

-W [*hour***:**]*minute*[**/***host_name* | **/***host_model*]

-We [*hour***:**]*minute*[**/***host_name* | **/***host_model*]

-w '*dependency_expression***'**

-wa '*signal***'**

-wt '[*hour***:**]*minute***'**

-Zs

-h

-V

Description

Submits a job for batch execution and assigns it a unique numerical job ID.

Runs the job on a host that satisfies all requirements of the job, when all conditions on the job, host, queue, application profile, and cluster are satisfied. If LSF cannot run all jobs immediately, LSF scheduling policies determine the order of dispatch. Jobs are started and suspended according to the current system load.

Sets the user's execution environment for the job, including the current working directory, file creation mask, and all environment variables, and sets LSF environment variables before starting the job.

When a job is run, the command line and `stdout`/`stderr` buffers are stored in the directory home_directory`/.lsbatch` on the execution host. If this directory is not accessible, `/tmp/.lsbtmp` user_ID is used as the job's home directory. If the current working directory is under the home directory on the submission host, then the current working directory is also set to be the same relative directory under the home directory on the execution host.

By default, if the current working directory is not accessible on the execution host, LSF finds a working direction to run the job in the following order:

1. *$HOME* on this host
2. *$PWD*
3. Strip `/tmp_mnt` if it is exists in the path
4. Replace the first component with a key in `/etc/auto.master` and try each key
5. Replace the first 2 components with a key in `/etc/auto.master` and try for each key
6. Strip the first level of the path and try the rest (for example, if the current working directory is `/abc/x/y/z`, try to change directory to the path `/x/y/z`)
7. `/tmp`

If the environment variable LSB_EXIT_IF_CWD_NOTEXIST is set to Y and the current working directory is not accessible on the execution host, the job exits with the exit code 2.

If no command is supplied, `bsub` prompts for the command from the standard input. On UNIX, the input is terminated by entering CTRL-D on a new line. On Windows, the input is terminated by entering CTRL-Z on a new line.

To kill a batch job submitted with `bsub`, use `bkill`.

Use `bmod` to modify jobs submitted with `bsub`. `bmod` takes similar options to `bsub`.

Jobs submitted to a chunk job queue with the following options are not chunked; they are dispatched individually:

- `-I` (interactive jobs)
- `-c` (jobs with CPU limit greater than 30)
- `-W` (jobs with run limit greater than 30 minutes)

To submit jobs from UNIX to display GUIs through Microsoft Terminal Services on Windows, submit the job with bsub and define the environment variables LSF_LOGON_DESKTOP=1 and LSB_TSJOB=1 on the UNIX host. Use `tssub` to submit a Terminal Services job from Windows hosts. See *Using Platform LSF on Windows* for more details.

If the parameter LSB_STDOUT_DIRECT in `lsf.conf` is set to Y or y, and you use the -o or -oo option, the standard output of a job is written to the file you specify as the job runs. If LSB_STDOUT_DIRECT is not set, and you use -o or -oo, the standard output of a job is written to a temporary file and copied to the specified file after the job finishes. LSB_STDOUT_DIRECT is not supported on Windows.

Default behavior

LSF assumes that uniform user names and user ID spaces exist among all the hosts in the cluster. That is, a job submitted by a given user runs under the same user's account on the execution host. For situations where nonuniform user names and user ID spaces exist, account mapping must be used to determine the account used to run a job.

`bsub` uses the command name as the job name. Quotation marks are significant.

Options related to file names and job spooling directories support paths that contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

Options related to command names and job names can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

Options for the following resource usage limits are specified in KB:

- Core limit (`-C`)
- Memory limit (`-M`)
- Stack limit (`-S`)
- Swap limit (`-v`)

Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

If fairshare is defined and you belong to multiple user groups, the job is scheduled under the user group that allows the quickest dispatch.

The job is not checkpointable.

`bsub` automatically selects an appropriate queue. If you defined a default queue list by setting LSB_DEFAULTQUEUE environment variable, the queue is selected from your list. If LSB_DEFAULTQUEUE is not defined, the queue is selected from the system default queue list specified by the LSF administrator with the DEFAULT_QUEUE parameter in `lsb.params`.

LSF tries to obtain resource requirement information for the job from the remote task list that is maintained by the load sharing library. If the job is not listed in the remote task list, the default resource requirement is to run the job on a host or hosts that are of the same host type as the submission host.

`bsub` assumes only one processor is requested.

`bsub` does not start a login shell but runs the job file under the execution environment from which the job was submitted.

The input file for the batch job is `/dev/null` (no input).

`bsub` sends mail to you when the job is done. The default destination is defined by LSB_MAILTO in `lsf.conf`. The mail message includes the job report, the job output (if any), and the error message (if any).

`bsub` charges the job to the default project. The default project is the project you define by setting the environment variable LSB_DEFAULTPROJECT. If you do not set LSB_DEFAULTPROJECT, the default project is the project specified by the LSF administrator with DEFAULT_PROJECT parameter in `lsb.params`. If DEFAULT_PROJECT is not defined, then LSF uses default as the default project name.

Options

-ar

Specifies that the job is autoresizable.

-B

Sends mail to you when the job is dispatched and begins execution.

-H

Holds the job in the PSUSP state when the job is submitted. The job is not scheduled until you tell the system to resume the job (see `bresume(1)`).

-I | -Ip | -Is [-tty]

Submits a batch interactive job. A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the -N option.

Terminal support is available for a batch interactive job.

When you specify the -Ip option, submits a batch interactive job and creates a pseudo-terminal when the job starts. Some applications (for example, `vi`) require a pseudo-terminal in order to run correctly.

When you specify the -Is option, submits a batch interactive job and creates a pseudo-terminal with shell mode support when the job starts. This option should be specified for submitting interactive shells, or applications which redefine the CTRL-C and CTRL-Z keys (for example, `jove`).

If the -i *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the -e *err_file* option is specified, sends the job's standard error to the specified error file.

If used with -tty, also displays output/error on the screen.

You cannot use -I, -Ip, or -Is with the -K option.

Interactive jobs cannot be checkpointed.

Interactive jobs cannot be rerunnable (`bsub -r`).

The options that create a pseudo-terminal (-Ip and -Is) are not supported on Windows.

-IS | -ISp | -ISs | -IX [-tty]

Submits a batch interactive job under a secure shell (`ssh`). A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the -N option.

Terminal support is available for a batch interactive job.

When you specify the -ISp option, submits a batch interactive job and creates a pseudo-terminal when the job starts. Some applications (for example, `vi`) require a pseudo-terminal to run correctly.

When you specify the -ISs option, submits a batch interactive job and creates a pseudo-terminal with shell mode support when the job starts. This option should be specified for submitting interactive shells, or applications that redefine the CTRL-C and CTRL-Z keys (for example, `jove`).

When you specify the -IX option, submits an interactive X-window job. The session between X-client and X-server is encrypted; the session between the execution host and submission host is also encrypted. The following must be satisfied:

- openssh must be installed and sshd must be running on the X-server
- `xhost + localhost` or `xhost + displayhost.domain.com` on the X-server
- ssh must be configured to run without a password or passphrase (`$HOME/.ssh/authorized_keys` must be set up)

Note:

In most cases ssh can be configured to run without a password by copying `id_rsa.pub` as `authorized_keys` with permission 600 (-rw-r--r--). Test by manually running `ssh host.domain.com` between the two hosts both ways and confirm there are no prompts using fully qualified host names.

If the -i *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the -e *err_file* option is specified, sends the job's standard error to the specified error file.

If used with -tty, also displays output/error on the screen.

You cannot use -I, -ISp, or -ISs with the -K option.

Interactive jobs cannot be checkpointed.

Interactive jobs cannot be rerunnable (`bsub -r`).

The options that create a pseudo-terminal (-ISp and -ISs) are not supported on Windows.

Optionally, you can enable SSH in `lsf.conf` to encrypt the communication for interactive jobs.

-K

Submits a batch job and waits for the job to complete. Sends the message "`Waiting for dispatch`" to the terminal when you submit the job. Sends the message "`Job is finished`" to the terminal when the job is done.

You are not able to submit another job until the job is completed. This is useful when completion of the job is required to proceed, such as a job script. If the job needs to be rerun due to transient failures, `bsub` returns after the job finishes successfully. `bsub` exits with the same exit code as the job so that job scripts can take appropriate actions based on the exit codes. `bsub` exits with value 126 if the job was terminated while pending.

You cannot use the -K option with the -I, -Ip, or -Is options.

-N

Sends the job report to you by mail when the job finishes. When used without any other options, behaves the same as the default.

Use only with -o, -oo, -I, -Ip, and -Is options, which do not send mail, to force LSF to send you a mail message when the job is done.

-r | -rn

If the execution host becomes unavailable while a job is running, specifies that the job be rerun on another host. LSF requeues the job in the same job queue with the same job ID. When an available execution host is found, reruns the job as if it were submitted new, even if the job has been checkpointed. You receive a mail message informing you of the host failure and requeuing of the job.

If the system goes down while a job is running, specifies that the job is requeued when the system restarts.

Reruns a job if the execution host or the system fails; it does not rerun a job if the job itself fails.

-rn specifies that the job is never rerunnable. `bsub –rn` disables job rerun if the job was submitted to a rerunnable queue or application profile with job rerun configured. The command level job rerunnable setting overrides the application profile and queue level setting. `bsub –rn` is different from `bmod -rn`, which cannot override the application profile and queue level rerunnable job setting.

Members of a chunk job can be rerunnable. If the execution host becomes unavailable, rerunnable chunk job members are removed from the queue and dispatched to a different execution host.

Interactive jobs (`bsub -I`) cannot be rerunnable.

-ul

Passes the current operating system user shell limits for the job submission user to the execution host. User limits cannot override queue hard limits. If user limits exceed queue hard limits, the job is rejected.

Restriction:

UNIX and Linux only. -ul is not supported on Windows.

The following `bsub` options for job-level runtime limits override the value of the user shell limits:

- Per-process (soft) core file size limit (-C)
- CPU limit (-c)
- Per-process (soft) data segment size limit (-D)
- File limit (-F)
- Per-process (soft) memory limit (-M)
- Process limit (-p)
- Per-process (soft) stack segment size limit (-S)
- Limit of the number of concurrent threads (-T)

- Total process virtual memory (swap space) limit (-v)
- Runtime limit (-W)

LSF collects the user limit settings from the user's running environment that are supported by the operating system, and sets the value to submission options if the value is no unlimited. If the operating system has other kinds of shell limits, LSF does not collect them. LSF collects the following operating system user limits:

- CPU time in milliseconds
- Maximum file size
- Data size
- Stack size
- Core file size
- Resident set size
- Open files
- Virtual (swap) memory
- Process limit
- Thread limit

-x

Puts the host running your job into exclusive execution mode.

In exclusive execution mode, your job runs by itself on a host. It is dispatched only to a host with no other jobs running, and LSF does not send any other jobs to the host until the job completes.

To submit a job in exclusive execution mode, the queue must be configured to allow exclusive jobs.

When the job is dispatched, `bhosts(1)` reports the host status as closed_Excl, and `lsload(1)` reports the host status as lockU.

Until your job is complete, the host is not selected by LIM in response to placement requests made by `lsplace(1)`, `lsrun(1)` or `lsgrun(1)` or any other load sharing applications.

You can force other batch jobs to run on the host by using the -m *host_name* option of `brun(1)` to explicitly specify the locked host.

You can force LIM to run other interactive jobs on the host by using the -m *host_name* option of `lsrun(1)` or `lsgrun(1)` to explicitly specify the locked host.

-a "*esub_application* ..."

Specifies one or more application-specific `esub` executables that you want LSF to associate with the job.

The value of -a must correspond to the application name of an actual `esub` file. For example, to use `bsub -a fluent`, the file `esub.fluent` must exist in LSF_SERVERDIR.

For example, to submit a job that invokes two application-specific `esub` executables named `esub.license` and `esub.fluent`, enter:

```
bsub -a "license fluent" my_job
```

`mesub` uses the method name license to invoke the `esub` named `LSF_SERVERDIR/esub.license`, and the method name fluent to invoke the `esub` named `LSF_SERVERDIR/esub.fluent`.

The name of an application-specific `esub` program is passed to the master `esub`. The master `esub` program (LSF_SERVERDIR/`mesub`) handles job submission requirements of the application. Application-specific `esub` programs can specify their own job submission requirements. The value of -a is set in the LSB_SUB_ADDITIONAL option in the LSB_SUB_PARM file used by `esub`.

If an LSF administrator specifies one or more mandatory `esub` executables using the parameter LSB_ESUB_METHOD, LSF invokes the mandatory executables first, followed by the executable named `esub` (without *.esub_application* in the file name) if it exists in LSF_SERVERDIR, and then any application-specific `esub` executables (with *.esub_application* in the file name) specified by -a.

The name of the `esub` program must be a valid file name. It can contain only alphanumeric characters, underscore (_) and hyphen (-).

Restriction:

After LSF version 5.1, the value of -a and LSB_ESUB_METHOD must correspond to an actual esub file in LSF_SERVERDIR. For example, to use bsub -a fluent, the file esub.fluent must exist in LSF_SERVERDIR.

Note:

If you have an esub that runs an interactive or X-window job and you have SSH enabled in `lsf.conf`, the communication between hosts is encrypted.

-app *application_profile_name*

Submits the job to the specified application profile. You must specify an existing application profile. If the application profile does not exist in `lsb.applications`, the job is rejected.

-b [[*year*:][*month*:]*day*:]*hour*:*minute*

Dispatches the job for execution on or after the specified date and time. The date and time are in the form of [[*year*:][*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: year after 1970, month 1-12, day 1-31, hour 0-23, minute 0-59.

At least two fields must be specified. These fields are assumed to be hour:minute. If three fields are given, they are assumed to be day:hour:minute, four fields are assumed to be *month*:*day*:*hour*:*minute*, and five fields are assumed to be *year*:*month*:*day*:*hour*:*minute*.

If the year field is specified and the specified time is in the past, the start time condition is considered reached and LSF dispatches the job if slots are available.

-C *core_limit*

Sets a per-process (soft) core file size limit for all the processes that belong to this batch job (see `getrlimit(2)`).

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

The behavior of this option depends on platform-specific UNIX or Linux systems.

In some cases, the process is sent a SIGXFSZ signal if the job attempts to create a core file larger than the specified limit. The SIGXFSZ signal normally terminates the process.

In other cases, the writing of the core file terminates at the specified limit.

-c [*hour*:]*minute*[/*host_name* | /*host_model*]

Limits the total CPU time the job can use. This option is useful for preventing runaway jobs or jobs that use up too many resources. When the total CPU time for the whole job has reached the limit, a SIGXCPU signal is first sent to the job, then SIGINT, SIGTERM, and SIGKILL.

If LSB_JOB_CPULIMIT in `lsf.conf` is set to n, LSF-enforced CPU limit is disabled and LSF passes the limit to the operating system. When one process in the job exceeds the CPU limit, the limit is enforced by the operating system.

The CPU limit is in the form of [*hour*:]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The CPU time you specify is the *normalized* CPU time. This is done so that the job does approximately the same amount of processing for a given CPU limit, even if it is sent to host with a faster or slower CPU. Whenever a normalized CPU time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert a slash (/) between the CPU limit and the host name or model name. If a host name or model name is not given, LSF uses the default CPU time normalization host defined at the queue level (DEFAULT_HOST_SPEC in `lsb.queues`) if it has been configured, otherwise uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in `lsb.params`) if it has been configured, otherwise uses the submission host.

Jobs submitted to a chunk job queue are not chunked if the CPU limit is greater than 30 minutes.

-cwd "*current_working_directory*"

Specifies the current working directory for the job.

By default, if the current working directory is not accessible on the execution host, the job runs in `/tmp`. If the environment variable LSB_EXIT_IF_CWD_NOTEXIST is set to Y and the current working directory is not accessible on the execution host, the job exits with the exit code 2.

-D *data_limit*

Sets a per-process (soft) data segment size limit for each of the processes that belong to the batch job (see `getrlimit`(2)). The limit is specified in KB.

This option affects calls to `sbrk()` and `brk()`. An `sbrk()` or `malloc()` call to extend the data segment beyond the data limit returns an error.

Note:

Linux does not use `sbrk()` and `brk()` within its `calloc()` and `malloc()`. Instead, it uses (`mmap()`) to create memory. DATALIMIT cannot be enforced on Linux applications that call `sbrk()` and `malloc()`.

-E "*pre_exec_command* [*arguments* ...]"

Runs the specified pre-execution command on the execution host before actually running the job. For a parallel job, the pre-execution command runs on the first host selected for the parallel job. If you want the pre-execution command to run on a specific first execution host, specify one or more first execution host candidates at the job level using -m, at the queue level with PRE_EXEC in `lsb.queues`, or at the application level with PRE_EXEC in `lsb.applications`.

If the pre-execution command returns a zero (0) exit code, LSF runs the job on the selected host. Otherwise, the job and its associated pre-execution command goes back to PEND status and is rescheduled. LSF keeps trying to run pre-execution commands and pending jobs. After the pre-execution command runs successfully, LSF runs the job. You must ensure that the pre-execution command can run multiple times without causing side effects, such as reserving the same resource more than once.

The standard input and output for the pre-execution command are directed to the same files as the job. The pre-execution command runs under the same user ID, environment, home, and working directory as the job. If the pre-execution command is not in the user's usual execution path (the $PATH variable), the full path name of the command must be specified.

Note:

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

-Ep "*post_exec_command* [*arguments* ...]"

Runs the specified post-execution command on the execution host after the job finishes.

If both application-level (POST_EXEC in `lsb.applications`) and job-level post-execution commands are specified, job level post-execution overrides application-level post-execution commands. Queue-level post-execution commands (POST_EXEC in `lsb.queues`) run after application-level post-execution and job-level post-execution commands.

Note:

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

-e *error_file*

Specify a file path. Appends the standard error output of the job to the specified file.

If the parameter LSB_STDOUT_DIRECT in `lsf.conf` is set to Y or y, the standard error output of a job is written to the file you specify as the job runs. If LSB_STDOUT_DIRECT is not set, it is written to a temporary file and copied to the specified file after the job finishes. LSB_STDOUT_DIRECT is not supported on Windows.

If you use the special character `%J` in the name of the error file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the error file, then `%I` is replaced by the index of the job in the array if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard error output file to `/tmp/`.

If the specified *error_file* path is not accessible, the output will not be stored.

Note:

The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

-eo *error_file*

Specify a file path. Overwrites the standard error output of the job to the specified file.

If the parameter LSB_STDOUT_DIRECT in `lsf.conf` is set to Y or y, the standard error output of a job is written to the file you specify as the job runs, which occurs every time the job is submitted with the overwrite option, even if it is requeued manually or by the system. If LSB_STDOUT_DIRECT is not set, it is written to a temporary file and copied to the specified file after the job finishes. LSB_STDOUT_DIRECT is not supported on Windows.

If you use the special character `%J` in the name of the error file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the error file, then `%I` is replaced by the index of the job in the array if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard error output file to `/tmp/`.

If the specified *error_file* path is not accessible, the output will not be stored.

Note:

The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

-ext[sched] "*external_scheduler_options*"

Application-specific external scheduling options for the job.

To enable jobs to accept external scheduler options, set LSF_ENABLE_EXTSCHEDULER=y in `lsf.conf`.

You can abbreviate the `-extsched` option to `-ext`.

You can specify only one type of external scheduler option in a single `-extsched` string.

For example, SGI IRIX hosts and AlphaServer SC hosts running RMS can exist in the same cluster, but they accept different external scheduler options. Use external scheduler options to define job requirements for either IRIX cpusets OR RMS, but not both. Your job runs either on IRIX or RMS. If external scheduler options are not defined, the job may run on IRIX but it does not run on an RMS host.

The options set by `-extsched` can be combined with the queue-level MANDATORY_EXTSCHED or DEFAULT_EXTSCHED parameters. If `-extsched` and MANDATORY_EXTSCHED set the same option, the MANDATORY_EXTSCHED setting is used. If `-extsched` and DEFAULT_EXTSCHED set the same options, the `-extsched` setting is used.

Use DEFAULT_EXTSCHED in `lsb.queues` to set default external scheduler options for a queue.

To make certain external scheduler options mandatory for all jobs submitted to a queue, specify MANDATORY_EXTSCHED in `lsb.queues` with the external scheduler options you need or your jobs.

See *Using Platform LSF HPC* for information about specific external scheduler options.

-F *file_limit*

Sets a per-process (soft) file size limit for each of the processes that belong to the batch job (see `getrlimit`(2)). The limit is specified in KB.

If a job process attempts to write to a file that exceeds the file size limit, then that process is sent a SIGXFSZ signal. The SIGXFSZ signal normally terminates the process.

-f "*local_file operator* [*remote_file*]" ...

Copies a file between the local (submission) host and the remote (execution) host. Specify absolute or relative paths, including the file names. You should specify the remote file as a file name with no path when running in non-shared systems.

If the remote file is not specified, it defaults to the local file, which must be given. Use multiple -f options to specify multiple files.

Note:

The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

operator

An operator that specifies whether the file is copied to the remote host, or whether it is copied back from the remote host. The operator must be surrounded by white space.

The following describes the operators:

> Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists.

< Copies the remote file to the local file after the job completes. Overwrites the local file if it exists.

<< Appends the remote file to the local file after the job completes. The local file must exist.

>< Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists. Then copies the remote file to the local file after the job completes. Overwrites the local file.

<> Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists. Then copies the remote file to the local file after the job completes. Overwrites the local file.

If you use the -i *input_file* option, then you do not have to use the -f option to copy the specified input file to the execution host. LSF does this for you, and removes the input file from the execution host after the job completes.

If you use the -o *out_file*,-e *err_file*, -oo *out_file*, or the -eo *err_file* option, and you want the specified file to be copied back to the submission host when the job completes, then you must use the -f option.

If the submission and execution hosts have different directory structures, you must make sure that the directory where the remote file and local file are placed exists.

If the local and remote hosts have different file name spaces, you must always specify relative path names. If the local and remote hosts do not share the same file system, you must make sure that the directory containing the remote file exists. It is recommended that only the file name be given for the remote file when running in heterogeneous file systems. This places the file in the job's current working directory. If the file is shared between the submission and execution hosts, then no file copy is performed.

LSF uses `lsrcp` to transfer files (see `lsrcp`(1) command). `lsrcp` contacts RES on the remote host to perform the file transfer. If RES is not available, `rcp` is used (see `rcp` (1)). The user must make sure that the `rcp` binary is in the user's $PATH on the execution host.

Jobs that are submitted from LSF client hosts should specify the -f option only if `rcp` is allowed. Similarly, `rcp` must be allowed if account mapping is used.

-G *user_group*

Only useful with fairshare scheduling.

Associates the job with the specified group. Specify any group that you belong to that does not contain any subgroups. You must be a direct member of the specified user group.

-g *job_group_name*

Submits jobs in the job group specified by *job_group_name* The job group does not have to exist before submitting the job. For example:

```
bsub -g /risk_group/portfolio1/current myjob
Job <105> is submitted to default queue.
```

Submits `myjob` to the job group `/risk_group/portfolio1/current`.

If group `/risk_group/portfolio1/current` exists, job 105 is attached to the job group.

Job group names can be up to 512 characters long.

If group `/risk_group/portfolio1/current` does not exist, LSF checks its parent recursively, and if no groups in the hierarchy exist, all three job groups are created with the specified hierarchy and the job is attached to group.

You can use -g with -sla. All jobs in a job group attached to a service class are scheduled as SLA jobs. It is not possible to have some jobs in a job group not part of the service class. Multiple job groups can be created under the same SLA. You can submit additional jobs to the job group without specifying the service class name again.

For example, the following attaches the job to the service class named opera, and the group `/risk_group/portfolio1/current`:

```
bsub -sla opera -g /risk_group/portfolio1/current myjob
```

To submit another job to the same job group, you can omit the SLA name:

```
bsub -g /risk_group/portfolio1/current myjob2
```

-i *input_file* | -is *input_file*

Gets the standard input for the job from specified file. Specify an absolute or relative path. The input file can be any type of file, though it is typically a shell script text file.

Unless you use -is, you can use the special characters `%J` and `%I` in the name of the input file. `%J` is replaced by the job ID. `%I` is replaced by the index of the job in the array, if the job is a member of an array, otherwise by 0 (zero). The special characters `%J` and `%I` are not valid with the -is option.

Note:

The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

If the file exists on the execution host, LSF uses it. Otherwise, LSF attempts to copy the file from the submission host to the execution host. For the file copy to be successful, you must allow remote copy (`rcp`) access, or you must submit the job from a server host where RES is running. The file is copied from the submission host to a temporary file in the directory specified by the JOB_SPOOL_DIR parameter in `lsb.params`, or your `$HOME/.lsbatch` directory on the execution host. LSF removes this file when the job completes.

By default, the input file is spooled to `LSB_SHAREDIR/`*cluster_name*`/lsf_indir`. If the `lsf_indir` directory does not exist, LSF creates it before spooling the file. LSF removes the spooled file when the job completes. Use the -is option if you need to modify or remove the input file before the job completes. Removing or modifying the original input file does not affect the submitted job.

If JOB_SPOOL_DIR is specified, the -is option spools the input file to the specified directory and uses the spooled file as the input file for the job.

JOB_SPOOL_DIR can be any valid path up to a maximum length up to 4094 characters on UNIX and Linux or up to 255 characters for Windows.

JOB_SPOOL_DIR must be readable and writable by the job submission user, and it must be shared by the master host and the submission host. If the specified directory is not accessible or does not exist, `bsub -is` cannot write to the default directory `LSB_SHAREDIR/`*cluster_name*`/lsf_indir` and the job fails.

-J *job_name* | -J "*job_name*[*index_list*]%*job_slot_limit*"

Assigns the specified name to the job, and, for job arrays, specifies the indices of the job array and optionally the maximum number of jobs that can run at any given time.

The job name does not need to be unique.

Job names can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

To specify a job array, enclose the index list in square brackets, as shown, and enclose the entire job array specification in quotation marks, as shown. The index list is a comma-separated list whose elements have the syntax start[-end[:step]] where start, end and step are positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one.

By default, the maximum number of jobs in a job array is 1000, which means the maximum size of a job array (that is, the maximum job array index) can never exceed 1000 jobs.

To change the maximum job array value, set MAX_JOB_ARRAY_SIZE in `lsb.params` to any positive integer between 1 and 2147483646. The maximum number of jobs in a job array cannot exceed the value set by MAX_JOB_ARRAY_SIZE.

You may also use a positive integer to specify the system-wide job slot limit (the maximum number of jobs that can run at any given time) for this job array.

All jobs in the array share the same job ID and parameters. Each element of the array is distinguished by its array index.

After a job is submitted, you use the job name to identify the job. Specify "*job_ID*[*index*]" to work with elements of a particular array. Specify "*job_name*[*index*]" to work with elements of all arrays with the same name. Since job names are not unique, multiple job arrays may have the same name with a different or same set of indices.

-jsdl *file_name* | -jsdl_strict *file_name*

Submits a job using a JSDL file to specify job submission options.

LSF provides an extension to the JSDL specification so that you can submit jobs using LSF features not defined in the JSDL standard schema. The JSDL schema (`jsdl.xsd`), the POSIX extension (`jsdl-posix.xsd`), and the LSF extension (`jsdl-lsf.xsd`) are located in the `LSF_LIBDIR` directory.

- To submit a job that uses the LSF extension, use the -jsdl option.
- To submit a job that uses only standard JSDL elements and POSIX extensions, use the -jsdl_strict option. You can use the -jsdl_strict option to verify that your file contains only valid JSDL elements and POSIX extensions. Error messages indicate invalid elements, including:
  - Elements that are not part of the JSDL specification
  - Valid JSDL elements that are not supported in this version of LSF
  - Extension elements that are not part of the JSDL standard and POSIX extension schemas

Note:

For a detailed mapping of JSDL elements to LSF submission options, and for a complete list of supported and unsupported elements, see the chapter "Submitting Jobs Using JSDL" in *Administering Platform LSF*.

If you specify duplicate or conflicting job submission parameters, LSF resolves the conflict by applying the following rules:

1. The parameters specified in the command line override all other parameters.
2. A job script or user input for an interactive job overrides parameters specified in the JSDL file.

-k "*checkpoint_dir* [init=*initial_checkpoint_period*] [*checkpoint_period*] [method=*method_name*]"

Makes a job checkpointable and specifies the checkpoint directory. Specify a relative or absolute path name. The quotes (") are required is you specify a checkpoint period, initial checkpoint period, or custom checkpoint and restart method name.

The job ID and job file name are concatenated to the checkpoint dir when creating a checkpoint file.

Note:

The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

When a job is checkpointed, the checkpoint information is stored in *checkpoint_dir*/*job_ID*/*file_name*. Multiple jobs can checkpoint into the same directory. The system can create multiple files.

The checkpoint directory is used for restarting the job (see `brestart`(1)). The checkpoint directory can be any valid path.

Optionally, specifies a checkpoint period in minutes. Specify a positive integer. The running job is checkpointed automatically every checkpoint period. The checkpoint period can be changed using `bchkpnt`. Because checkpointing is a heavyweight operation, you should choose a checkpoint period greater than half an hour.

Optionally, specifies an initial checkpoint period in minutes. Specify a positive integer. The first checkpoint does not happen until the initial period has elapsed. After the first checkpoint, the job checkpoint frequency is controlled by the normal job checkpoint interval.

Optionally, specifies a custom checkpoint and restart method to use with the job. Use `method=default` to indicate to use the default LSF checkpoint and restart programs for the job, `echkpnt.default` and `erestart.default`.

The `echkpnt.method_name` and `erestart.`*method_name* programs must be in LSF_SERVERDIR or in the directory specified by LSB_ECHKPNT_METHOD_DIR (environment variable or set in `lsf.conf`).

If a custom checkpoint and restart method is already specified with LSB_ECHKPNT_METHOD (environment variable or in `lsf.conf`), the method you specify with `bsub -k` overrides this.

Process checkpointing is not available on all host types, and may require linking programs with a special libraries (see `libckpt.a`(3)). LSF invokes `echkpnt` (see `echkpnt`(8)) found in LSF_SERVERDIR to checkpoint the job. You can override the default `echkpnt` for the job by defining as environment variables or in `lsf.conf` LSB_ECHKPNT_METHOD and LSB_ECHKPNT_METHOD_DIR to point to your own `echkpnt`. This allows you to use other checkpointing facilities, including application-level checkpointing.

The checkpoint method directory should be accessible by all users who need to run the custom `echkpnt` and `erestart` programs.

Only running members of a chunk job can be checkpointed.

-L ***login_shell***

Initializes the execution environment using the specified login shell. The specified login shell must be an absolute path. This is not necessarily the shell under which the job is executed.

Login shell is not supported on Windows.

-Lp ***ls_project_name***

Assigns the job to the specified License Scheduler project.

-M ***mem_limit***

Sets a per-process (soft) memory limit for all the processes that belong to this batch job (see `getrlimit(2)`).

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

If LSB_MEMLIMIT_ENFORCE or LSB_JOB_MEMLIMIT are set to `y` in `lsf.conf`, LSF kills the job when it exceeds the memory limit. Otherwise, LSF passes the memory limit to the operating system. UNIX operating systems that support RUSAGE_RSS for `setrlimit()` can apply the memory limit to each process.

The following operating systems do not support the memory limit at the OS level:

- Windows

- Sun Solaris 2.x

-m "*host_name***[@***cluster_name***][[!] | +[***pref_level***]] | *host_group*[[!] |+[pref_level]] | *compute_unit*[[!] |+[pref_level]]..."**

Runs the job on one of the specified hosts, or host groups, or within the specified compute units.

By default, if multiple hosts are candidates, runs the job on the least-loaded host.

When used with a compound resource requirement, the first host allocated must satisfy the simple resource requirement string appearing first in the compound resource requirement.

To change the order of preference, put a plus (+) after the names of hosts or host groups that you would prefer to use, optionally followed by a preference level. For preference level, specify a positive integer, with higher numbers indicating greater preferences for those hosts. For example, -m "hostA groupB+2 hostC+1" indicates that `groupB` is the most preferred and `hostA` is the least preferred.

The keyword others can be specified with or without a preference level to refer to other hosts not otherwise listed. The keyword others must be specified with at least one host name or host group, it cannot be specified by itself. For example, -m "hostA+ others" means that `hostA` is preferred over all other hosts.

If you also use -q, the specified queue must be configured to include all the hosts in the your host list. Otherwise, the job is not submitted. To find out what hosts are configured for the queue, use `bqueues -l`.

If the host group contains the keyword all, LSF dispatches the job to any available host, even if the host is not defined for the specified queue.

To display configured host groups and compute units, use `bmgroup`.

For the MultiCluster job forwarding model, you cannot specify a remote host by name.

For parallel jobs, specify first execution host candidates when you want to ensure that a host has the required resources or runtime environment to handle processes that run on the first execution host.

To specify one or more hosts or host groups as first execution host candidates, add the (!) symbol after the host name, as shown in the following example:

bsub -n 2 -m "host1 host2! hostgroupA! host3 host4" my_parallel_job

LSF runs `my_parallel_job` according to the following steps:

1. LSF selects either host2 or a host defined in `hostgroupA` as the first execution host for the parallel job.

   Note:

   First execution host candidates specified at the job-level (command line) override candidates defined at the queue level (in `lsb.queues`).

2. If any of the first execution host candidates have enough processors to run the job, the entire job runs on the first execution host, and not on any other hosts.

   In the example, if `host2` or a member of `hostgroupA` has two or more processors, the entire job runs on the first execution host.

3. If the first execution host does not have enough processors to run the entire job, LSF selects additional hosts that are not defined as first execution host candidates.

   Follow these guidelines when you specify first execution host candidates:

   - If you specify a host group, you must first define the host group in the file `lsb.hosts`.
   - Do not specify a dynamic host group as a first execution host.
   - Do not specify all, allremote, or others, or a host partition as a first execution host.
   - Do not specify a preference (+) for a host identified by (!) as a first execution host candidate.
   - For each parallel job, specify enough regular hosts to satisfy the processor requirement for the job. Once LSF selects a first execution host for the current job, the other first execution host candidates become unavailable to the current job, but remain available to other jobs as either regular or first execution hosts.
   - You cannot specify first execution host candidates when you use the `brun` command.

In a MultiCluster environment, insert the (!) symbol after the cluster name, as shown in the following example:

bsub -n 2 -m "host2@cluster2! host3@cluster2" my_parallel_job

When specifying compute units, the job runs within the listed compute units. Used in conjunction with a mandatory first execuation host, the compute unit containing the first execution host is given preference.

In the following example one host from host group `hg` appears first, followed by other hosts within the same compute unit. Remaining hosts from other compute units appear grouped by compute, and in the same order as configured in the ComputeUnit section of `lsb.hosts`.

bsub -n 64 -m "hg! cu1 cu2 cu3 cu4" -R "cu[pref=config]" my_job

-mig *migration_threshold*

Specifies the migration threshold for checkpointable or rerunnable jobs in minutes. Enables automatic job migration and specifies the migration threshold, in minutes. A value of 0 (zero) specifies that a suspended job should be migrated immediately.

Command-level job migration threshold overrides application profile and queue-level settings.

Where a host migration threshold is also specified, and is lower than the job value, the host value is used.

-n *min_proc*[,*max_proc*]

Submits a parallel job and specifies the number of processors required to run the job (some of the processors may be on the same multiprocessor host).

You can specify a minimum and maximum number of processors to use. The job can start if at least the minimum number of processors is available. If you do not specify a maximum, the number you specify represents the exact number of processors to use.

If PARALLEL_SCHED_BY_SLOT=Y in `lsb.params`, this option specifies the number of slots required to run the job, not the number of processors.

When used with the -R option and a compound resource requirement, the number of slots in the compound resource requirement must be compatible with the minimum and maximum specified.

Jobs that request fewer slots than the minimum PROCLIMIT defined for the queue or application profile to which the job is submitted, or more slots than the maximum PROCLIMIT are rejected. If the job requests minimum and maximum job slots, the maximum slots requested cannot be less than the minimum PROCLIMIT, and the minimum slots requested cannot be more than the maximum PROCLIMIT.

For example, if the queue defines PROCLIMIT=4 8:

- `bsub -n 6` is accepted because it requests slots within the range of PROCLIMIT
- `bsub -n 9` is rejected because it requests more slots than the PROCLIMIT allows
- `bsub -n 1` is rejected because it requests fewer slots than the PROCLIMIT allows
- `bsub -n 6,10` is accepted because the minimum value 6 is within the range of the PROCLIMIT setting
- `bsub -n 1,6` is accepted because the maximum value 6 is within the range of the PROCLIMIT setting
- `bsub -n 10,16` is rejected because its range is outside the range of PROCLIMIT
- `bsub -n 1,3` is rejected because its range is outside the range of PROCLIMIT

See the PROCLIMIT parameter in `lsb.queues`(5) and `lsb.applications`(5) for more information.

In a MultiCluster environment, if a queue exports jobs to remote clusters (see the SNDJOBS_TO parameter in `lsb.queues`), then the process limit is not imposed on jobs submitted to this queue.

Once at the required number of processors is available, the job is dispatched to the first host selected. The list of selected host names for the job are specified in the environment variables LSB_HOSTS and LSB_MCPU_HOSTS. The job itself is expected to start parallel components on these hosts and establish communication among them, optionally using RES.

Specify first execution host candidates using the -m option when you want to ensure that a host has the required resources or runtime environment to handle processes that run on the first execution host.

If you specify one or more first execution host candidates, LSF looks for a first execution host that satisfies the resource requirements. If the first execution host does not have enough processors or job slots to run the entire job, LSF looks for additional hosts.

-o *output_file*

Specify a file path. Appends the standard output of the job to the specified file. Sends the output by mail if the file does not exist, or the system has trouble writing to it.

If only a file name is specified, LSF writes the output file to the current working directory. If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard output file to `/tmp/`.

If the specified *output_file* path is not accessible, the output will not be stored.

If you use the special character `%J` in the name of the output file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the output file, then `%I` is replaced by the index of the job in the array, if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

Note:

The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

If the parameter LSB_STDOUT_DIRECT in `lsf.conf` is set to Y or y, the standard output of a job is written to the file you specify as the job runs. If LSB_STDOUT_DIRECT is not set, it is written to a temporary file and copied to the specified file after the job finishes. LSB_STDOUT_DIRECT is not supported on Windows.

If you use -o without -e or -eo, the standard error of the job is stored in the output file.

If you use -o without -N, the job report is stored in the output file as the file header.

If you use both -o and -N, the output is stored in the output file and the job report is sent by mail. The job report itself does not contain the output, but the report advises you where to find your output.

-oo *output_file*

Specify a file path. Overwrites the standard output of the job to the specified file if it exists, or sends the output to a new file if it does not exist. Sends the output by mail if the system has trouble writing to the file.

If only a file name is specified, LSF writes the output file to the current working directory. If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard output file to `/tmp/`.

If the specified *output_file* path is not accessible, the output will not be stored.

If you use the special character `%J` in the name of the output file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the output file, then `%I` is replaced by the index of the job in the array, if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

Note:

The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

If the parameter LSB_STDOUT_DIRECT in `lsf.conf` is set to Y or y, the standard output of a job overwrites the output file you specify as the job runs, which occurs every time the job is submitted with the overwrite option, even if it is requeued manually or by the system. If LSB_STDOUT_DIRECT is not set, the output is written to a temporary file that overwrites the specified file after the job finishes. LSB_STDOUT_DIRECT is not supported on Windows.

If you use -oo without -e or -eo, the standard error of the job is stored in the output file.

If you use -oo without -N, the job report is stored in the output file as the file header.

If you use both -oo and -N, the output is stored in the output file and the job report is sent by mail. The job report itself does not contain the output, but the report advises you where to find your output.

-P *project_name*

Assigns the job to the specified project.

On IRIX 6, you must be a member of the project as listed in `/etc/project`(4). If you are a member of the project, then `/etc/projid`(4) maps the project name to a numeric project ID. Before the submitted job executes, a new array session (`newarraysess`(2)) is created and the project ID is assigned to it using `setprid`(2).

-p *process_limit*

Sets the limit of the number of processes to *process_limit* for the whole job. The default is no limit. Exceeding the limit causes the job to terminate.

-Q "[*exit_code* …] [EXCLUDE(*exit_code* …)]"

Specify automatic job requeue exit values. Use spaces to separate multiple exit codes. The reserved keyword all specifies all exit codes. Exit codes are typically between 0 and 255. Use a tilde (~) to exclude specified number or numbers from the list.

exit_code has the following form:

```
"[all] [~number ...] | [number ...]"
```

Job level exit values override application-level and queue-level values.

Jobs running with the specified exit code share the same application and queue with other jobs.

Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue. Exclusive job requeue does not work for parallel jobs.

If `mbatchd` is restarted, it does not remember the previous hosts from which the job exited with an exclusive requeue exit code. In this situation, it is possible for a job to be dispatched to hosts on which the job has previously exited with an exclusive exit code.

-q "*queue_name* ..."

Submits the job to one of the specified queues. Quotes are optional for a single queue. The specified queues must be defined for the local cluster. For a list of available queues in your local cluster, use `bqueues`.

When a list of queue names is specified, LSF selects the most appropriate queue in the list for your job based on the job's resource limits, and other restrictions, such as the requested hosts, your accessibility to a queue, queue status (closed or open), etc. The order in which the queues are considered is the same order in which these queues are listed. The queue listed first is considered first.

-R "*res_req*" [-R "*res_req*" …]

Runs the job on a host that meets the specified resource requirements. A resource requirement string describes the resources a job needs. LSF uses resource requirements to select hosts for job execution. Resource requirement strings can be simple (applying to the entire job) or compound (applying to the specified number of slots).

Simple resource requirement strings are divided into the following sections. Each section has a different syntax.

- A selection section (select). The selection section specifies the criteria for selecting execution hosts from the system.
- An ordering section (order). The ordering section indicates how the hosts that meet the selection criteria should be sorted.
- A resource usage section (rusage). The resource usage section specifies the expected resource consumption of the task.
- A job spanning section (span). The job spanning section indicates if a parallel batch job should span across multiple hosts.

- A same resource section (same). The same section indicates that all processes of a parallel job must run on the same type of host.
- A compute unit resource section (cu). The compute unit section specifies topological requirements for spreading a job over the cluster.

The resource requirement string sections have the following syntax:

```
select[selection_string] order[order_string] rusage[usage_string [, usage_string]
[|| usage_string]...] span[span_string] same[same_string] cu[cu_string]
```

The square brackets must be typed as shown for each section. A blank space must separate each resource requirement section.

You can omit the select keyword and the square brackets, but the selection string must be the first string in the resource requirement string. If you do not give a section name, the first resource requirement string is treated as a selection string (select [*selection_string*]).

Compound resource requirement strings are made up of one or more simple resource requirement strings as follows:

```
num1*{simple_string1} + num2*{simple_string2} + ...
```

where *numx* is the number of slots affected and *simple_stringx* is a simple resource requirement string.

For jobs without the number of total slots specified using `bsub -n`, the final *numx* can be omitted. The final resource requirement is then applied to the zero or more slots not yet accounted for as follows:

- (final res_req number of slots) = (total number of job slots)-(*num1*+*num2*+...)

For jobs with the total number of slots specified using `bsub -n` *num_slots*, the total number of slots must match the number of slots in the resource requirement as follows:

- *num_slots*=(*num1*+*num2*+*num3*+...)

For jobs with the minimum amd maximum number of slots specified using `bsub -n` *min*, *max* , the number of slots in the compound resource requirement must be compatible with the minimum and maximum specified.

Compound resource requirements do not support use of the || operator within the component rusage simple resource requirements, multiple -R options, or use of the cu section.

Each simple resource requirement string must be contained in curly brackets. Each section has a different syntax.

The size of the resource requirement string cannot exceed 512 characters. If you need to include a hyphen (-) or other non-alphabet characters within the string, enclose the text in single quotation marks, for example, `bsub -R "select[hname!='host06-x12']"`.

If LSF_STRICT_RESREQ=Y in `lsf.conf`, the selection string must conform to the stricter resource requirement string syntax described in *Administering Platform LSF*. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu).

When LSF_STRICT_RESREQ=Y in `lsf.conf`, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

Any resource for run queue length, such as r15s, r1m or r15m, specified in the resource requirements refers to the normalized run queue length.

By default, memory (mem) and swap (swp) limits in select[] and rusage[] sections are specified in MB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the these limits (MB, GB, TB, PB, or EB).

For example, to submit a job that runs on Solaris 7 or Solaris 8:

```
bsub -R "sol7 || sol8" myjob
```

The following command runs the job called `myjob` on an HP-UX host that is lightly loaded (CPU utilization) and has at least 15 MB of swap memory available.

```
bsub -R "swp > 15 && hpux order[ut]" myjob
```

`bsub` also accepts multiple -R options for the order, same, rusage, and select sections. You can specify multiple strings instead of using the `&&` operator:

```
bsub -R "select[swp > 15]" -R "select[hpux] order[r15m]" -R rusage[mem=100]" -R "order [ut]" -R "same[type]" -R "rusage[tmp=50:duration=60]" -R "same[model]" myjob
```

LSF merges the multiple -R options into one string and selects a host that meets all of the resource requirements. The number of -R option sections is unlimited, up to a maximum of 512 characters for the entire string.

Remember:

Use multiple -R options only with the order, same, rusage, and select sections of simple resource requirement strings and with the `bsub` and `bmod` commands.

When application-level, and queue-level cu sections are also defined, the job-level cu section takes precedence and overwrites both the application-level and queue-level requirement definitions.

`EXCLUSIVE=CU[enclosure]` in `lsb.queues`, with a compute unit type `enclosure` in `lsf.params`, and `ComputeUnit` section in `lsb.hosts`. Use the following command to submit a job that runs on 64 slots over 4 enclosures or less, and uses the enclosures exclusively:

```
bsub -n 64 -R "cu[excl:type=enclosure:maxcus=4]" myjob
```

A resource called bigmem is defined in `lsf.shared` as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command to submit a job that runs on `hostE`:

```
bsub -R "bigmem" myjob
```

or

```
bsub -R "defined(bigmem)" myjob
```

A static shared resource is configured for licenses for the Verilog application as a resource called `verilog_lic`. To submit a job that runs on a host when there is a license available:

```
bsub -R "select[defined(verilog_lic)] rusage[verilog_lic=1]" myjob
```

The following job requests 20 MB memory for the duration of the job, and 1 license for 2 minutes:

```
bsub -R "rusage[mem=20, license=1:duration=2]" myjob
```

The following job requests 20 MB of memory and 50 MB of swap space for 1 hour, and 1 license for 2 minutes:

```
bsub -R "rusage[mem=20:swp=50:duration=1h, license=1:duration=2]" myjob
```

The following job requests 20 MB of memory for the duration of the job, 50 MB of swap space for 1 hour, and 1 license for 2 minutes.

```
bsub -R "rusage[mem=20,swp=50:duration=1h, license=1:duration=2]" myjob
```

The following job requests 50 MB of swap space, linearly decreasing the amount reserved over a duration of 2 hours, and requests 1 license for 2 minutes:

```
bsub -R "rusage[swp=50:duration=2h:decay=1, license=1:duration=2]" myjob
```

The following job requests two resources with same duration but different decay:

```
bsub -R "rusage[mem=20:duration=30:decay=1, lic=1:duration=30]" myjob
```

You are running an application version 1.5 as a resource called `app_lic_v15` and the same application version 2.0.1 as a resource called `app_lic_v201`. The license key for version 2.0.1 is backward compatible with version 1.5, but the license key for version 1.5 does not work with 2.0.1.

Job-level resource requirement specifications that use the `||` operator take precedence over any queue-level resource requirement specifications.

- If you can only run your job using one version of the application, submit the job without specifying an alternative resource. To submit a job that only uses `app_lic_v201`:

  ```
  bsub -R "rusage[app_lic_v201=1]" myjob
  ```
- If you can run your job using either version of the application, try to reserve version 2.0.1 of the application. If it is not available, you can use version 1.5. To submit a job that tries `app_lic_v201` before trying `app_lic_v15`:

  ```
  bsub -R "rusage[app_lic_v201=1||app_lic_v15=1]" myjob
  ```
- If different versions of an application require different system resources, you can specify other resources in your `rusage` strings. To submit a job that uses 20 MB of memory for `app_lic_v201` or 20 MB of memory and 50 MB of swap space for `app_lic_v15`:

  ```
  bsub -R "rusage[mem=20:app_lic_v15=1||mem=20:swp=50:app_lic_v201=1]" myjob
  ```

-S *stack_limit*

Sets a per-process (soft) stack segment size limit for each of the processes that belong to the batch job (see `getrlimit`(2)).

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

-s *signal*

Send the specified signal when a queue-level run window closes.

By default, when the window closes, LSF suspends jobs running in the queue (job state becomes SSUSP) and stops dispatching jobs from the queue.

Use `-s` to specify a signal number; when the run window closes, the job is signalled by this signal instead of being suspended.

-rnc *resize_notification_cmd*

Specify the full path of an executable to be invoked on the first execution host when the job allocation has been modified (both shrink and grow). -rnc overrides the notification command specified in the application profile (if specified). The maximum length of the notification command is 4 KB.

-sla *service_class_name*

Specifies the service class where the job is to run.

If the SLA does not exist or the user is not a member of the service class, the job is rejected.

If EGO-enabled SLA scheduling is configured with ENABLE_DEFAULT_EGO_SLA in `lsb.params`, jobs submitted without -sla are attached to the configured default SLA.

You can use -g with -sla. All jobs in a job group attached to a service class are scheduled as SLA jobs. It is not possible to have some jobs in a job group not part of the service class. Multiple job groups can be created under the same SLA. You can submit additional jobs to the job group without specifying the service class name again.

Tip:

You should submit your jobs with a runtime limit (-W option) or you should specify a run time limit in a queue or application profile (RUNLIMIT in the queue definition in `lsb.queues` or RUNLIMIT in the application profile definition in `lsb.applications`). If you do not specify a run time limit, LSF automatically adjusts the optimum number of running jobs according to the observed run time of finished jobs.

Use `bsla` to display the properties of service classes configured in `LSB_CONFDIR/`*`cluster_name`*`/configdir/lsb.serviceclasses` (see `lsb.serviceclasses`(5)) and dynamic information about the state of each service class.

-sp *priority*

Specifies user-assigned job priority that allows users to order their jobs in a queue. Valid values for priority are any integers between 1 and MAX_USER_PRIORITY (configured in `lsb.params`, displayed by `bparams -l`). Job priorities that are not valid are rejected. LSF and queue administrators can specify priorities beyond MAX_USER_PRIORITY.

The job owner can change the priority of their own jobs. LSF and queue administrators can change the priority of all jobs in a queue.

Job order is the first consideration to determine job eligibility for dispatch. Jobs are still subject to all scheduling policies regardless of job priority. Jobs with the same priority are ordered first come first served.

User-assigned job priority can be configured with automatic job priority escalation to automatically increase the priority of jobs that have been pending for a specified period of time (JOB_PRIORITY_OVER_TIME in `lsb.params`).

When absolute priority scheduling is configured in the submission queue (APS_PRIORITY in `lsb.queues`), the user-assigned job priority is used for the JPRIORITY factor in the APS calculation.

-T *thread_limit*

Sets the limit of the number of concurrent threads to *thread_limit* for the whole job. The default is no limit.

Exceeding the limit causes the job to terminate. The system sends the following signals in sequence to all processes belongs to the job: SIGINT, SIGTERM, and SIGKILL.

-t [[*year*:][*month*:]*day*:]*hour*:*minute*

Specifies the job termination deadline.

If a UNIX or Linux job is still running at the termination time, the job is sent a SIGUSR2 signal, and is killed if it does not terminate within ten minutes.

If a Windows job is still running at the termination time, it is killed immediately. (For a detailed description of how these jobs are killed, see `bkill`.)

In the queue definition, a TERMINATE action can be configured to override the `bkill` default action (see the JOB_CONTROLS parameter in `lsb.queues`(5)).

In an application profile definition, a TERMINATE_CONTROL action can be configured to override the `bkill` default action (see the TERMINATE_CONTROL parameter in `lsb.applications`(5)).

The format for the termination time is [[*year*:][*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: year after 1970, month 1-12, day 1-31, hour 0-23, minute 0-59.

At least two fields must be specified. These fields are assumed to be *hour*:*minute*. If three fields are given, they are assumed to be *day*:*hour*:*minute*, four fields are assumed to be *month*:*day*:*hour*:*minute* and five fields are assumed to be *year*:*month*:*day*:*hour*:*minute*.

If the year field is specified and the specified time is in the past, the job submission request is rejected.

-U *reservation_ID*

If an advance reservation has been created with the `brsvadd` command, the -U option makes use of the reservation.

For example, if the following command was used to create the reservation user1#0,

```
brsvadd -n 1024 -m hostA -u user1 -b 13:0 -e 18:0
Reservation "user1#0" is created
```

The following command uses the reservation:

```
bsub -U user1#0 myjob
```

The job can only use hosts reserved by the reservation `user1#0`. LSF only selects hosts in the reservation. You can use the -m option to specify particular hosts within the list of hosts reserved by the reservation, but you cannot specify other hosts not included in the original reservation.

If you do not specify hosts (`bsub -m`) or resource requirements (`bsub -R`), the default resource requirement is to select hosts that are of any host type (LSF assumes "type==any" instead of "type==local" as the default select string).

If you later delete the advance reservation while it is still active, any pending jobs still keep the "type==any" attribute.

A job can only use one reservation. There is no restriction on the number of jobs that can be submitted to a reservation; however, the number of slots available on the hosts in the reservation may run out. For example, reservation `user2#0` reserves 128 slots on `hostA`. When all 128 slots on `hostA` are used by jobs referencing `user2#0`, `hostA` is no longer available to other jobs using reservation `user2#0`. Any single user or user group can have a maximum of 100 reservation IDs

Jobs referencing the reservation are killed when the reservation expires. LSF administrators can prevent running jobs from being killed when the reservation expires by changing the termination time of the job using the reservation (`bmod -t`) before the reservation window closes.

To use an advance reservation on a remote host, submit the job and specify the remote advance reservation ID. For example:

```
bsub -U user1#01@cluster1
```

In this example, we assume the default queue is configured to forward jobs to the remote cluster.

-u *mail_user*

Sends mail to the specified email destination. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

-v *swap_limit*

Set the total process virtual memory limit to *swap_limit* for the whole job. The default is no limit. Exceeding the limit causes the job to terminate.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

-W [*hour*:]*minute*[/*host_name* | /*host_model*]

Sets the runtime limit of the batch job. If a UNIX or Linux job runs longer than the specified run limit, the job is sent a SIGUSR2 signal, and is killed if it does not terminate within ten minutes. If a Windows job runs longer than the specified run limit, it is killed immediately. (For a detailed description of how these jobs are killed, see `bkill`.)

In the queue definition, a TERMINATE action can be configured to override the `bkill` default action (see the JOB_CONTROLS parameter in `lsb.queues`(5)).

In an application profile definition, a TERMINATE_CONTROL action can be configured to override the `bkill` default action (see the TERMINATE_CONTROL parameter in `lsb.applications`(5)).

If you want to provide LSF with an estimated run time without killing jobs that exceed this value, submit the job with -We, or define the RUNTIME parameter in `lsb.applications` and submit the job to that application profile. LSF uses the estimated runtime value for scheduling purposes only.

The run limit is in the form of [*hour*:]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The run limit you specify is the normalized run time. This is done so that the job does approximately the same amount of processing, even if it is sent to host with a faster or slower CPU. Whenever a normalized run time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

If ABS_RUNLIMIT=Y is defined in `lsb.params`, the runtime limit and the runtime estimate are not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted with a runtime limit or runtime estimate.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert '`/`' between the run limit and the host name or model name.

If no host or host model is given, LSF uses the default runtime normalization host defined at the queue level (DEFAULT_HOST_SPEC in `lsb.queues`) if it has been configured; otherwise, LSF uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in `lsb.params`) if it has been configured; otherwise, LSF uses the submission host.

For MultiCluster jobs, if no other CPU time normalization host is defined and information about the submission host is not available, LSF uses the host with the largest CPU factor (the fastest host in the cluster).

If the job also has termination time specified through the `bsub -t` option, LSF determines whether the job can actually run for the specified length of time allowed by the run limit before the termination time. If not, then the job is aborted.

If the IGNORE_DEADLINE parameter is set in `lsb.queues`(5), this behavior is overridden and the run limit is ignored.

Jobs submitted to a chunk job queue are not chunked if the run limit is greater than 30 minutes.

-We [*hour*:]*minute*[/*host_name* | /*host_model*]

Specifies an estimated run time for the job. LSF uses the estimated value for job scheduling purposes only, and does not kill jobs that exceed this value unless the jobs also exceed a defined runtime limit. The format of runtime estimate is same as run limit set by the -W option.

Use JOB_RUNLIMIT_RATIO in `lsb.params` to limit the runtime estimate users can set. If JOB_RUNLIMIT_RATIO is set to 0 no restriction is applied to the runtime estimate.

The job-level runtime estimate setting overrides the RUNTIME setting in an application profile in `lsb.applications`.

-w '*dependency_expression*'

LSF does not place your job unless the dependency expression evaluates to TRUE. If you specify a dependency on a job that LSF cannot find (such as a job that has not yet been submitted), your job submission fails.

The dependency expression is a logical expression composed of one or more dependency conditions. To make dependency expression of multiple conditions, use the following logical operators:

&& (AND)

|| (OR)

! (NOT)

Use parentheses to indicate the order of operations, if necessary.

Enclose the dependency expression in single quotes (') to prevent the shell from interpreting special characters (space, any logic operator, or parentheses). If you use single quotes for the dependency expression, use double quotes (") for quoted items within it, such as job names.

In a Windows environment with multiple job dependencies, use only double quotes.

In dependency conditions, job names specify only your own jobs, unless you are the LSF administrator. By default, if you use the job name to specify a dependency condition, and more than one of your jobs has the same name, all of your jobs that have that name must satisfy the test. If JOB_DEP_LAST_SUB in `lsb.params` is set to 1, the test is done on the job submitted most recently.

Use double quotes (") around job names that begin with a number. In the job name, specify the wildcard character asterisk (*) at the end of a string, to indicate all jobs whose name begins with the string. For example, if you use `jobA*` as the job name, it specifies jobs named `jobA`, `jobA1`, `jobA_test`, `jobA.log`, etc.

Use the * with dependency conditions to define one-to-one dependency among job array elements such that each element of one array depends on the corresponding element of another array. The job array size must be identical.

For example:

```
bsub -w "done(myarrayA[*])" -J "myArrayB[1-10]" myJob2
```

indicates that before element 1 of `myArrayB` can start, element 1 of `myArrayA` must be completed, and so on.

You can also use the * to establish one-to-one array element dependencies with `bmod` after an array has been submitted.

If you want to specify array dependency by array name, set JOB_DEP_LAST_SUB in `lsb.params`. If you do not have this parameter set, the job is rejected if one of your previous arrays has the same name but a different index.

In dependency conditions, the variable *op* represents one of the following relational operators:

>

>=

<

<=

==

!=

Use the following conditions to form the dependency expression.

done(*job_ID* |"*job_name*" ...)

The job state is DONE.

LSF refers to the oldest job of *job_name* in memory.

ended(*job_ID* | "*job_name*")

The job state is EXIT or DONE.

exit(*job_ID* | "*job_name*" [,[*operator*] *exit_code*])

The job state is EXIT, and the job's exit code satisfies the comparison test.

If you specify an exit code with no operator, the test is for equality (== is assumed).

If you specify only the job, any exit code satisfies the test.

external(*job_ID* | "*job_name*", "*status_text*")

The job has the specified job status.

If you specify the first word of the message description (no spaces), the text of the job's status begins with the specified word. Only the first word is evaluated.

***job_ID* | "*job_name*"**

If you specify a job without a dependency condition, the test is for the DONE state (LSF assumes the "done" dependency condition by default).

numdone(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the DONE state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numended(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the DONE or EXIT states satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numexit(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the EXIT state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numhold(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the PSUSP state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numpend(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the PEND state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numrun(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the RUN state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numstart(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the RUN, USUSP, or SSUSP states satisfies the test. Use * (with no operator) to specify all the jobs in the array.

post_done(*job_ID* | "*job_name*")

The job state is POST_DONE (post-execution processing of the specified job has completed without errors).

post_err(*job_ID* | "*job_name*")

The job state is POST_ERR (post-execution processing of the specified job has completed with errors).

started(*job_ID* | "*job_name*")

The job state is:

- USUSP, SSUSP, DONE, or EXIT
- RUN and the job has a pre-execution command (`bsub -E`) that is done.

-wa '*signal*'

Specifies the job action to be taken before a job control action occurs.

A job warning action must be specified with a job action warning time in order for job warning to take effect.

If -wa is specified, LSF sends the warning action to the job before the actual control action is taken. This allows the job time to save its result before being terminated by the job control action.

The warning action specified by -wa option overrides JOB_WARNING_ACTION in the queue. JOB_WARNING_ACTION is used as the default when no command line option is specified.

For example the following specifies that 2 minutes before the job reaches its runtime limit, an URG signal is sent to the job:

```
bsub -W 60 -wt '2' -wa 'URG' myjob
```

-wt '[*hour*:]*minute*'

Specifies the amount of time before a job control action occurs that a job warning action is to be taken. Job action warning time is not normalized.

A job action warning time must be specified with a job warning action in order for job warning to take effect.

The warning time specified by the `bsub -wt` option overrides JOB_ACTION_WARNING_TIME in the queue. JOB_ACTION_WARNING_TIME is used as the default when no command line option is specified.

For example the following specifies that 2 minutes before the job reaches its runtime limit, an URG signal is sent to the job:

```
bsub -W 60 -wt '2' -wa 'URG' myjob
```

-Zs

Spools a job command file to the directory specified by the JOB_SPOOL_DIR parameter in `lsb.params`, and uses the spooled file as the command file for the job.

By default, the command file is spooled to `LSB_SHAREDIR/`*cluster_name*`/lsf_cmddir`. If the `lsf_cmddir` directory does not exist, LSF creates it before spooling the file. LSF removes the spooled file when the job completes.

If JOB_SPOOL_DIR is specified, the -Zs option spools the command file to the specified directory and uses the spooled file as the input file for the job.

JOB_SPOOL_DIR can be any valid path up to a maximum length up to 4094 characters on UNIX and Linux or up to 255 characters for Windows.

JOB_SPOOL_DIR must be readable and writable by the job submission user, and it must be shared by the master host and the submission host. If the specified directory is not accessible or does not exist, `bsub -Zs` cannot write to the default directory `LSB_SHAREDIR/`*cluster_name*`/lsf_cmddir` and the job fails.

The -Zs option is not supported for embedded job commands because LSF is unable to determine the first command to be spooled in an embedded job command.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

***command* [*argument*]**

The job can be specified by a command line argument command, or through the standard input if the command is not present on the command line. The *command* can be anything that is provided to a UNIX Bourne shell (see `sh`(1)). command is assumed to begin with the first word that is not part of a `bsub` option. All arguments that follow *command* are provided as the arguments to the *command*.

The job command can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. If no job name is specified with `-J`, `bjobs`, `bhist` and `bacct` displays the command as the job name.

If the batch job is not given on the command line, `bsub` reads the job commands from standard input. If the standard input is a controlling terminal, the user is prompted with `bsub>` for the commands of the job. The input is terminated by entering CTRL-D on a new line. You can submit multiple commands through standard input.

The commands are executed in the order in which they are given. `bsub` options can also be specified in the standard input if the line begins with `#BSUB`; for example, `#BSUB -x`. If an option is given on both the `bsub` command line, and in the standard input, the command line option overrides the option in the standard input. The user can specify the shell to run the commands by specifying the shell path name in the first line of the standard input, such as `#!/bin/csh`. If the shell is not given in the first line, the Bourne shell is used. The standard input facility can be used to spool a user's job script; such as `bsub < script`.

Output

If the job is successfully submitted, displays the job ID and the queue to which the job has been submitted.

Examples

bsub sleep 100

Submit the UNIX command `sleep` together with its argument `100` as a batch job.

bsub -q short -o my_output_file "pwd; ls"

Submit the UNIX command `pwd` and `ls` as a batch job to the queue named `short` and store the job output in `my_output` file.

bsub -m "host1 host3 host8 host9" my_program

Submit `my_program` to run on one of the candidate hosts: host1, host3, host8 and host9.

bsub -q "queue1 queue2 queue3" -c 5 my_program

Submit `my_program` to one of the candidate queues: `queue1`, `queue2`, and `queue3` that are selected according to the CPU time limit specified by `-c 5`.

bsub -I ls

Submit a batch interactive job that displays the output of `ls` at the user's terminal.

bsub -Ip vi myfile

Submit a batch interactive job to edit `myfile`.

bsub -Is csh

Submit a batch interactive job that starts `csh` as an interactive shell.

bsub -b 20:00 -J my_job_name my_program

Submit `my_program` to run after 8 p.m. and assign it the job name my_job_name.

bsub my_script

Submit `my_script` as a batch job. Since my_script is specified as a command line argument, the `my_script` file is not spooled. Later changes to the `my_script` file before the job completes may affect this job.

bsub < default_shell_script

where default_shell_script contains:

```
sim1.exe
sim2.exe
```

The file `default_shell_script` is spooled, and the commands are run under the Bourne shell since a shell specification is not given in the first line of the script.

bsub < csh_script

where `csh_script` contains:

```
#!/bin/csh
sim1.exe
sim2.exe
```

`csh_script` is spooled and the commands are run under `/bin/csh`.

bsub -q night < my_script

where `my_script` contains:

```
#!/bin/sh
#BSUB -q test
#BSUB -m "host1 host2" # my default candidate hosts
#BSUB -f "input > tmp" -f "output << tmp"
#BSUB -D 200 -c 10/host1
#BSUB -t 13:00
#BSUB -k "dir 5"
sim1.exe
sim2.exe
```

The job is submitted to the `night` queue instead of `test`, because the command line overrides the script.

bsub -b 20:00 -J my_job_name

```
bsub> sleep 1800
bsub> my_program
bsub> CTRL-D
```

The job commands are entered interactively.

bsub -T 4 myjob

Submits `myjob` with a maximum number of concurrent threads of 4.

bsub -W 15 -sla Kyuquot sleep 100

Submit the UNIX command `sleep` together with its argument `100` as a batch job to the service class named `Kyuquot`.

Limitations

When using account mapping, the command `bpeek` does not work. File transfer via the -f option to `bsub` requires `rcp` to be working between the submission and execution hosts. Use the -N option to request mail, and/or the -o and -e options to specify an output file and error file, respectively.

See also

bjobs, bkill, bqueues, bhosts, bmgroup, bmod, bchkpnt, brestart, bgadd, bgdel, bjgroup, `sh`, `getrlimit`, `sbrk`, `libckpt.a`, `lsb.users`, `lsbqueues`, `lsb.params`, `lsb.hosts`, `lsb.serviceclasses`, `mbatchd`

bswitch

switches unfinished jobs from one queue to another

Synopsis

bswitch [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group* | **-m** *compute_unit*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] *destination_queue* [**0**]

bswitch *destination_queue* [*job_ID* | **"***job_ID***[***index_list***]"**] ...

bswitch [**-h** | **-V**]

Description

Switches one or more of your unfinished jobs to the specified queue. LSF administrators and root can switch jobs submitted by other users.

By default, switches one job, the most recently submitted job, or the most recently submitted job that also satisfies other specified options (-m, -q, -u, or -J). Specify 0 (zero) to switch multiple jobs.

The switch operation can be done only if a specified job is acceptable to the new queue as if it were submitted to it, and, in case the job has been dispatched to a host, if the host can be used by the new queue. If the switch operation is unsuccessful, the job stays where it is.

If a switched job has not been dispatched, then its behavior is as if it were submitted to the new queue in the first place.

If a switched job has been dispatched, then it is controlled by the `loadSched` and `loadStop` vectors and other configuration parameters of the new queue, but its nice value and resource limits remain the same.

Also, if a switched job has been dispatched, it is controlled by the PRIORITY and RUN_WINDOW configuration parameters of the new queue.

Members of a chunk job can be switched to another queue. Running chunk job members are removed from the chunk and switched; all other WAIT jobs are requeued to PEND. For chunk jobs in WAIT state, only the WAIT job is removed from the chunk and switched, and requeued to PEND.

The `bswitch` command is useful to change a job's attributes inherited from the queue.

`bswitch` can switch resizable jobs between queues regardless of job state. Once the job is switched, the parameters in new queue apply, including threshold configuration, run limit, CPU limit, queue-level resource requirements, etc.

Options

0

(Zero). Switches multiple jobs. Switches all the jobs that satisfy other specified options (-m, -q, -u and -J).

-J *job_name*

Only switches jobs that have the specified job name.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but cannot appear within array indices. For example `job*` returns jobA and jobarray[1], `*AAA*[1]`

returns the first element in all job arrays with names containing AAA, however `job1[*]` will not return anything since the wildcard is within the array index.

-m *host_name* | -m *host_group* | -m *compute_unit*

Only switches jobs dispatched to the specified host, host group, or compute unit.

-q *queue_name*

Only switches jobs in the specified queue.

-u *user_name* | -u *user_group* | -u all

Only switches jobs submitted by the specified user, or all users if you specify the keyword all. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME\user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME\\user_name*) in a UNIX command line.

If you specify a user group, switches jobs submitted by all users in the group.

destination_queue

Required. Specify the queue to which the job is to be moved.

job_ID ... |"job_ID[index_list]" ...

Switches only the specified jobs.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Limitations

You cannot switch a MultiCluster job.

See also

bqueues, bhosts, bugroup, bsub, bjobs

btop

moves a pending job relative to the first job in the queue

Synopsis

btop *job_ID* | **"***job_ID***[***index_list***]"** [*position*]

btop [**-h** | **-V**]

Description

Changes the queue position of a pending job or a pending job array element, to affect the order in which jobs are considered for dispatch.

By default, LSF dispatches jobs in a queue in the order of their arrival (that is, first come, first served), subject to availability of suitable server hosts.

The `btop` command allows users and the LSF administrator to manually change the order in which jobs are considered for dispatch. Users can only operate on their own jobs, whereas the LSF administrator can operate on any user's jobs. Users can only change the relative position of their own jobs.

If invoked by the LSF administrator, `btop` moves the selected job before the first job with the same priority submitted to the queue. The positions of all users' jobs in the queue can be changed by the LSF administrator.

If invoked by a regular user, `btop` moves the selected job before the first job with the same priority submitted by the user to the queue. Pending jobs are displayed by `bjobs` in the order in which they are considered for dispatch.

A user may use `btop` to change the dispatch order of his/her jobs scheduled using a fairshare policy. However, if a job scheduled using a fairshare policy is moved by the LSF administrator using `btop`, the job is not subject to further fairshare scheduling unless the same job is subsequently moved by the LSF administrator using `bbot`; in this case the job is scheduled again using the same fairshare policy (see the FAIRSHARE keyword in `lsb.queues`(5) and HostPartition keyword in `lsb.hosts` (5)).

To prevent users from changing the queue position of a pending job with `btop`, configure JOB_POSITION_CONTROL_BY_ADMIN=Y in `lsb.params`.

You cannot run `btop` on jobs pending in an absolute priority scheduling (APS) queue.

Options

***job_ID* | "*job_ID[index_list]*"**

Required. Job ID of the job or of the job array on which to operate.

For a job array, the index list, the square brackets, and the quotation marks are required. An index list is used to operate on a job array. The index list is a comma separated list whose elements have the syntax start_index[-end_index[:step]] where start_index, end_index and step are positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one. The maximum job array index is 1000. All jobs in the array share the same job_ID and parameters. Each element of the array is distinguished by its array index.

position

Optional. The *position* argument can be specified to indicate where in the queue the job is to be placed. position is a positive number that indicates the target position of

the job from the beginning of the queue. The positions are relative to only the applicable jobs in the queue, depending on whether the invoker is a regular user or the LSF administrator. The default value of 1 means the position is before all the other jobs in the queue that have the same priority.

-h

Prints command usage to `stderr` and exits

-V

Prints LSF release version to `stderr` and exits

See also

bbot(1), bjobs(1), bswitch(1)

bugroup

displays information about user groups

Synopsis

bugroup [**-l**] [**-r**] [**-w**] [*user_group ...*]

bugroup [**-h** | **-V**]

Description

Displays user groups, user names, and group administrators for each group.

The default is to display information about all user groups.

Options

-l

Displays information in a long multi-line format. Also displays share distribution if shares are configured.

-r

Expands the user groups recursively. The expanded list contains only user names; it does not contain the names of subgroups. Duplicate user names are listed only once.

-w

Wide format. Displays user and user group names without truncating fields.

user_group ...

Only displays information about the specified user groups. Do not use quotes when specifying multiple user groups.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

In the list of users, a name followed by a slash (/) indicates a subgroup.

Files

User groups and user shares are defined in the configuration file `lsb.users`(5).

See also

lsb.users, bmgroup, busers

busers

displays information about users and user groups

Synopsis

busers [**-w**] [*user_name* ... | *user_group* ... | **all**]

busers [**-h** | **-V**]

Description

Displays information about users and user groups.

By default, displays information about the user who runs the command.

When a resizable job has a resize allocation request, `busers` displays pending requests. When LSF adds more resources to a running resizable job, `busers` decreases job PEND counts and displays the added resources. When LSF removes resources from a running resizable job, `busers` displays the updated resources.

Options

***user_name* ... | *user_group* ... | all**

Displays information about the specified users or user groups, or about all users if you specify `all`. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME\user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME\\user_name*) in a UNIX command line.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

-w

Prints user and user group pending job thresholds and exits.

Output

A listing of the users and user groups is displayed with the following fields:

USER/GROUP

The name of the user or user group.

JL/P

The maximum number of job slots that can be processed simultaneously for the specified users on each processor. For non-preemptive scheduling, these job slots are used by running and suspended jobs or by pending jobs that have jobs slots reserved for them. For preemptive scheduling, these job slots are used by running jobs or by pending jobs that have slots reserved for them. This job limit is configured per

processor so that multiprocessor hosts have more job slots. If the dash character (-) is displayed, there is no limit. JL/P is defined in the LSF configuration file `lsb.users`.

MAX

The maximum number of job slots that can be processed concurrently for the specified users' jobs. For non-preemptive scheduling, these job slots are used by running and suspended jobs or by pending jobs that have job slots reserved for them. For preemptive scheduling, these job slots are used by running jobs or by pending jobs that have job slots reserved for them. If the character '–' is displayed, there is no limit. MAX is defined by the MAX_JOBS parameter in the configuration file `lsb.users(5)`.

NJOBS

The current number of job slots used by specified users' jobs. A parallel job that is pending is counted as *n* job slots for it uses *n* job slots in the queue when it is dispatched.

PEND

The number of pending job slots used by jobs of the specified users.

RUN

The number of job slots used by running jobs of the specified users.

SSUSP

The number of job slots used by the system-suspended jobs of the specified users.

USUSP

The number of job slots used by user-suspended jobs of the specified users.

RSV

The number of job slots used by pending jobs of the specified users that have job slots reserved for them.

MPEND

The pending job threshold for the specified users or user groups. MPEND is defined by the MAX_PEND_JOBS parameter in the configuration file `lsb.users`.

See also

bugroup, `lsb.users`, `lsb.queues`

ch

changes the host on which subsequent commands are to be executed

Synopsis

ch [**-S**] [**-t**] [*host_name*]

ch [**-h** | **-V**]

Description

Changes the host on which subsequent commands are to be executed.

By default, if no arguments are specified, changes the current host to the home host, the host from which the `ch` command was issued.

By default, executes commands on the home host.

By default, shell mode support is not enabled.

By default, does not display execution time of tasks.

The `ch` command allows you to quickly change to a designated host with the same execution environment. A simple shell is started that delivers all subsequent commands (except built-in commands) to the designated host for execution.

When the simple shell starts, it is in the current working directory and has the same command execution environment as that of the parent shell. Every remotely dispatched command is executed with the same environment as that on the home host. The syntax of the `ch` command is similar to that of the Bourne shell. However, there are some important differences.

The ampersand (`&`) following a command line (representing a background job in the Bourne shell) is ignored by `ch`. You can submit background jobs in `ch` with the built-in `post` command and bring them into the foreground with the built-in `contact` command (see below for details).

`ch` recognizes a ~ (tilde) as a special path name. If a ~ (tilde) is followed by a space, tab, new line or / (slash) character, then the ~ character is translated into the user's home directory. Otherwise, the ~ is translated as the home directory of the user name given by the string following the ~ character. Pipelines, lists of commands and redirection of standard input/output are all handled by invoking `/bin/sh`.

The following sequence of commands illustrates the behavior of the `ch` command. For example, the user is currently on `hostA`:

```
ch hostB
hostB> ch hostC
hostC> ch
hostA> ... ...
```

Options

-S

Starts remote tasks with shell mode support. Shell mode support is required for running interactive shells or applications that redefine the CTRL-C and CTRL-Z keys (for example, `jove`).

-t

Turns on the timing option. The amount of time each subsequent command takes to execute is displayed.

host_name

Executes subsequent commands on the specified host.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Usage

The `ch` command interprets the following built-in commands:

cd [*directory_name*]

Changes the current working directory to the specified directory. If a directory is not specified, changes to the user's home directory by default.

ch [*host_name*]

Changes the current working host to the specified host. If a host is not specified, changes to the home host by default.

post [*command* [*argument* ...]]

Posts the specified command for execution in the background on the current working host. `ch` assigns a unique task ID to this command and displays this ID, then continues to interact with the user. However, the output of background jobs may disturb the screen. You can post multiple commands on one host or on different hosts. When a previously posted command is completed, `ch` reports its status to the standard error. If a command is not specified, `ch` displays all currently running background commands.

contact *task_ID*

Brings a previously posted background command into the foreground. task_ID is the ID returned by the `post` command. Standard input is now passed to this foreground command. You cannot put an active foreground job into the background. A command that has been brought into the foreground with the `contact` command cannot be put back into the background.

exit

Exits `ch` if there are no posted commands running. Typing an EOF character (usually CTRL-D but may be set otherwise, see `stty(1)`) forces `ch` to exit; uncompleted posted commands are killed.

Limitations

Currently, the `ch` command does not support script, history, nor alias.

The `ch` prompt is always the current working host:current working directory followed by a > (right angle bracket) character. If the `ch` session is invoked by a shell that supports job control (such as `tcsh` or `ksh`), CTRL-Z suspends the whole `ch` session. The exit status of a command line is printed to `stderr` if the status is non-zero.

See also

lsrun(1), `rsh`(1), `stty`(1)

lsacct

displays accounting statistics on finished RES tasks in the LSF system

Synopsis

lsacct [**-l**] [**-C** *time0,time1*] [**-S** *time0,time1*] [**-f** *logfile_name*] [**-m** *host_name*] [**-u** *user_name ...* | **-u all**] [*pid ...*]

lsacct [**-h** | **-V**]

Description

Displays statistics on finished tasks run through RES. When a remote task completes, RES logs task statistics in the task log file.

By default, displays accounting statistics for only tasks owned by the user who invoked the `lsacct` command.

By default, displays accounting statistics for tasks executed on all hosts in the LSF system.

By default, displays statistics for tasks logged in the task log file currently used by RES: `LSF_RES_ACCTDIR/lsf.acct.`*host_name* `or /tmp/lsf.acct.`*host_name* (see `lsf.acct`(5)).

If -l is not specified, the default is to display the fields in SUMMARY only (see OUTPUT).

The RES on each host writes its own accounting log file. These files can be merged using the `lsacctmrg` command to generate statistics for the entire LSF cluster.

All times are reported in seconds. All sizes are reported in kilobytes.

Options

-l

Per-task statistics. Displays statistics about each task. See OUTPUT for a description of information that is displayed.

-C *time0,time1*

Displays accounting statistics for only tasks that completed or exited during the specified time interval.

The time format is the same as in `bhist`(1).

-f *logfile_name*

Searches the specified task log file for accounting statistics. Specify either an absolute or a relative path.

Useful for analyzing old task log files or files merged with the `lsacctmrg` command.

-m *host_name ...*

Displays accounting statistics for only tasks executed on the specified hosts.

If a list of hosts is specified, host names must be separated by spaces and enclosed in quotation marks (") or (’).

-S *time0,time1*

Displays accounting statistics for only tasks that began executing during the specified time interval.

The time format is the same as in `bhist`(1).

-u *user_name* ... | -u all

Displays accounting statistics for only tasks owned by the specified users, or by all users if the keyword `all` is specified.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or (’). You can specify both user names and user IDs in the list of users. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME\user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME\\user_name*) in a UNIX command line.

***pid* ...**

Displays accounting statistics for only tasks with the specified pid. This option overrides all other options except for -l, -f, -h, -V.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output: SUMMARY (default format)

Overall statistics for tasks.

The total, average, maximum and minimum resource usage statistics apply to all specified tasks.

The following fields are displayed:

Total number of tasks

Total number of tasks including tasks completed successfully and total number of exited tasks.

Time range of started tasks

Start time of the first and last task selected.

Time range of ended tasks

Completion or exit time of the first and last task selected.

Resource usage of tasks selected

See `getrusage` (2).

CPU time

Total CPU time consumed by the task.

Page faults

Number of page faults.

Swaps

Number of times the process was swapped out.

Blocks in

Number of input blocks.

Blocks out

Number of output blocks.

Messages sent

Number of System VIPC messages sent.

Messages rcvd

Number of IPC messages received.

Voluntary cont sw

Number of voluntary context switches.

Involuntary con sw

Number of involuntary context switches.

Turnaround

Elapsed time from task execution to task completion.

Output: Per Task Statistics (-l)

In addition to the fields displayed by default in SUMMARY, displays the following fields for each task:

Starting time

Time the task started.

User and host name

User who submitted the task, host from which the task was submitted, in the format *user_name@host*.

PID

UNIX process ID of the task.

Execution host

Host on which the command was run.

Command line

Complete command line that was executed.

CWD

Current working directory of the task.

Completion time

Time at which the task completed.

Exit status

UNIX exit status of the task.

Files

Reads `lsf.acct.`*host_name*

See also

lsf.acct(5), lsacctmrg(1), `res`(8), bhist(1)

lsacctmrg

merges task log files

Synopsis

lsacctmrg [**-f**] *logfile_name ... target_logfile_name*

lsacctmrg [**-h** | **-V**]

Description

Merges specified task log files into the specified target file in chronological order according to completion time.

All files must be in the format specified in `lsf.acct` (see `lsf.acct`(5)).

Options

-f

Overwrites the target file without prompting for confirmation.

logfile_name ...

Specify log files to be merged into the target file, separated by spaces. Specify either an absolute or a relative path.

target_logfile_name

Specify the file into which all log files are to be merged. Specify either an absolute or a relative path. The target file cannot be part of the files to be merged.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

lsf.acct(5), `res`(8)

lsadmin

administrative tool for LSF

Synopsis

lsadmin *subcommand*

lsadmin [**-h** | **-V**]

Description

Caution:

This command can only be used by LSF administrators.

`lsadmin` is a tool that executes privileged commands to control LIM and RES operations in the LSF cluster.

If no subcommands are supplied for `lsadmin`, `lsadmin` prompts for subcommands from the standard input.

For subcommands for which multiple host names or host groups can be specified, do not enclose the multiple names in quotation marks.

Subcommand List

ckconfig [**-v**]

reconfig [**-f**] [**-v**]

limstartup [**-f**] [*host_name* ... |**all**]

limshutdown [**-f**] [*host_name* ... | **all**]

limrestart [**-v**] [**-f**] [*host_name* ... | **all**]

limlock [**-l** *time_seconds*]

limunlock

resstartup [**-f**] [*host_name* ... | **all**]

resshutdown [**-f**] [*host_name* ... | **all**]

resrestart [**-f**] [*host_name* ... | **all**]

reslogon [**-c** *cpu_time*] [*host_name* ... | **all**]

reslogoff [*host_name* ... | **all**]

limdebug [**-c "***class_name* ...**"**] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [**"***host_name* ...**"**]

resdebug [**-c "***class_name***"**] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [**"***host_name* ...**"**]

limtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [**"***host_name* ...**"**]

restime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [**"***host_name* ...**"**]

showconf lim [*host_name* … | **all**]

lsflic [*feature_name* |**-c** |**-s** | **-l**]

help [*subcommand* ...] | **?** [*subcommand* ...]

quit

-h

-V

Options

subcommand

Executes the specified subcommand. See Usage section.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Usage

ckconfig [-v]

Checks LSF configuration files.

-v

Displays detailed messages about configuration file checking.

reconfig [-f] [-v]

Restarts LIMs on all hosts in the cluster. You should use `reconfig` after changing configuration files. The configuration files are checked before all LIMs in the cluster are restarted. If the configuration files are not correct, reconfiguration is not initiated.

If LSF_MASTER_LIST is specified in `lsf.conf`, you are prompted to confirm the reconfiguration for only the master candidate hosts.

-f

Disables user interaction and forces LIM to restart on all hosts in the cluster if no fatal errors are found. This option is useful in batch mode.

-v

Displays detailed messages about configuration file checking.

limstartup [-f] [*host_name* ... |all]

Starts LIM on the local host if no arguments are specified.

Starts LIMs on the specified hosts or on all hosts in the cluster if the word all is the only argument provided. You are prompted to confirm LIM startup.

Only root and users listed in the parameter LSF_STARTUP_USERS in `lsf.sudoers` (5) can use the all and -f options to start LIM as root.

These users must also be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. If permission to start up LIMs as root is not configured, `limstartup` starts up LIMs as yourself after your confirmation.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

-f

Disables interaction and does not ask for confirmation for starting LIMs.

limshutdown [-f] [*host_name* ... | all]

Shuts down LIM on the local host if no arguments are supplied.

Shuts down LIMs on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm LIM shutdown.

-f

Disables interaction and does not ask for confirmation for shutting down LIMs.

limrestart [-v] [-f] [*host_name* ... | all]

Restarts LIM on the local host if no arguments are supplied.

Restarts LIMs on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm LIM restart.

`limrestart` should be used with care. Do not make any modifications until all the LIMs have completed the startup process. If you execute `limrestart` *host_name*`...` to restart some of the LIMs after changing the configuration files, but other LIMs are still running the old configuration, confusion arises among these LIMs. To avoid this situation, use `reconfig` instead of `limrestart`.

-v

Displays detailed messages about configuration file checking.

-f

Disables user interaction and forces LIM to restart if no fatal errors are found. This option is useful in batch mode. `limrestart` -f all is the same as `reconfig -f`.

limlock [-l *time_seconds*]

Locks LIM on the local host until it is explicitly unlocked if no time is specified. When a host is locked, LIM's load status becomes `lockU`. No job is sent to a locked host by LSF.

-l time_seconds

The host is locked for the specified time in seconds.

LSF suspends all non-exclusive jobs running on the host. This is useful if a machine is running an exclusive job requiring all the available CPU time and/or memory. If LSB_DISABLE_LIMLOCK_EXCL=y (to enable preemption of exclusive jobs, for example) LSF suspends all jobs, including exclusive jobs.

limunlock

Unlocks LIM on the local host.

resstartup [-f] [*host_name* ... | all]

Starts RES on the local host if no arguments are specified.

Starts RESs on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm RES startup.

Only root and users defined by the LSF_STARTUP_USERS parameter in `lsf.sudoers`(5) can use the all and -f options to start RES as root.

These users must be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. For root installation to work properly, `lsadmin` must be installed as a `setuid` to root program.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

-f

Disables interaction and does not ask for confirmation for starting RESs.

resshutdown [-f] [*host_name* ... | all]

Shuts down RES on the local host if no arguments are specified.

Shuts down RESs on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm RES shutdown.

If RES is running, it keeps running until all remote tasks exit.

-f

Disables interaction and does not ask for confirmation for shutting down RESs.

resrestart [-f] [*host_name* ... | all]

Restarts RES on the local host if no arguments are specified.

Restarts RESs on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm RES restart.

If RES is running, it keeps running until all remote tasks exit. While waiting for remote tasks to exit, another RES is restarted to serve the new queries.

-f

Disables interaction and does not ask for confirmation for restarting RESs.

reslogon [-c *cpu_time*] [*host_name* ... | all]

Logs all tasks executed by RES on the local host if no arguments are specified.

Logs tasks executed by RESs on the specified hosts or on all hosts in the cluster if all is specified.

RES writes the task's resource usage information into the log file `lsf.acct.`*host_name*. The location of the log file is determined by LSF_RES_ACCTDIR defined in `lsf.conf`. If LSF_RES_ACCTDIR is not defined, or RES cannot access it, the log file is created in `/tmp` instead.

-c *cpu_time*

Logs only tasks that use more than the specified amount of CPU time. The amount of CPU time is specified by cpu_time in milliseconds.

reslogoff [*host_name* ... | all]

Turns off RES task logging on the specified hosts or on all hosts in the cluster if all is specified.

If no arguments are specified, turns off RES task logging on the local host.

limdebug [-c "*class_name* ..."] [-l *debug_level*] [-f *logfile_name*] [-o] ["*host_name* ..."]

Sets the message log level for LIM to include additional information in log files. You must be root or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

class_name=0 (no additional classes are logged)

debug_level=0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

logfile_name=current LSF system log file in the LSF system log file directory, in the format *daemon_name*.`log`.*host_name*

host_name= local host (host from which command was submitted)

In MultiCluster, debug levels can only be set for hosts within the same cluster. For example, you could not set debug or timing levels from a host in `clusterA` for a host in `clusterB`. You need to be on a host in `clusterB` to set up debug or timing levels for `clusterB` hosts.

-c "*class_name* ..."

Specify software classes for which debug messages are to be logged. If a list of classes is specified, they must be enclosed in quotation marks and separated by spaces.

Possible classes:

LC_AFS - Log AFS messages

LC_AUTH - Log authentication messages

LC_CHKPNT - log checkpointing messages

LC_COMM - Log communication messages

LC_CONF - Print out all parameters in `lsf.conf` (and `ego.conf`)

LC_DCE - Log messages pertaining to DCE support

LC_EXEC - Log significant steps for job execution

LC_FILE - Log file transfer messages

LC_HANG - Mark where a program might hang

LC_LICENCE or LC_LICENSE - Log license management messages

LC_MULTI - Log messages pertaining to MultiCluster

LC_PIM - Log PIM messages

LC_SIGNAL - Log messages pertaining to signals

LC_TRACE - Log significant program walk steps

LC_XDR - Log everything transferred by XDR

Default: 0 (no additional classes are logged)

Note:

Classes are also listed in `lsf.h`.

-l *debug_level*

Specify level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

Possible values:

`0` - LOG_DEBUG level in parameter LSF_LOG_MASK in `lsf.conf`.

`1` - LOG_DEBUG1 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

`2` - LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

`3` - LOG_DEBUG3 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2, LOG_DEBUG1, and LOG_DEBUG levels.

Default: `0` (LOG_DEBUG level in parameter LSF_LOG_MASK)

-f *logfile_name*

Specify the name of the file into which debugging messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file created has the following format:

logfile_name`.`*daemon_name*`.log.`*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, no log file is created.

Default: current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*.

-o

Turns off temporary debug settings and reset them to the daemon starting state. The message log level is reset back to the value of LSF_LOG_MASK and classes are reset to the value of LSF_DEBUG_RES, LSF_DEBUG_LIM.

Log file is reset back to the default log file.

"*host_name* ..."

Sets debug settings on the specified host or hosts.

Default: local host (host from which command was submitted)

resdebug [-c "*class_name*"] [-l *debug_level*] [-f *logfile_name*] [-o] ["*host_name* ..."]

Sets the message log level for RES to include additional information in log files. You must be the LSF administrator to use this command, not root.

See description of `limdebug` for an explanation of options.

limtime [-l *timing_level*] [-f *logfile_name*] [-o] ["*host_name* ..."]

Sets timing level for LIM to include additional timing information in log files. You must be root or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

timing_level=no timing information is recorded

logfile_name=current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*

host_name=local host (host from which command was submitted)

In MultiCluster, timing levels can only be set for hosts within the same cluster. For example, you could not set debug or timing levels from a host in `clusterA` for a host in `clusterB`. You need to be on a host in `clusterB` to set up debug or timing levels for `clusterB` hosts.

-l *timing_level*

Specifies detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

Valid values: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

Default: undefined (no timing information is logged)

-f *logfile_name*

Specify the name of the file into which timing messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file created has the following format:

logfile_name`.`*daemon_name*`.log.`*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, no log file is created.

Note:

Both timing and debug messages are logged in the same files.

Default: current LSF system log file in the LSF system log file directory, in the format *daemon_name*.`log`.*host_name*.

-o

Turns off temporary timing settings and resets them to the daemon starting state. The timing level is reset back to the value of the parameter for the corresponding daemon (LSF_TIME_LIM, LSF_TIME_RES).

Log file is reset back to the default log file.

"*host_name* ..."

Sets the timing level on the specified host or hosts.

Default: local host (host from which command was submitted)

restime [-l *timing_level*] [-f *logfile_name*] [-o] ["*host_name* ..."]

Sets timing level for RES to include additional timing information in log files. You must be the LSF administrator can use this command, not root.

See description of `limtime` for an explanation of options.

showconf lim [*host_name* ... | all]

Display all configured parameters and their values set in `lsf.conf` or `ego.conf` that affect `lim`.

Use `lsadmin showconf lim` to display the parameters configured in `lsf.conf` and `ego.conf` that apply to root LIM. By default, `lsadmin` displays the local LIM parameters. You can optionally specify the host to display the LIM parameters.

In a MultiCluster environment, `lsadmin showconf` only displays the parameters of daemons on the local cluster.

Running `lsadmin showconf` from a master candidate host reaches all server hosts in the cluster. Running `lsadmin showconf` from a slave-only host may not be able to reach other slave-only hosts.

You cannot run `lsadmin showconf lim` from client hosts. `lsadmin` shows only server host configuration, not client host configuration.

`lsadmin showconf` only displays the values used by LSF.

LIM reads EGO_MASTER_LIST from wherever it is defined. You can define either LSF_MASTER_LIST in `lsf.conf` or EGO_MASTER_LIST in `ego.conf`. LIM reads `lsf.conf` first, and ego.conf if EGO is enabled in the LSF cluster. LIM only takes the value of LSF_MASTER_LIST if EGO_MASTER_LIST is not defined at all in `ego.conf`.

For example, if EGO is enabled in the LSF cluster, and you define LSF_MASTER_LIST in `lsf.conf`, and EGO_MASTER_LIST in `ego.conf`, `lsadmin showconf` displays the value of EGO_MASTER_LIST in `ego.conf`.

If EGO is disabled, `ego.conf` not loaded, so whatever is defined in `lsf.conf` is displayed.

lsflic [*feature_name* | -c |-s | -l]

Displays LSF license usage. The parameter LSF_MONITOR_LICENSE_TOOL in `lsf.conf` enables data collection for lsflic.

feature_name

Displays current LSF license usage. An example license is **lsf_client**.

-c

Displays the licenses by class level for the cluster. Two numbers display for each class: number of hosts/number of cores.

License class levels are (from most powerful to least powerful) E, S, and B. A B license can be substituted by an S or E license. An S license can be substituted by an E license.

-s

Displays license information including any substitutions made between licenses.

License class levels are (from most powerful to least powerful) E, S, and B. A B license can be substituted by an S or E license. An S license can be substituted by an E license.

-l

Displays all LSF license usage, including the class level and any substitutions.

help [*subcommand* ...] | ? [*subcommand* ...]

Displays the syntax and functionality of the specified commands. The commands must be explicit to `lsadmin`.

From the command prompt, you may use `help` or `?`.

quit

Exits the `lsadmin` session.

See also

`ls_limcontrol`, `ls_rescontrol`, `ls_readconfenv`, `ls_gethostinfo`, `ls_connect`, `ls_initrex`, `lsf.conf`, `lsf.sudoers`, `lsf.acct`, bmgroup, busers

lsclusters

displays configuration information about LSF clusters

Synopsis

lsclusters [**-l**] [*cluster_name* ...]

lsclusters [**-h** | **-V**]

Description

Displays configuration information about LSF clusters.

By default, returns information about the local cluster and all other clusters that the local cluster is aware of (all clusters defined in the `RemoteClusters` section of `lsf.cluster.`*cluster_name* if that section exists, otherwise all clusters defined in `lsf.shared`).

Options

-l

Long format. Displays additional information.

***cluster_name* ...**

Only displays information about the specified clusters.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Default Output

The information includes: cluster name, cluster master host, primary cluster administrator's login name, total number of hosts in the cluster, and the number of server hosts in the cluster.

A listing of the clusters is displayed with the following fields:

CLUSTER_NAME

The name of the cluster.

STATUS

The current status of the cluster. Possible values are:

ok

The cluster is in normal load sharing state, and exchanges load information with the local cluster.

unavail

The cluster is unavailable.

MASTER_HOST

The name of the cluster's master host.

ADMIN

The user account name of the cluster's primary LSF administrator.

HOSTS

Number of LSF static client and server hosts in the cluster. The HOSTS field does not include floating clients.

SERVERS

Number of LSF server hosts in the cluster.

Long Format (-l)

If this option is specified, the command also lists available resource names, host types, host models and cluster administrator's login names, and whether local cluster accepts or sends interactive jobs to this cluster.

See also

`ls_info`, `ls_policy`, `ls_clusterinfo` `lsf.cluster`

lseligible

displays whether a task is eligible for remote execution

Synopsis

lseligible [**-r**] [**-q**] [**-s**] *task*

lseligible [**-h** | **-V**]

Description

Displays whether the specified task is eligible for remote execution.

By default, only tasks in the remote task list are considered eligible for remote execution.

Options

-q

Quiet mode. Displays only the resource requirement string defined for the task. The string ELIGIBLE or NON-ELIGIBLE is omitted.

-r

Remote mode. Considers eligible for remote execution any task not included in the local task list.

-s

Silent mode. No output is produced. The -q and -s options are useful for shell scripts that operate by testing the exit status (see DIAGNOSTICS).

task

Specify a command.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

If the task is eligible, the string ELIGIBLE followed by the resource requirements associated with the task are printed to stdout. Otherwise, the string NON-ELIGIBLE is printed to stdout.

If `lseligib`le prints ELIGIBLE with no resource requirements, the task has the default requirements of CPU consumption and memory usage.

Diagnostics

`lseligible` has the following exit statuses:

0 Task is eligible for remote execution

1 Command is to be executed locally

-1 Syntax errors

-10 A failure is detected in the LSF system

See also

`ls_eligible`(3), lsrtasks(1), lsf.task(5)

lsfinstall

runs `lsfinstall`, the Platform LSF installation and configuration script

Synopsis

lsfinstall -f install.configlsfinstall -s -f slave.configlsfinstall -h

Description

`lsfinstall` runs the LSF installation scripts and configuration utilities to install a new Platform LSF cluster or upgrade LSF from a previous release.

To install a fully operational LSF cluster that all users can access, you should install as root.

You can run `lsfinstall` as a non-root user, with limitations.

Required install.config variables

- LSF_TOP="/*path*"
- LSF_ADMINS="*user_name* [*user_name*...]"
- LSF_CLUSTER_NAME="*cluster_name*"

Required slave.config variables

If you use `slave.config` for dynamic slave host installation, the following parameters are required:

- LSF_TOP="/*path*"
- LSF_TARDIR="/*path*"
- LSF_SERVER_HOSTS="*host_name* [*host_name* ...]"

Variables that require an absolute path

- LSF_LICENSE="/*path*/*license_file*"
- LSF_TOP="/*path*"
- LSF_TARDIR="/*path*"

What lsfinstall does

Before installing and configuring LSF, `lsfinstall` checks the installation prerequisites, and outputs the results to `lsfprechk.rpt`. `lsfinstall` writes any unrecoverable errors to the `Install.err file` and exits. You must correct these errors before continuing to install and configure LSF.

During installation, `lsfinstall` logs installation progress in the `Install.log` file, calls other utilities to uncompress, extract and copy product files, installs a license, and configures the cluster.

After installation, you should run `hostsetup` to set up each server host in the cluster. After setting up the server hosts, you should start your cluster and test the installation by running some basic commands.

Where lsfinstall is located

`lsfinstall` is included in the LSF installation script tar file `lsf7Update4_lsfinstall.tar.Z` and is located in the `lsf7Update4_lsfinstall` directory created when you uncompress and extract installation script tar file.

After installation, `lsfinstall` is located in `LSF_TOP/7.0/install/`.

Options

-f *option_file*

Name of the file containing the installation options. The file can be any name you choose. The name of the default template file for normal installation is `install.config`. To install slave hosts for dynamic host configuration, use the template file `slave.config`.

-s

Install a dynamic slave host.

Specify installation options in the `slave.config` file.

Required parameters:

- LSF_SERVER_HOSTS="*host_name* [*host_name* ...]"
- LSF_TOP="/*path*"
- LSF_TARDIR="/*path*"

Optional parameters:

LSF_LIM_PORT=*port_number*

If the master host does not use the default LSF_LIM_PORT, you must specify the same LSF_LIM_PORT defined in `lsf.conf` on the master host.

LSF_LOCAL_RESOURCES="*resource* ..."

Defines the local resources for a dynamic host.

- For numeric resources, defined name-value pairs:

  ```
  "[resourcemap value*resource_name]"
  ```

- For Boolean resources, the value is the resource name in the form:

  ```
  "[resource resource_name]"
  ```

For example:

```
LSF_LOCAL_RESOURCES="[hostname hostA] [server 1] [resourcemap 1*verilog]
[resource linux]"
```

Tip:

If LSF_LOCAL_RESOURCES are already defined in a local `lsf.conf` on the slave host, `lsfinstall` does not add resources you define in LSF_LOCAL_RESOURCES in `slave.config`.

`lsfinstall` creates a local `lsf.conf` for the slave host, which sets the following parameters:

- LSF_CONFDIR="/*path*"
- LSF_GET_CONF=lim
- LSF_LIM_PORT=*port_number*

- LSF_LOCAL_RESOURCES="*resource* ..."
- LSF_SERVER_HOSTS="*host_name* [*host_name* ...]"
- LSF_VERSION=7.0

-h

Prints command usage and exits.

See also

`lsf.conf`, `install.config`, `slave.config`

lsfmon

installs or uninstalls LSF Monitor

Synopsis

lsfmon -install

lsfmon -remove

Description

Installs or uninstalls LSF Monitor in an existing cluster.

LSF Monitor runs on Microsoft Windows and allows you to use Windows Performance Monitor to chart information about the LSF cluster.

The LSF Monitor service runs under the account of an LSF cluster administrator.

Options

-install

Installs LSF Monitor on the host.

-remove

Removes LSF Monitor from the host.

lsfrestart

restarts LIM, RES, `sbatchd` and `mbatchd` on all hosts in the cluster

Synopsis

lsfrestart [**-f** | **-h** | **-V**]

Description

Caution:

This command can only be used by root or users listed in lsf.sudoers.

Restarts LIM, RES, `sbatchd` and `mbatchd`, in that order, on all hosts in the local cluster.

By default, prompts for confirmation of the next operation if an error is encountered.

To be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly.
- You must be able to run the `rsh` or `ssh` command across all LSF hosts without having to enter a password. See your operating system documentation for information about configuring the `rsh` and `ssh` commands.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

Options

-f

Force mode. Continues to restart daemons even if an error is encountered.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

lsadmin(8), badmin(8), lsfshutdown(8), lsf.sudoers(5)

lsfshutdown

shuts down LIM, RES, `sbatchd` and `mbatchd` on all hosts in the cluster

Synopsis

lsfshutdown [**-f** | **-h** | **-V**]

Description

Caution:

This command can only be used by root or users listed in lsf.sudoers.

Shuts down sbatchd, RES, LIM, and mbatchd, in that order, on all hosts.

By default, prompts for confirmation of the next operation if an error is encountered.

To be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly.
- You must be able to run the `rsh` or `ssh` command across all LSF hosts without having to enter a password. See your operating system documentation for information about configuring the `rsh` and `ssh` commands.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

Options

-f

Force mode. Continues to shut down daemons even if an error is encountered.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

lsadmin(8), badmin(8), lsfrestart(8), lsf.sudoers(5)

lsfstartup

starts LIM, RES, `sbatchd`, and `mbatchd` on all hosts in the cluster

Synopsis

lsfstartup [**-f**]

lsfstartup [**-h** | **-V**]

Description

Caution:

This command can only be used by root or users listed in lsf.sudoers.

Starts LIM, RES, sbatchd, and mbatchd, in that order, on all hosts.

By default, prompts for confirmation of the next operation if an error is encountered.

If LSF daemons are already running, use `lsfrestart` instead, or use `lsfshutdown` and shut down the running daemons before you use `lsfstartup`.

To be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly.
- You must be able to run the `rsh` or `ssh` command across all LSF hosts without having to enter a password. See your operating system documentation for information about configuring the `rsh` and `ssh` commands.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

Options

-f

Force mode. Continues to start daemons even if an error is encountered.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

lsadmin(8), badmin(8), lsfshutdown(8), lsfrestart(8), lsf.sudoers(5)

lsgrun

executes a task on a set of hosts

Synopsis

lsgrun [**-i**] [**-p** | **-P** | **-S**] [**-v**] **-f** *host_file* | **-m** *host_name* ... | **-n** *num_hosts* [**-R "***res_req***"**] [*command* [*argument* ...]]

lsgrun [**-h** | **-V**]

Description

Executes a task on the specified hosts. `lsgrun` is useful for fast global operations such as starting daemons, replicating files to or from local disks, looking for processes running on all hosts, checking who is logged in on each host, and so on. The hosts can be specified using a host file, a list of host names or by letting the system select the hosts. If LSB_DISABLE_LIMLOCK_EXCL=y (to enable preemption of exclusive jobs, for example), you can use `lsgrun` to start a task on hosts that are currently running exclusive jobs.

By default:

- `lsgrun` is not interactive.
- The specified task is executed sequentially on hosts with full pseudo `tty` support.
- `lsgrun` does not create a pseudo-terminal.
- LSF uses as many processors as available to run the specified task.
- The resource requirement for host selection is `r15s:pg`.
- The prompt `Command>` is displayed to allow users to type in a command (task) terminated by a `CTRL-D` or `EOF`. The command is then executed on the specified hosts.

Options

-i

Interactive operation mode. You are asked whether the task is to be executed on all hosts. If you answer y, the task is started on all specified hosts; otherwise, you are asked to specify hosts interactively.

-P

Creates a pseudo-terminal on UNIX hosts. This is necessary to run programs requiring a pseudo-terminal (for example, `vi`).

This option is not supported on Windows.

-p

Parallel run mode. Executes the task on all hosts simultaneously and without pseudo `tty` support.

If this option is specified and the -P option is specified, the -P option is ignored.

This option is useful for fast start-up of tasks. However, any output from remote tasks arrive at the terminal in arbitrary order, depending on task execution speeds on individual hosts.

-S

Creates a pseudo-terminal with shell mode support on UNIX hosts.

Shell mode support is required for running interactive shells or applications that redefine the `CTRL-C` and `CTRL-Z keys` (such as `jove`).

This option is not supported on Windows.

-v

Verbose mode. Displays the name of the host or hosts running the task.

-f *host_file*

Either -f *host_file*, -m *host_name* or -n *num_processors* is required.

Executes the task on all hosts listed in the *host_file*.

Specify a file that contains a list of host names. Host names must be separated by white space characters (for example, SPACE, TAB, and NEWLINE).

This option is exclusive of options -n, -R, and -m.

-m *host_name* ...

Either -f *host_file*, -m *host_name* or -n *num_processors* is required.

Executes the task on all specified hosts.

Specify hosts on which to execute the task. If multiple host names are specified, the host names must be enclosed by " or ' and separated by white space.

This option is exclusive of options -n, -R, and -f.

-n *num_hosts*

Either -f *host_file*, -m *host_name* or -n *num_hosts* is required.

Executes the task in a cluster with the required number of available hosts.

One host may be used to start several tasks if the host is multiprocessor. This option can be used together with option -R to select desired hosts.

This option is exclusive of options -m and -f.

-R "*res_req*"

Executes the task on hosts with the required resource requirements.

Specify the resource requirement expression for host selection. The resource requirement is used to choose from all hosts with the same host type as the local host, unless a "`type == value`" exists in res_req to specify otherwise.

This option can be used together with option -n to choose a specified number of processors to run the task.

Exclusive resources need to be explicitly specified within the resource requirement string. For example, you defined a resource called `bigmem` in `lsf.shared` and defined it as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command to submit a task to run on `hostE`:

```
lsgrun -R "bigmem" myjob
```

or

```
lsgrun -R "defined(bigmem)" myjob
```

If the -m option is specified with a single host name, the -R option is ignored.

***command* [*argument* ...]**

Specify the command to execute. This must be the last argument on the command line.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Diagnostics

Exit status is 0 if all commands are executed correctly.

Otherwise, the exit status is the first non-zero status returned by a remotely executed task. `lsgrun` executes the task on all hosts even if some have non-zero exit status.

Exit status is -10 if a problem is detected in LSF.

See also

lsrun, lsplace

lshosts

displays hosts and their static resource information

Synopsis

lshosts [**-w** | **-l**] [**-R "***res_req***"**] [*host_name* | *cluster_name*] ...

lshosts -s [*resource_name* ...]

lshosts [**-h** | **-V**]

Description

Displays static resource information about hosts.

By default, returns the following information: host name, host type, host model, CPU factor, number of CPUs, total memory, total swap space, whether or not the host is a server host, and static resources. Displays information about all hosts in the cluster. See `lsf.cluster`.

In MultiCluster job forwarding model, the default behavior is to return the following information: host name, host type, host model, CPU factor, number of CPUs, total memory, total swap space, whether or not the host is a server host, and static resources. Displays information about all hosts in the local cluster and for all hosts in equivalent remote clusters that the local cluster sees. See `lsf.cluster`(5).

In MultiCluster resource leasing model, returns information about hosts in the local cluster.

The -s option displays information about the static resources (shared or host-based) and their associated hosts.

Options

-l

Displays host information in a long multi-line format. In addition to the default fields, displays additional information, including maximum `/tmp` space, the number of local disks, the execution priority for remote jobs, load thresholds, run windows, and the license classes used or needed.

-w

Displays host information in wide format. Fields are displayed without truncation.

-R "*res_req*"

Only displays information about the hosts that satisfy the resource requirement expression. For more information about resource requirements, see *Administering Platform LSF*. The size of the resource requirement string is limited to 512 bytes. LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

In MultiCluster, only displays information about the hosts in the local cluster that satisfy the resource requirement expression.

***host_name*...| *cluster_name*...**

Only displays information about the specified hosts. Do not use quotes when specifying multiple hosts.

For MultiCluster, displays information about hosts in the specified clusters. The names of the hosts belonging to the cluster are displayed instead of the name of the cluster. Do not use quotes when specifying multiple clusters.

-s [*resource_name* ...]

Displays information about the specified resources. The resources must be static resources (shared or host-based). If no resource is specified, then displays information about all resources. Returns the following information: the resource names, the values of the resources, and the resource locations.

-h

Prints command usage to `stderr` and exits.

-V

Prints the LSF release version to `stderr` and exits.

Output: Host-Based Default

Displays the following fields:

HOST_NAME

The name of the host. This display field is truncated.

type

The host type. This display field is truncated.

With MultiCluster, if the host type of a remote cluster's host is not defined in the local cluster, the keyword unknown is displayed.

model

The host model. This display field is truncated.

With MultiCluster, if the host model of a remote cluster's host is not defined in the local cluster, the keyword unknown is displayed.

cpuf

The relative CPU performance factor. The CPU factor is used to scale the CPU load value so that differences in CPU speeds are considered. The faster the CPU, the larger the CPU factor.

The CPU factor of a host with an `unknown` host type is 1.0.

ncpus

The number of processors on this host.

If LSF_ENABLE_DUALCORE=Y in `lsf.conf` for multi-core CPU hosts, displays the number of cores instead of physical CPUs.

If EGO is enabled in the LSF cluster and EGO_DEFINE_NCPUS is specified in `lsf.conf` or `ego.conf`, the appropriate value for ncpus is displayed, depending on the value of EGO_DEFINE_NCPUS:

- EGO_DEFINE_NCPUS=procs—ncpus=number of processors
- EGO_DEFINE_NCPUS=cores—ncpus=number of processors x number of cores per processor
- EGO_DEFINE_NCPUS=threads—ncpus=number of processors x number of cores per processor x number of threads per core

EGO_DEFINE_NCPUS=cores is the same as setting LSF_ENABLE_DUALCORE=Y.

nprocs

The number of physical processors per CPU configured on a host.

ncores

The number of cores per processor configured on a host.

nthreads

The number of threads per core configured on a host.

maxmem

The maximum amount of physical memory available for user processes.

By default, the amount is displayed in KB. The amount may appear in MB depending on the actual system memory. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (GB, TB, PB, or EB).

maxswp

The total available swap space.

By default, the amount is displayed in KB. The amount may appear in MB depending on the actual system swap space. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (GB, TB, PB, or EB).

server

Indicates whether the host is a server or client host. “Yes” is displayed for LSF servers. “No” is displayed for LSF clients. “Dyn” is displayed for dynamic hosts.

RESOURCES

The Boolean resources defined for this host, denoted by resource names, and the values of external numeric and string static resources. See `lsf.cluster`(5), and `lsf.shared` (5) on how to configure external static resources.

Output: Host Based -l Option

In addition to the above fields, the -l option also displays the following:

ndisks

The number of local disk drives directly attached to the host.

maxtmp

The maximum `/tmp` space in MB configured on a host.

rexpri

UNIX only. The execution priority of remote jobs run by the RES. rexpri is a number between -20 and 20, with -20 representing the highest priority and 20 the lowest. The default rexpri is 0, which corresponds to the default scheduling priority of 0 on BSD-based UNIX systems and 20 on System V-based systems.

nprocs

The number of physical processors per CPU configured on a host.

ncores

The number of cores per processor configured on a host.

nthreads

The number of threads per core configured on a host.

RUN_WINDOWS

The time windows during which LIM considers the host as available to execute remote jobs. These run windows have the same function for LSF hosts as dispatch windows have for LSF hosts.

LICENSES_ENABLED

The licenses that are enabled for each specified host.

LICENSE_NEEDED

The required banded license class for each specified host.

If LSF_ENABLE_DUALCORE=Y in `lsf.conf` for dual-core CPU hosts, `lshosts -l` also displays if dual-core CPU license is enabled for the hosts and the number of dual-core licenses needed.

LOAD_THRESHOLDS

The thresholds for scheduling interactive jobs. If a load index exceeds the load threshold (or falls below the load threshold, for decreasing load indices), the host status is changed to “busy.” If the threshold is displayed as a dash “-”, the value of that load index does not affect the host’s status.

Output: Resource-Based -s Option

Displays the static resources (shared or host-based). Each line gives the value and the associated hosts for the static resource. See `lsf.shared`, and `lsf.cluster` on how to configure static shared resources.

The following fields are displayed:

RESOURCE

The name of the resource.

VALUE

The value of the static resource.

LOCATION

The hosts that are associated with the static resource.

Files

Reads `lsf.cluster.`*`cluster_name`*.

See also

`ls_info`, `ls_policy`, `ls_gethostinfo`, `lsf.cluster`, `lsf.shared`

lsid

displays the current LSF version number, the cluster name, and the master host name

Synopsis

lsid [**-h** | **-V**]

Description

Displays the current LSF version number, the cluster name, and the master host name.

The master host is dynamically selected from all hosts in the cluster.

In MultiCluster, the master host is dynamically selected from all hosts in the local cluster.

Options

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Files

The host names and cluster names are defined in `lsf.cluster.`*cluster_name* and `lsf.shared`, respectively.

See also

`ls_getclustername`(3), `ls_getmastername`(3), lsinfo(1)

lsinfo

displays load sharing configuration information

Synopsis

lsinfo [**-l**] [**-m** | **-M**] [**-r**] [**-t**] [*resource_name* ...]

lsinfo [**-h** | **-V**]

Description

By default, displays all load sharing configuration information including resource names and their meanings, host types and models, and associated CPU factors known to the system.

By default, displays information about all resources. Resource information includes resource name, resource type, description, and the default sort order for the resource.

You can use resource names in task placement requests.

Use this command with options to selectively view configured resources, host types, and host models.

Options

-l

Displays resource information in a long multi-line format. Additional parameters are displayed including whether a resource is built-in or configured, and whether the resource value changes dynamically or is static. If the resource value changes dynamically then the interval indicates how often it is evaluated.

-M

Displays information about all host models in the file `lsf.shared`.

-m

Displays only information about host models that exist in the cluster.

-r

Displays only information about configured resources.

-t

Displays only information about configured host types. See `lsload(1)` and `lshosts(1)`.

***resource_name* ...**

Displays only information about the specified resources.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output: -l option

The `-l` option displays all information available about load indices.

TYPE

Indicates whether the resource is numeric, string, or Boolean.

ORDER

- Inc—If the numeric value of the load index increases as the load it measures increases, such as CPU utilization (`ut`).
- Dec—If the numeric value decreases as the load increases.
- N/A—If the resource is not numeric.

INTERVAL

The number of seconds between updates of that index. Load indices are updated every INTERVAL seconds. A value of 0 means the value never changes.

BUILTIN

If BUILTIN is `Yes`, the resource name is defined internally by LIM. If BUILTIN is `No`, the resource name is site-specific defined externally by the LSF administrator.

DYNAMIC

If DYNAMIC is `Yes` the resource is a load index that changes over time. If DYNAMIC is `No` the resource represents information that is fixed such as the total swap space on a host. Resources are Static or Boolean.

RELEASE

Applies to numeric shared resources only, such as floating licenses. Indicates whether LSF releases the resource when a job using the resource is suspended. When a job using a shared resource is suspended, the resource is held or released by the job depending on the configuration of the RELEASE parameter in `lsf.shared`.

`No` indicates the resource is held. `Yes` indicates the resource is released.

CONSUMABLE

If CONSUMABLE is Yes the resource is a static or dynamic numeric resource that is specified as consumable in the Resource section of lsf.shared.

See also

lshosts, lsload, `lsf.shared`, `ls_info`, `ls_policy`

lsload

displays load information for hosts

Synopsis

lsload [**-l**] [**-N** | **-E**] [**-I** *load_index*[**:***load_index*] ...] [**-n** *num_hosts*] [**-R** *res_req*] [*host_name* ... | *cluster_name* ...]

lsload -s [*resource_name* ...]

lsload [**-h** | **-V**]

Description

Displays load information for hosts. Load information can be displayed on a per-host basis, or on a per-resource basis.

By default, displays load information for all hosts in the local cluster, per host.

With MultiCluster, also displays load information for all hosts in equivalent clusters (see `lsf.cluster(5)`).

By default, displays raw load indices.

By default, load information for resources is displayed according to CPU and paging load.

Options

-l

Long format. Displays load information without truncation along with additional fields for I/O and external load indices.

This option overrides the index names specified with the -I option.

-N

Displays normalized CPU run queue length load indices.

-E

Displays effective CPU run queue length load indices. Options -N and -E are mutually exclusive.

-w

Displays load information in wide format. Fields are displayed without truncation.

-I *load_index*[:*load_index*] ...

Displays only the specified load indices. Separate multiple index names with colons (for example, `r1m:pg:ut`).

Specify any built-in load index. Specify external load indices only for host-based resources that are numeric and dynamic (you cannot specify external load indices for shared, string or Boolean resources).

-n *num_hosts*

Displays only load information for the requested number of hosts. Information for up to num_hosts hosts that best satisfy the resource requirements is displayed.

-R *res_req*

Displays only load information for hosts that satisfy the specified resource requirements. See *Administering Platform LSF* for a list of built-in resource names.

Load information for the hosts is sorted according to load on the specified resources.

If *res_req* contains special resource names, only load information for hosts that provide these resources is displayed (run `lshosts` to find out what resources are available on each host).

If one or more host names are specified, only load information about the hosts that satisfy the resource requirements is displayed.

With MultiCluster, when a cluster name is specified, displays load information of hosts in the specified cluster that satisfy the resource requirements.

***host_name* ... | *cluster_name* ...**

Displays only load information for the specified hosts.

With MultiCluster, displays only load information for hosts in the specified clusters.

-s [*resource_name* ...]

Displays information about all dynamic resources configured in the cluster, or about the specified resources only. Specify dynamic resources (shared or host-based).

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Host-Based Output (default output)

Built-in load indices include `r15s`, `r1m`, `r15m`, `ut`, `pg`, `io`, `ls`, `it`, `swp`, `mem` and `tmp`. External load indices are configured in the file `lsf.cluster.`*cluster_name* (see `lsf.cluster(5)`). The selection and order sections of res_req control for which hosts are displayed and how the information is ordered.

The display includes the following fields:

HOST_NAME

Standard host name used by LSF, typically an Internet domain name with two components.

status

Status of the host. A minus sign (-) may precede the status, indicating that RES is not running on the host.

Possible statuses are:

ok

The host is in normal load sharing state and can accept remote jobs. The ok status indicates that the Load Information Manager (LIM) is unlocked and that both LIM and the Remote Execution Server (RES) are running.

-ok

The (LIM) on the host is running but RES is unreachable.

busy

The host is overloaded because some load indices exceed configured thresholds. Load index values that caused the host to be busy are preceded by an asterisk (`*`).

lockW

The host is locked by its run window. Run windows for a host are specified in the configuration file (see `lsf.conf(5)`) and can be displayed by `lshosts`. A locked host does not accept load shared jobs from other hosts.

lockU

The host is locked by the LSF administrator or root.

unavail

The host is down or the LIM on the host is not running.

unlicensed

The host does not have a valid LSF license.

r15s

The 15-second exponentially averaged CPU run queue length.

r1m

The 1-minute exponentially averaged CPU run queue length.

r15m

The 15-minute exponentially averaged CPU run queue length.

ut

The CPU utilization exponentially averaged over the last minute, between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in `/tmp`, in MB.

swp

The amount of available swap space.

By default, the amount is displayed in KB. The amount may appear in MB depending on the actual system swap space. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (GB, TB, PB, or EB).

mem

The amount of available RAM.

By default, the amount is displayed in KB. The amount may appear in MB depending on the actual system memory. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (GB, TB, PB, or EB).

io

By default, `io` is not shown.

If -l is specified, shows the disk I/O rate exponentially averaged over the last minute, in KB per second.

external_index

By default, external load indices are not shown.

If -l is specified, shows indices for all dynamic custom resources available on the host, including shared, string and Boolean resources.

If -I *load_index* is specified, only shows indices for specified non-shared (host-based) dynamic numeric custom resources.

Resource-Based Output (lsload -s)

Displays information about dynamic resources (shared or host-based). Each line gives the value and the associated hosts for an instance of the resource. See lim(8), and `lsf.cluster(5)` for information on configuring dynamic shared resources.

The displayed information consists of the following fields:

RESOURCE

Name of the resource.

VALUE

Value for an instance of the resource.

LOCATION

Hosts associated with the instance of the resource.

Examples

lsload -R "select[r1m<=0.5 && swp>=20 && type==ALPHA]"

OR, in restricted format:

lsload -R r1m=0.5:swp=20:type=ALPHA

Displays the load of ALPHA hosts with at least 20 MB of swap space, and a 1-minute run queue length less than 0.5.

lsload -R "select[(1-swp/maxswp)<0.75] order[pg]"

Displays the load of the hosts whose swap space utilization is less than 75%. The resulting hosts are ordered by paging rate.

lsload -l r1m:ut:io:pg

Displays the 1-minute CPU raw run queue length, the CPU utilization, the disk I/O and paging rates for all hosts in the cluster.

lsload -E

Displays the load of all hosts, ordered by `r15s:pg`, with the CPU run queue lengths being the effective run queue lengths.

lsload -s verilog_license

Displays the value and location of all the `verilog_license` dynamic shared resource instances.

Diagnostics

Exit status is -10 for LSF problems or a bad resource names.

Exit status is -1 if a bad parameter is specified, otherwise `lsload` returns 0.

See also

lim(8), `lsf.cluster`(5), lsplace(1), lshosts(1), lsinfo(1), `lslockhost`(8), `ls_load`(3)

lsloadadj

adjusts load indices on hosts

Synopsis

lsloadadj [**-R** *res_req*] [*host_name*[**:***num_task*] ...]

lsloadadj [**-h** | **-V**]

Description

Adjusts load indices on hosts. This is useful if a task placement decision is made outside LIM by another application.

By default, assumes tasks are CPU-intensive and memory-intensive. This means the CPU and memory load indices are adjusted to a higher number than other load indices.

By default, adjusts load indices on the local host, the host from which the command was submitted.

By default, starts 1 task.

Upon receiving a load adjustment request, LIM temporarily increases the load on hosts according to resource requirements. This helps LIM avoid sending too many jobs to the same host in quick succession. The adjusted load decays over time before the real load produced by the dispatched task is reflected in LIM's load information.

`lsloadadj` adjusts all indices except for `ls` (login sessions), `it` (idle time), `r15m` (15-minute run queue length) and external load indices. Other load indices can only be adjusted beyond specific maximum values.

- tmp is -0.5
- swp is -1.5
- mem is -1.0
- r1m is 0.4
- ut is 15%
- r15s is 0.1
- pg is 0.3

Options

-R *res_req*

Specify resource requirements for tasks. Only the resource usage (rusage) section of the resource requirement string is considered. This is used by LIM to determine by how much individual load indices are to be adjusted.

For example, if a task is swap-space-intensive, load adjustment on the `swp` load index is higher; other indices are increased only slightly.

***host_name*[:*num_task*] ...**

Specify a list of hosts for which load is to be adjusted. num_task indicates the number of tasks to be started on the host.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

lsloadadj -R "rusage[swp=20:mem=10]"

Adjusts the load indices `swp` and `mem` on the host from which the command was submitted.

Diagnostics

Returns -1 if a bad parameter is specified; otherwise returns 0.

See also

lsinfo(1), lsplace(1), lsload(1), `ls_loadadj(3)`

lslogin

remotely logs in to a lightly loaded host

Synopsis

lslogin [**-v**] [**-m "***host_name ...***"** | **-m "***cluster_name ...***"**] [**-R "***res_req***"**] [*rlogin_options*]

lslogin [**-h** | **-V**]

Description

Remotely logs in to a lightly loaded host.

By default, `lslogin` selects the least loaded host, with few users logged in, and remotely logs in to that host using the UNIX `rlogin` command.

In a MultiCluster environment, the default is to select the least loaded host in the local cluster.

As an alternative to `rlogin`, you can use an SSH connection by enabling LSF_LSLOGIN_SSH in `lsf.conf`.

Options

-v

Displays the name of the host to which `lslogin` remotely logs you in.

-m "*host_name ...*" | -m "*cluster_name ...*"

Remotely logs in to the specified host.

With MultiCluster job forwarding, when a cluster name is specified, remotely logs in to the least loaded host in the specified cluster, if the remote cluster accepts interactive jobs from the local cluster (see `lsf.cluster(5)`).

-R "*res_req*"

Remotely logs in to a host that meets the specified resource requirement. The resource requirement expression restricts the set of candidate hosts and determines the host selection policy.

For a complete explanation of resource requirement expressions, see *Administering Platform LSF*. To find out what resources are configured in your system, use `lsinfo` and `lshosts`.

rlogin_options

Specify remote login options passed to the `rlogin` command.

If remote execution fails, `lslogin` logs in locally only if the local host also satisfies required resources; otherwise, log in fails.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Example

lslogin -R "select[it>1 && bsd]"

Remotely logs in to a host that has been idle for at least 1 minute, runs BSD UNIX, and is lightly loaded both in CPU resources and the number of users logged in.

Diagnostics

Because `lslogin` passes all unrecognized arguments to `rlogin`, incorrect options usually cause the `rlogin` usage message to be displayed rather than the `lslogin` usage message.

See also

`ls_placereq`, `rlogin`

lsltasks

displays or updates a local task list

Synopsis

lsltasks [**+** *task_name* ... | **–** *task_name* ...]

lsltasks [**-h** | **-V**]

Description

Displays or updates a user local task list in `$HOME/.lsftask`.

When no options are specified, displays tasks listed in the system task file `lsf.task` and the user task file `.lsftask`.

If there is a conflict between the system task file `lsf.task` and the user task file (`.lsftask`), the user task file overrides the system task file.

Tasks in the local task list are not eligible for remote execution, either because they are trivial tasks or because they need resources on the local host.

Options

+ *task_name*

If + is specified and the specified task names are not already in the user task file (`.lsftask`), adds the task names to the file with a plus sign (+) preceding them.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a + or nothing, replaces the entry with the specified content; if the entry starts with a minus sign (-), deletes the entry from the `.lsftask` file.

- *task_name*

If – is specified and specified task names are not already in the user `.lsftask` file, adds the task names to the file with a – preceding the task name.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a –, no operation is done; if the entry starts with a +, deletes the entry from the `.lsftask` file.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

lsltasks + foo

Adds the command `foo` to the local task list.

Files

Reads the system task file `lsf.task`, and the user `.lsftask` file. See `lsf.task`(5) for more details.

The system and user task files contain two sections, one for the remote task list, the other for the local task list. The local tasks section starts with `Begin LocalTasks` and ends with `End LocalTasks`. Each line in the section is an entry consisting of a task name.

A plus sign (+) or a minus sign (–) can optionally precede each entry. If no + or – is specified, then + is assumed.

See also

lseligible, `ls_task`, lsrtasks, `lsf.task`, `ls_eligible`

lsmon

displays load information for LSF hosts and periodically updates the display

Synopsis

lsmon [**-N** | **-E**] [**-n** *num_hosts*] [**-R** *res_req*] [**-I** *index_list*] [**-i** *interval*] [**-L** *file_name*] [*host_name* ...]

lsmon [**-h** | **-V**]

Description

`lsmon` is a full-screen LSF monitoring utility that displays and updates load information for hosts in a cluster.

By default, displays load information for all hosts in the cluster, up to the number of lines that fit on-screen.

By default, displays raw load indices.

By default, load information is sorted according to CPU and paging load.

By default, load information is updated every 10 seconds.

Options

-N

Displays normalized CPU run queue length load indices.

-E

Displays effective CPU run queue length load indices. Options -N and -E are mutually exclusive.

-n *num_hosts*

Displays only load information for the requested number of hosts. Information for up to num_hosts hosts that best satisfy resource requirements is displayed.

-R *res_req*

Displays only load information for hosts that satisfy the specified resource requirements. See *Administering Platform LSF* for a list of built-in resource names.

Load information for the hosts is sorted according to load on the specified resources.

If res_req contains special resource names, only load information for hosts that provide these resources is displayed (use `lshosts` to find out what resources are available on each host).

If one or more host names are specified, only load information for the hosts that satisfy the resource requirements is displayed.

-I *index_list*

Displays only load information for the specified load indices. Load index names must be separated by a colon (for example, `r1m:pg:ut`).

If the index list *index_list* is too long to fit in the screen of the user who invoked the command, the output is truncated. For example, if the invoker's screen is 80 characters wide, then up to 10 load indices are displayed.

-i *interval*

Sets how often load information is updated on-screen, in seconds.

-L *file_name*

Saves load information in the specified file while it is displayed on-screen.

If you do not want load information to be displayed on your screen at the same time, use lsmon -L *file_name* < /dev/null. The format of the file is described in `lim.acct` (5).

***host_name* ...**

Displays only load information for the specified hosts.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Usage

You can use the following commands while `lsmon` is running:

[^L | i | n | N | E | R | q]

^L

Refreshes the screen.

i

Prompts you to input a new update interval.

n

Prompts you to input a new number of hosts to display.

N

Toggles between displaying raw CPU run queue length load indices and normalized CPU run queue length load indices.

E

Toggles between displaying raw CPU run queue length load indices and effective CPU run queue length load indices.

R

Prompts you to input new resource requirements.

q

Quits `lsmon`.

Output

The following fields are displayed by default.

HOST_NAME

Name of specified hosts for which load information is displayed, or if resource requirements were specified, name of hosts that satisfied the specified resource requirement and for which load information is displayed.

status

Status of the host. A minus sign (-) may precede the status, indicating that the Remote Execution Server (RES) on the host is not running.

Possible statuses are:

ok

The host is in normal load sharing state and can accept remote jobs.

busy

The host is overloaded because some load indices exceed configured thresholds. Load index values that caused the host to be busy are preceded by an asterisk (`*`). Built-in load indices include `r15s`, `r1m`, `r15m`, `ut`, `pg`, `io`, `ls`, `it`, `swp`, `mem` and `tmp` (see below). External load indices are configured in the file `lsf.cluster.`*cluster_name*.

lockW

The host is locked by its run window. Run windows for a host are specified in `lsf.conf` and can be displayed by `lshosts`. A locked host does not accept load shared jobs from other hosts.

lockU

The host is locked by the LSF administrator or root.

unavail

The host is down or the Load Information Manager (LIM) on the host is not running.

unlicensed

The host does not have a valid LSF license.

r15s

The 15-second exponentially averaged CPU run queue length.

r1m

The 1-minute exponentially averaged CPU run queue length.

r15m

The 15-minute exponentially averaged CPU run queue length.

ut

The CPU utilization exponentially averaged over the last minute, between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in `/tmp`, in megabytes.

swp

The amount of currently available swap space, in megabytes.

mem

The amount of currently available memory, in megabytes.

Diagnostics

Specifying an incorrect resource requirement string while `lsmon` is running (via the R option) causes `lsmon` to exit with an appropriate error message.

`lsmon` exits if it does not receive a reply from LIM within the update interval.

See also

lshosts, lsinfo, lsload, `lslockhost`, `lim.acct`, `ls_load`

lspasswd

registers user passwords in LSF on Windows

Synopsis

lspasswd [**-u** *user_name*] [**-p** *password*]

lspasswd [**-r**] [**-u** *user_name*]

lspasswd [**-c**] [**-u** *user_name*] [**-p** *password*]

lspassword [**-h** | **-V**]

Description

Registers user passwords in LSF on Windows. Passwords must be longer than 3 characters and 31 characters or less.

By default, if no options are specified, the password applies to the user who issued the command.

Users must update the password maintained by LSF if they change their Windows user account password.

Passwords are Windows user account passwords and are saved in the LSF database. LSF uses the passwords to start jobs on behalf of the user.

`lspasswd` communicates with LSF services to store the password. The password is stored in encrypted format and the password database is protected by file access permissions. The password remains encrypted as it travels through the network.

The -p option allows scripts to use `lspasswd`. You should not use this option directly on the command line because the password is entered in full view on the command line. Only error messages are displayed when using the -p option.

Options

-c -u *user_name* -p *password*

Check that the password is valid for the specified user.

-r -u *user_name*

Remove the user entry from the password database.

-u *user_name* -p *password*

Specify the user and password for the user whose password you want to register or change. Include the domain name in uppercase letters (*DOMAIN_NAME* *user_name*).

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Limitations

You must run `lspasswd` from an LSF Windows host.

lsplace

displays hosts available to execute tasks

Synopsis

lsplace [**-L**] [**-n** *minimum* | **-n 0**] [**-R** *res_req*] [**-w** *maximum* | **-w 0**] [*host_name* ...]

lsplace [**-h** | **-V**]

Description

Displays hosts available for the execution of tasks, and temporarily increases the load on these hosts (to avoid sending too many jobs to the same host in quick succession). The inflated load decays slowly over time before the real load produced by the dispatched task is reflected in the LIM's load information. Host names may be duplicated for multiprocessor hosts, to indicate that multiple tasks can be placed on a single host.

By default, displays only one host name.

By default, uses LSF default resource requirements.

Options

-L

Attempts to place tasks on as few hosts as possible. This is useful for distributed parallel applications to minimize communication costs between tasks.

-n *minimum* | -n 0

Displays at least the specified number of hosts. Specify 0 to display as many hosts as possible.

Prints `Not enough host(s) currently eligible` and exits with status 1 if the required number of hosts holding the required resources cannot be found.

-R *res_req*

Displays only hosts with the specified resource requirements. When LSF_STRICT_RESREQ=Y in `lsf.conf`, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

-w *maximum* | -w 0

Displays no more than the specified number of hosts. Specify 0 to display as many hosts as possible.

***host_name* ...**

Displays only hosts that are among the specified hosts.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

`lsplace` is mostly used in backquotes to pick out a host name that is then passed to other commands. The following example issues a command to display a lightly loaded HPPA-RISC host for your program to run on:

```
lsrun -m ‘lsplace -R hppa‘ myprogram
```

In order for a job to land on a host with an exclusive resource, you need to explicitly specify that resource for the resource requirements. The following example issues a command to display the host with the bigmem exclusive resource for your program to run on:

```
lsrun -m ‘lsplace -R "bigmem"‘ myprogram
```

The -w and -n options can be combined to specify the upper and lower bounds in processors to be returned, respectively. For example, the command

```
lsplace -n 3 -w 5
```

returns at least 3 and not more than 5 host names.

Diagnostics

`lsplace` returns 1 if insufficient hosts are available. The exit status is -10 if a problem is detected in LSF, -1 for other errors, otherwise 0.

See also

lsinfo(1), `ls_placereq`(3), lsload(1), lsrun(1)

lsrcp

remotely copies files using LSF

Synopsis

lsrcp [**-a**] *source_file target_file*

lsrcp [**-h** | **-V**]

Description

Remotely copies files using LSF.

`lsrcp` is an LSF-enabled remote copy program that transfers a single file between hosts in an LSF cluster. `lsrcp` uses RES on an LSF host to transfer files. If LSF is not installed on a host or if RES is not running then `lsrcp` uses `rcp` to copy the file.

To use `lsrcp`, you must have read access to the file being copied.

Both the source and target file must be owned by the user who issues the command.

`lsrcp` uses `rcp` to copy a source file to a target file owned by another user. See `rcp(1)` and LIMITATIONS below for details.

Options

-a

Appends source_file to target_file.

source_file target_file

Specify an existing file on a local or remote host that you want to copy, and a file to which you want to copy the source file.

File format is as follows:

[[*user_name*@][*host_name*:][*path*/]*file_name*

user_name

Login name to be used for accessing files on the remote host. If *user_name* is not specified, the name of the user who issued the command is used.

host_name

Name of the remote host on which the file resides. If *host_name* is not specified, the local host, the host from which the command was issued, is used.

path

Absolute path name or a path name relative to the login directory of the user. Shell file name expansion is not supported on either the local or remote hosts. Only single files can be copied from one host to another.

Use "\" to transfer files from a Windows host to another Windows host. For example:

```
c:\share>lsrcp file1 hostA:c:\temp\file2
```

Use "/" to transfer files from a UNIX host to a UNIX host. For example:

lsrcp file1 hostD:/home/usr2/test/file2

Always use "/" to transfer files from a UNIX host to a Windows host, or from a Windows host to a UNIX host. This is because the operating system interprets "\" and `lsrcp` opens the wrong files.

For example, to transfer a file from UNIX to a Windows host:

```
lsrcp file1 hostA:c:/temp/file2
```

To transfer a file from Windows to a UNIX host:

```
c:\share>lsrcp file1 hostD:/home/usr2/test/file2
```

file_name

Name of source file. File name expansion is not supported.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

lsrcp myfile @hostC:/home/usr/dir1/otherfile

Copies file `myfile` from the local host to file `otherfile` on `hostC`.

lsrcp user1@hostA:/home/myfile user1@hostB:otherfile

Copies the file `myfile` from `hostA` to file `otherfile` on `hostB`.

lsrcp -a user1@hostD:/home/myfile /dir1/otherfile

Appends the file `myfile` on `hostD` to the file `otherfile` on the local host.

lsrcp /tmp/myfile user1@hostF:~/otherfile

Copies the file `myfile` from the local host to file `otherfile` on `hostF` in `user1`'s home directory.

Diagnostics

`lsrcp` attempts to copy source_file to target_file using RES. If RES is down or fails to copy the source_file, `lsrcp` uses either `rsh` or the shell command specified by LSF_RSH in `lsf.conf` when the -a option is specified. When -a is not specified, `lsrcp` uses `rcp`.

Limitations

File transfer using `lsrcp` is not supported in the following contexts:

- If LSF account mapping is used; `lsrcp` fails when running under a different user account
- On LSF client hosts. LSF client hosts do not run RES, so `lsrcp` cannot contact RES on the submission host
- Third party copies. `lsrcp` does not support third party copies, when neither source nor target file are on the local host. In such a case, `rcp` or `rsh` (or the shell command specified by LSF_RSH in `lsf.conf`) is used. If the target_file

exists, `lsrcp` preserves the modes; otherwise, `lsrcp` uses the source_file modes modified with the `umask` (see `umask(2)`) of the source host.

You can do the following:

- `rcp` on UNIX. If `lsrcp` cannot contact RES on the submission host, it attempts to use `rcp` to copy the file. You must set up the `/etc/hosts.equiv` or `HOME/.rhosts` file to use `rcp`. See the `rcp`(1), `rsh`(1), `ssh`(1) manual pages for more information on using the `rcp`, `rsh`, and `ssh` commands.
- You can replace `lsrcp` with your own file transfer mechanism as long as it supports the same syntax as `lsrcp`. This might be done to take advantage of a faster interconnection network, or to overcome limitations with the existing `lsrcp`. sbatchd looks for the `lsrcp` executable in the LSF_BINDIR directory.

See also

`rsh`, `rcp`, `res`

lsrtasks

displays or updates a remote task list

Synopsis

lsrtasks [**+** *task_name*[/*res_req*] ... | **–** *task_name*[/*res_req*] ...]

lsrtasks [**-h** | **-V**]

Description

Displays or updates a user's remote task list in `$HOME/.lsftask`.

When no options are specified, displays tasks listed in the system task file `lsf.task` and the user's task file (`.lsftask`).

If there is a conflict between the system task file `lsf.task` and the user task file, the user task file overrides the system task file.

Tasks in the remote task list are eligible for remote execution. You can associate resource requirements with each task name. Eligibility of tasks not specified in a task list for remote execution depends on the operation mode: local or remote. In local mode, tasks are not eligible for remote execution; in remote mode, tasks are eligible. You can specify the operation mode when deciding the eligibility of a task (see `lseligible(1)`, and `ls_eligible(3)`).

Options

+ *task_name*[/*res_req*] ...

If plus sign (+) is specified and the specified task names are not already in the user task file (`.lsftask`), adds the task names to the file with a + sign preceding them.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a + or nothing, replaces the entry with the specified content; if the entry starts with a minus sign (–), deletes the entry from the `.lsftask` file.

Remote tasks can have a resource requirement expression associated with them, separated by a backslash (/). See `ls_task(3)`.

- *task_name*[/*res_req*] ...

If – is specified and specified task names are not already in the user task file (`.lsftask`), adds the task names to the file with a – preceding the task name.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a –, no operation is done; if the entry starts with a +, deletes the entry from the `.lsftask` file.

Remote tasks can have a resource requirement expression associated with them, separated by a backslash /. See `ls_task`(3).

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

```
% lsrtasks + task1 task2/"select[cpu && mem]" - task3
```

or in restricted form:

% lsrtasks + task1 task2/cpu:mem - task3

Adds the command `task1` to the remote task list with no resource requirements, adds `task2` with the resource requirement `cpu:mem`, and removes `task3` from the remote task list.

% **lsrtasks + myjob/swap>=100 && cpu**

Adds `myjob` to the remote tasks list with its resource requirements.

Running `lsrtasks` with no arguments displays the resource requirements of tasks in the remote list, separated from the task name by a slash (/):

```
% lsrtasks
cc/cpu              cfd3d/type == SG1 && cpu  compressdir/cpu:mem
f77/cpu             verilog/cpu && cadence    compress/cpu
dsim/type == any    hspice/cpu && cadence      nas/swp > 200 && cpu
compress/-:cpu:mem  epi/hpux11 sparc           regression/cpu
cc/type == local    synopsys/swp >150 && cpu
```

Files

Reads the system task file `lsf.task`, and the user task file (`.lsftask`). See `lsf.task`(5) for more details.

The system and user task files contain two sections, one for the remote task list, the other for the local task list. The remote tasks section starts with `Begin RemoteTasks` and ends with `End RemoteTasks`. Each line in the section is an entry consisting of a task name.

A plus sign + or a minus sign – can optionally precede each entry. If no + or – is specified, then + is assumed.

See also

lseligible, `ls_task`, lsltasks, `lsf.task`, `ls_eligible`

lsrun

runs an interactive task through LSF

Synopsis

lsrun [**-l**] [**-L**] [**-P**] [**-S**] [**-v**] [**-m "***host_name ...***"** | **-m "***cluster_name ...***"**] [**-R "***res_req***"**] *command* [*argument ...*]

lsrun [**-h** | **-V**]

Description

Submits a task to LSF for execution.

With MultiCluster job forwarding model, the default is to run the task on a host in the local cluster.

By default, `lsrun` first tries to obtain resource requirement information from the remote task list to find an eligible host. (See `lseligible`(1) and `ls_task`(3).) Otherwise, `lsrun` runs the task on a host that is of the same host type (or architecture) as the submission host. If several hosts of the same architecture are available, the host with the lowest CPU and memory load is selected.

By default, if execution fails and the local host satisfies resource requirements, LSF runs the task locally.

By default, `lsrun` does not create a pseudo-terminal when running the task.

Options

-l

If execution on another host fails, runs the task locally.

-L

Forces `lsrun` to go through RES to execute a task. By default, `lsrun` does not use RES if the task is going to run on the current host.

-P

Creates a pseudo-terminal when starting the task on UNIX hosts. This is necessary to run programs that require a pseudo-terminal (for example, `vi`).

This option is not supported on Windows.

-S

Creates a pseudo-terminal with shell mode support when starting the task on a UNIX host. Shell mode support is required for running interactive shells or applications that redefine the `CTRL-C` and `CTRL-Z` keys (for example, `jove`).

This option is not supported on Windows.

-v

Displays the name of the host running the task.

-m "*host_name* ..." | -m "*cluster_name* ..."

The execution host must be one of the specified hosts. If a single host is specified, all resource requirements are ignored.

If multiple hosts are specified and you do not use the -R option, the execution host must satisfy the resource requirements in the remote task list (see `lsrtasks`(1)). If none of the specified hosts satisfy the resource requirements, the task does not run.

With MultiCluster job forwarding model, the execution host can be a host in one of the specified clusters, if the remote cluster accepts tasks from the local cluster. (See `RemoteClusters` section in `lsf.cluster`(5).)

-R "*res_req*"

Runs the task on a host that meets the specified resource requirement. The size of the resource requirement string is limited to 512 bytes. For a complete explanation of resource requirement expressions, see *Administering Platform LSF*. To find out what resources are configured in your system, use `lsinfo` and `lshosts`.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

Exclusive resources need to be explicitly specified within the resource requirement string. For example, you defined a resource called `bigmem` in `lsf.shared` and defined it as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command to submit a task to run on `hostE`:

lsrun -R "bigmem" myjob

or

lsrun -R "defined(bigmem)" myjob

If the -m option is specified with a single host name, the -R option is ignored.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Usage

You can use `lsrun` together with other utility commands such as `lsplace`, `lsload`, `lsloadadj`, and `lseligible` to write load sharing applications in the form of UNIX shell scripts.

`lsrun` supports interactive job control. Suspending `lsrun` suspends both the task and `lsrun`, and continuing `lsrun` continues the task.

If LSB_DISABLE_LIMLOCK_EXCL=y (to enable preemption of exclusive jobs, for example), you can use `lsrun` to start a task on a host that is currently running an exclusive job.

The -n option of `rsh` can be simulated by redirecting input from `/dev/null`. For example:

```
lsrun cat </dev/null &
```

Diagnostics

`lsrun` exits with status -10 and prints an error message to `stderr` if a problem is detected in LSF and the task is not run.

The exit status is -1 and an error message is printed to `stderr` if a system call fails or incorrect arguments are specified.

Otherwise, the exit status is the exit status of the task.

See also

`rsh`, `ls_rexecv`, lsplace, lseligible, lsload, lshosts, lsrtasks, `lsf.cluster`

lstcsh

load sharing `tcsh` for LSF

Synopsis

lstcsh [*tcsh_options*] [**-L**] [*argument ...*]

Description

`lstcsh` is an enhanced version of `tcsh`. `lstcsh` behaves exactly like `tcsh`, except that it includes a load sharing capability with transparent remote job execution for LSF.

By default, a `lstcsh` script is executed as a normal `tcsh` script with load sharing disabled.

If a command line is considered eligible for remote execution, LSF selects a suitable host— typically a powerful and/or lightly loaded host that can execute the command line correctly—and sends the command line to that host.

You can restrict who can use `@` for host redirection in `lstcsh` with the parameter LSF_SHELL_AT_USERS in `lsf.conf`.

Remote Hosts

`lstcsh` provides a high degree of network transparency. Command lines executed on remote hosts behave the same as they do on the local host. The remote execution environment is designed to mirror the local one as closely as possible by using the same values for environment variables, terminal setup, current working directory, file creation mask, and so on. Each modification to the local set of environment variables is automatically reflected on remote hosts.

Shell variables, nice values, and resource limits are not automatically propagated to remote hosts.

Job Control

Job control in `lstcsh` is exactly the same as in `tcsh` except for remote background jobs. `lstcsh` numbers background jobs separately for each of the hosts that are used to execute them. The output of the built-in command `job` lists background jobs together with their execution hosts.

To bring a remote background job to the foreground, the host name must be specified together with an at sign (@), as in the following example:

```
fg %2 @hostA
```

Similarly, the host name must be specified when killing a remote job. For example:

```
kill %2 @hostA
```

Options

tcsh_options

`lstcsh` accepts all the options used by `tcsh`. See `tcsh(1)` for the meaning of specific options.

-L

Executes a script with load sharing enabled.

There are three ways to run a `lstcsh` script with load sharing enabled:

- Execute the script with the -L option

- Use the built-in command `source` to execute the script

- Insert "`#!/local/bin/lstcsh -L`" as the first line of the script (assuming you install `lstcsh` in `/local/bin`).

Using `@` or `lsmode` in a script does not enable load sharing if the script has not been executed using one of these three ways.

Usage

In addition to the built-in commands in `tcsh`, `lstcsh` provides the following built-in commands:

lsmode [**on** | **off**] [**local** | **remote**] [@] [**v** | **-v**] [**e** | **-e**] [**t** | **-t**] [**connect** [*host_name* ...]] [**lsrtasks** [*lsrtasks_options*]] [**lsltasks** [*lsltasks_options*]] [**jobs**]

on | off

Turns load sharing on or off. When off, you can specify `@` to send a command line to a remote host.

local | remote

Sets operation mode of `lstcsh`.

The default is local.

local

Local operation mode. This is the default mode.

In this mode, a command line is eligible for remote execution only if all the specified tasks are present in the remote task list in the user's tasks file `$HOME/.lsftask`, or if `@` is specified on the command line to force specified tasks to be eligible for remote execution.

Tasks in the local task list must be executed locally.

The local mode of operation is conservative, and can fail to take advantage of the performance benefits and load balancing advantages of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

remote

Remote operation mode.

In this mode, a command line is considered eligible for remote execution only if none of the specified tasks are present in the local task list in the user's tasks file `$HOME/.lsftask.`

Tasks in the remote list can be executed remotely.

The remote mode of operation is aggressive, and promotes extensive use of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

@

Specify @ to explicitly specify the eligibility of a command for remote execution.

The @ may be anywhere in the command line except in the first position (which is used to set the value of shell variables).

There are several ways to use @:

@

Specify @ followed by nothing to indicate the command line is eligible for remote execution.

@ host_name

Specify @ followed by a host name to force the command line to be executed on that host.

Host names and the reserved word local following @ can all be abbreviated as long as they do not cause ambiguity.

@ local

Specify @ followed by the reserved word local to force the command line to executed on the local host.

@ /res_req

Specify @ followed by / and a resource requirement string to indicate the command is eligible for remote execution, and that the specified resource requirements must be used instead of those in the remote task list.

When specifying resource requirements following the @ it is necessary to use / only if the first requirement characters specified are also the first characters of a host name.

e | -e

Turns eligibility verbose mode on (e) or off (-e).

If eligibility verbose mode is on, `lstcsh` shows whether the command is eligible for remote execution, and displays the resource requirement used if the command is eligible.

The default is off.

v | -v

Turns task placement verbose mode on (v) or off (-v). If verbose mode is on, `lstcsh` displays the name of the host on which the command is run if the command is not run on the local host.

The default is on.

t | -t

Turns wall clock timing on (t) or off (-t).

If timing is on, the actual response time of the command is displayed. This is the total elapsed time in seconds from the time you submit the command to the time the prompt comes back.

This time includes all remote execution overhead. The `csh` time built-in does not include the remote execution overhead.

This is an impartial way of comparing the response time of jobs submitted locally or remotely, because all the load sharing overhead is included in the displayed elapsed time.

The default is off.

connect [*host_name* ...]

Establishes connections with specified remote hosts. If no hosts are specified, lists all the remote hosts to which an `lstcsh` connection has been established.

A plus sign (+) with a remote host indicates that a server-shell has also been started on it.

lsrtasks [+ *task_name*[*/res_req* ...] | - *task_name*[*/res_req* ...]]

Displays or update a user's remote task list in the user's task list `$HOME/.lsftask`.

This command has the same function as the external command `lsrtasks`, except that the modified remote task list takes effect immediately for the current `lstcsh` session.

See `lsrtasks(1)` for more details.

lsltasks [+ *task_name* ... | - *task_name* ...]

Displays or update a user's local task list in the user's task list `$HOME/.lsftask`.

This command has the same function as the external command `lsltasks`, except that the modified local task list takes effect immediately for the current `lstcsh` session.

See `lsltasks`(1) for more details.

jobs

Lists background jobs together with the execution hosts. This break of transparency is intentional to provide you with more control over your background jobs.

Files

There are three optional configuration files for `lstcsh`:

`.shrc`

`.hostrc`

`.lsftask`

The `.shrc` and `.hostrc` files are used by `lstcsh` alone, whereas `.lsftask` is used by LSF to determine general task eligibility.

~/.shrc

Use this file when you want an execution environment on remote hosts that is different from that on the local host. This file is sourced automatically on a remote host when a connection is established. For example, if the remote host is of different type, you may need to run a version of the executable for that particular host type, therefore it may be necessary to set a different path on the remote host.

~/.hostrc

Use this file to indicate a list of host names to which the user wants to be connected (asynchronously in the background) at `lstcsh` startup time. This saves the time spent in establishing the connections dynamically during execution of shell commands. Once a connection is set up, you can execute further remote commands on those connected hosts with very little overhead.

~/.lsftask

Use this file to specify lists of remote and local tasks that you want to be added to the respective system default lists. Each line of this file is of the form task_name/res_req, where task_name is the name of a task, and res_req is a string specifying the resource requirements of the task. If res_req is not specified, the command is executed on machines of the same type as the local host.

Limitations

Type-ahead for the `next` command is discarded when a job is executing in the foreground on a remote host.

It is not possible to provide input data to load sharing shell scripts (that is, shell scripts whose content is load shared).

The `lstcsh` is fully compatible with tcsh 6.03 7-bit mode. Any feature that is not included in `tcsh` 6.03 is not supported.

See also

`csh`, `tcsh`, lsrtasks, lsltasks, lseligible, lsinfo, lsload

pam

Parallel Application Manager – job starter for MPI applications

HP-UX vendor MPI syntax

bsub pam -mpi mpirun [*mpirun_options*] *mpi_app* [*argument* ...]

SGI vendor MPI syntax

bsub pam [**-n** *num_tasks*] **-mpi -auto_place** *mpi_app* [*argument* ...]

Generic PJL framework syntax

bsub pam [**-t**] [**-v**] [**-n** *num_tasks*] **-g** [*num_args*] *pjl_wrapper* [*pjl_options*] *mpi_app* [*argument* ...] **pam** [**-h**] [**-V**]

Description

The Parallel Application Manager (PAM) is the point of control for Platform LSF. PAM is fully integrated with Platform LSF to interface the user application with LSF. PAM acts as the supervisor of a parallel LSF job.

MPI jobs started by `pam` can only be submitted through the LSF Batch system. PAM cannot be used interactively to start parallel jobs. sbatchd starts PAM on the first execution host.

For all parallel application processes (tasks), PAM:

- Uses a vendor MPI library or an MPI Parallel Job Launcher (PJL); for example, `mpirun`, `poe` start a parallel job on a specified set of hosts in an LSF cluster.
- PAM contacts RES on each execution host allocated to the parallel job.
- PAM queries RES periodically to collect resource usage for each parallel task and passes control signals through RES to all process groups and individual running tasks, and cleans up tasks as needed.
- Passes job-level resource usage and process IDs (PIDs and PGIDs) to sbatchd for enforcement
- Collects resource usage information and exit status upon termination

Task startup for vendor MPI jobs

The `pam` command starts a vendor MPI job on a specified set of hosts in a LSF cluster. Using `pam` to start an MPI job requires the underlying MPI system to be LSF aware, using a vendor MPI implementation that supports LSF (SGI IRIX vendor MPI or HP-UX vendor MPI).

PAM uses the vendor MPI library to spawn the child processes needed for the parallel tasks that make up your MPI application. It starts these tasks on the systems allocated by LSF. The allocation includes the number of execution hosts needed, and the number of child processes needed on each host.

Task startup for LSF HPC generic PJL jobs

For parallel jobs submitted with `bsub`:

- PAM invokes the PJL, which in turn invokes the TaskStarter (TS).
- TS starts the tasks on each execution host, reports the process ID to PAM, and waits for the task to finish.

Two environment variables allow you to run scripts or binaries before or after PAM is invoked. These are useful if you customize `mpirun.lsf` and have job scripts that call `mpirun.lsf` more than once.

- $MPIRUN_LSF_PRE_EXEC: Runs before PAM is invoked.
- $MPIRUN_LSF_POST_EXEC: Runs after PAM is invoked.

Options for vendor MPI jobs

-auto_place

The -auto_place option on the `pam` command line tells the SGI IRIX `mpirun` library to launch the MPI application according to the resources allocated by LSF.

-mpi

In the SGI environment, the -mpi option on the `bsub` and `pam` command line is equivalent to the `mpirun` command.

On HP-UX, you can have LSF manage the allocation of hosts to achieve better resource utilization by coordinating the start-up phase with `mpirun`. This is done by preceding the regular HP MPI `mpirun` command with:

bsub pam -mpi

For HP-UX vendor MPI jobs, the -mpi option must be the first option of the `pam` command.

For example, to run a single-host job and have LSF select the host, the command:

mpirun -np 14 a.out

is entered as:

bsub pam -mpi mpirun -np 14 a.out

-n *num_tasks*

The number of processors required to run the parallel application, typically the same as the number of parallel tasks in the job. If the host is a multiprocessor, one host can start several tasks.

You can use both `bsub -n` and `pam -n` in the same job submission. The number specified in the `pam -n` option should be less than or equal to the number specified by `bsub -n`. If the number of tasks specified with `pam -n` is greater than the number specified by `bsub -n`, the `pam -n` is ignored.

For example, on SGI IRIX or SGI Altix, you can specify:

bsub -n 5 pam -n 2 -mpi -auto_place a.out

Here, the job requests 5 processors, but PAM only starts 2 parallel tasks.

***mpi_app* [*argument* ...]**

The name of the MPI application to be run on the listed hosts. This must be the last argument on the command line.

-h

Prints command usage to `stderr` and exit.

-V

Prints LSF release version to `stderr` and exit.

Options for LSF HPC generic PJL jobs

-t

This option tells `pam` not to print out the MPI job tasks summary report to the standard output. By default, the summary report prints out the task ID, the host on which it was executed, the command that was executed, the exit status, and the termination time.

-v

Verbose mode. Displays the name of the execution host or hosts.

-g [*num_args*] *pjl_wrapper* [*pjl_options*]

The -g option is required to use the LSF generic PJL framework. You must specify all the other `pam` options before -g.

num_args

Specifies how many space-separated arguments in the command line are related to the PJL (after that, the remaining section of the command line is assumed to be related to the binary application that launches the parallel tasks).

pjl_wrapper

The name of the PJL

pjl_options

Optional arguments to the PJL

For example:

- A PJL named `no_arg_pjl` takes no options, so *num_args*=1. The syntax is:
  ```
  pam [pam_options] -g 1 no_arg_pjl job [job_options]
  ```
- A PJL is named `3_arg_pjl` and takes the options -a, -b, and *group_name*, so *num_args*=4. The syntax is:

```
pam [pam_options] -g 4 3_arg_pjl -a -b group_name job [job_options]
```

-n *num_tasks*

The number of processors required to run the MPI application, typically the number of parallel tasks in the job. If the host is a multiprocessor, one host can start several tasks.

You can use both `bsub -n` and `pam -n` in the same job submission. The number specified in the `pam -n` option should be less than or equal to the number specified by bsub -n. If the number of tasks specified with pam -n is greater than the number specified by bsub -n, the pam -n is ignored.

***mpi_app* [*argument* ...]**

The name of the MPI application to be run on the listed hosts. This must be the last argument on the command line.

-h

Prints command usage to stderr and exit.

-V

Prints LSF release version to stderr and exit.

Exit Status

`pam` exits with the exit status of mpirun or the PJL wrapper.

See also

bsub(1)

patchinstall

UNIX only. Manage patches in a licensed Platform cluster.

Synopsis

patchinstall [**-f** *env_file*] [**--silent**] *package...*

patchinstall -c [**-f** *env_file*] [**--silent**] *package...*

patchinstall -r [**-f** *env_file*] [**--silent**] *package*

patchinstall -r [**-f** *env_file*] [**--silent**] *build_number*

patchinstall -h

Description

Permission required to run this command depends on the package contents and the original cluster installation account; you should normally log on as root, but you can patch some binaries as cluster administrator (lsfadmin).

By default, the command installs one or more packages in an existing cluster.

The cluster location is normally determined by your environment setting, so ensure your environment is set before you run this command (for example, you sourced `cshrc.lsf` or `profile.lsf`).

Specify the packages you want to install.

The installer does some checking first. If it does not find a problem, it prompts you to proceed with installation. If you confirm, it backs up the current binaries to the patch backup directory and then installs the specified packages on the cluster, updating or adding new binaries. It does not modify any existing configuration files. If there is any problem during installation of a package, it automatically rolls back to the cluster's previous state. It records the changes in the patch history directory. This additional checking can take more time than installing with `lsfinstall`.

The command can also be used to do the following:

- Check—do the checking for the packages without installing them. For more information, see the -c option.
- Roll back—remove the most recent patch and return the cluster to the previous patch level. If you want to roll back multiple versions, you must roll back one patch level at a time, in the reverse order of installation. For more information, see the -r option.

Options

-c

Check. Perform checking as if to install, but do not proceed with installation.

Specify each package you want to check. You may specify multiple packages.

Checks that the existing cluster is compatible with the patch (the same version of the product is already installed on the same binary types). Fixes and fix packs may also require that a specific enhancement pack be installed.

Checks that your user account has permission to write to the installation directory, backup directory, and history directory.

Lists existing files that will be overwritten by the patch.

Lists files that to be added by the patch.

-f *env_file*

This option should only be used if you cannot set your environment (for example, you cannot source `cshrc.lsf` or `profile.lsf`).

Specify the full path and file name of a file (such as your LSF `install.config` file) that properly defines the parameter LSF_TOP.

If you use this option, the command gets the cluster location from this file, not from the settings in your environment.

-h

Outputs command usage and exits.

-r

Rollback. You must specify the most recently installed patch. The installer checks all binary types and finds all instances where the most recently installed patch has the same build number. These packages are removed and the cluster reverts to the previous patch level.

Specify the build number of the most recent patch or specify full path to the package you used to install the most recent patch, The installer automatically checks the package to determine the build. You cannot specify any other build.

To remove multiple patches and roll back multiple versions, you must run the command multiple times and roll back one patch level at a time.

You cannot roll back if the backup files from the previous patch level are unavailable (if you deleted them from the patch backup directory).

--silent

Silent mode. Install or roll back without any interactive prompts for confirmation.

Output

Status information and prompts are displayed in your command console.

Status information is also logged to `patch.log` (when patching or rolling back the cluster) or `precheck.log` (when checking a package).

If there are any problems found when checking a package, errors are displayed in your command console and also logged to `patch.err`.

See also

- `pversions` command: displays the patch level of products installed in your cluster
- `install.config` file: describes the parameter LSF_TOP
- `patch.conf` file: defines backup and history directories

perfadmin

Administer the LSF Reports (PERF) services.

Synopsis

perfadmin start *service_name* | **all**

perfadmin stop *service_name* | **all**

perfadmin [**list** | **-h**]

Description

Caution:

This command can only be used by LSF administrators.

Starts or stops the PERF services, or shows status.

Run the command on the PERF host to control the following PERF services: loader controller (plc), job data transformer (jobdt), and data purger (purger). Run the command on the Derby database host to control the Derby database service (derbydb).

If PERF services are controlled by EGO, let the EGO service controller start and stop the PERF services.

Options

start *service_name* | all

Starts the PERF services on the local host. You must specify the service name or the keyword all. Do not run this command on a host that is not the PERF host or Derby database host, you should only run one set of services per cluster.

stop *service_name* | all

Stops the PERF services on the local host. You must specify the service name or the keyword all.

list

Lists status of PERF services. Run this command on the PERF host or Derby database host.

-h

Outputs command usage and exits.

Output

Status information and prompts are displayed in your command console.

SERVICE

The name of the PERF service.

STATUS

STARTED

Service is running.

STOPPED

Service is not running.

UNKNOWN

Service status is unknown. The local host may not be the PERF host or Derby database host.

HOSTNAME

Name of the host.

WSM_PID

Process ID of the running service.

See also

- pmcadmin command: administer the Platform Management Console (PMC)
- perfsetrc command: enable automatic startup of PERF services on a host
- perfremoverc command: disable automatic startup of PERF services on a host

perfremoverc.sh

Prevents automatic startup of the LSF Reporting (PERF) daemons on a UNIX host.

Synopsis

perfremoverc.sh

Description

This is an administrative command. You must be logged on as `root` to issue this command.

Prevents automatic startup of PERF daemons on a UNIX host when a system reboot command is issued. After this script/command is issued, PERF daemons no longer start automatically if the host gets rebooted. In such a case, you must manually start daemons after the host has started up.

Removes the file `perf` created in the system startup directory by `perfsetrc.sh`.

Run the command on the PERF host to control the following PERF services: loader controller (plc), job data transformer (jobdt), and data purger (purger). Run the command on the Derby database host to control the Derby database service (derbydb).

perfsetrc.sh

Configures automatic startup of the LSF Reporting (PERF) daemons on a UNIX host.

Synopsis

perfsetrc.sh

Description

This is an administrative command. You must be logged on as `root` to issue this command.

Configures a UNIX host to allow automatic startup of PERF daemons on the machine when a system reboot command is issued. Creates the file `perf` under the system startup directory.

For ease of administration, you should enable automatic startup. This starts PERF daemons automatically when the host restarts.

If you do not configure hosts to start automatically, PERF daemons must be started manually.

Run the command on the PERF host to control the following PERF services: loader controller (plc), job data transformer (jobdt), and data purger (purger). Run the command on the Derby database host to control the Derby database service (derbydb).

pmcadmin

Administer the Platform Management Console (PMC).

Synopsis

pmcadmin [**start** | **stop** | **list** | **addrtm** *URL* | **-h**]

Description

Caution:

This command can only be used by LSF administrators.

Always run this command on the host that runs PMC.

This command is used to administer the PMC.

Options

start

Starts the Platform Management Console on the local host.

stop

Stops the Platform Management Console on the local host.

list

Status of the PMC service on the local host.

addrtm *URL*

RTM only. Integrate RTM web pages with PMC. Specify the RTM main web page access URL. For example, specify `http://host_name/cacti`

-h

Outputs command usage and exits.

Output

Output of `pmcadmin list`.

SERVICE

WEBGUI, the name of the PMC service

STATUS

STARTED

PMC is running.

STOPPED

PMC is not running.

UNKNOWN

PMC status is unknown. The local host may not be the PMC host.

HOSTNAME

Name of the host.

WSM_PID

Process ID of the service.

PORT

Web server port.

See also

- perfadmin command: administer PERF services
- pmcsetrc command: enable automatic startup of PMC on a host
- pmcremoverc command: disable automatic startup of PMC on a host

pmcremoverc.sh

Prevents automatic startup of the Platform Management Console (PMC) on a UNIX host.

Synopsis

pmcremoverc.sh

Description

This is an administrative command. You must be logged on as `root` to issue this command.

Prevents automatic startup of PMC on a UNIX host when a system reboot command is issued. After this command is issued, PMC no longer starts automatically if the host gets rebooted. In such a case, you must manually start PMC after the host has started up.

Removes the file `pmc` created in the system startup directory by `pmcsetrc.sh`.

pmcsetrc.sh

Configures automatic startup of the Platform Management Console (PMC) on a UNIX host.

Synopsis

pmcsetrc.sh

Description

Configures a UNIX host to allow automatic startup of PMC on the machine when a system reboot command is issued. Creates the file `pmc` under the system startup directory.

This is an administrative command. You must be logged on as `root` to issue this command.

For ease of administration, you should enable automatic startup. This starts PMC automatically when the host restarts.

If you do not configure hosts to start automatically, PMC must be started manually.

pversions (UNIX)

UNIX version of the command: displays the version information for Platform products installed on UNIX hosts.

Synopsis

pversions [**-f** *env_file*] [**-p** *product_name*]

pversions [**-f** *env_file*] **-b** *build_number*

pversions [**-f** *env_file*] **-q** *file_name*

pversions -c *package_name*

pversions -h

Description

By default, displays the version and patch level of Platform products.

The cluster location is normally determined by your environment setting, so ensure your environment is set before you run this command (for example, you sourced `cshrc.lsf` or `profile.lsf`).

For each binary type, displays basic version information (package build date, build number, package installed date) and lists patches installed (package type, build number, date installed, fixes).

Optionally, the command can also be used to do the following:

- Check the contents of a package before installing it
- Show information about a specific Platform product installed
- Show information about installed packages from specific build
- Find current versions of a specific Platform file and see information for each

Options

-f *env_file*

This option should only be used if you cannot set your environment (for example, you cannot source `cshrc.lsf` or `profile.lsf`).

Specify the full path and file name of a file (such as your LSF `install.config` file) that properly defines the parameter LSF_TOP.

If you use this option, the command gets the cluster location from this file, not from the settings in your environment.

-b build_number

Specify the build number of an installed patch (you can specify the most recent full installation or patches installed after the most recent full installation).

Displays information and the contents of the build (binary type and install date, notes, fixes, and files in the package).

-c package_name

Specify the full path and file name of an uninstalled package. For this option, you do not need to set your environment because a cluster is not required.

Displays package contents (notes, fixes, and files in the patch).

-p product_name

Specify one Platform product to see information for that product only. Specify **LSF** to see information about LSF or **PMC** to see information about the Console package.

-q file_name

Specify the file name of one installed file.

For each binary type, displays basic version information and file location. If the binary has been updated after the most recent full installation, displays additional information about the most recent patch that updated the file (build number, fixes, notes, date installed)

-h

Outputs command usage and exits.

Output

Information is displayed in your command console.

Product Version Information (Default and -p)

By default, displays product information for entire cluster. With -p, displays information for specified product only.

For each Platform product, displays product name and version followed by specific information about each binary type.

For each binary type, displays basic version information (package build date, build number, package installed date) and lists any patches installed (package type, build number or fix number, date installed).

binary type

Binary type, build number of binary, and build date of the binary for the most recent full installation (a full installation is installation of any distribution that contains a complete set of new binaries. A full installation can be a new cluster, a licensed upgrade, or patching with an enhancement pack).

installed

Date the binary was installed for the most recent full installation.

patched

For each patch after the most recent full installation, displays fix number, build number, and date patch was installed. If the patch was a fix pack, multiple fixes are listed.

File Version Information (-q)

With -q, displays information for specified file only.

For each Platform product that contains the specified file, displays product name and version followed by specific information about each binary type.

For each binary type that contains the specified file, displays basic version information and file location. If the binary has been updated after the most recent full installation, displays additional information about the most recent patch that updated the file (build number, fixes, notes, date installed).

binary type

Binary type, build number of binary, and build date of the binary for the most recent full installation (a full installation is any distribution that contains a complete set of new binaries. A full installation can be a new cluster installation, a licensed version upgrade, or patching with an enhancement pack).

installed

Date the binary was installed for the most recent full installation.

file

Full path to the version of the file being used for this binary type.

last patched

For the last patch to update the file after the most recent full installation, displays build number and date patch was installed.

last patch notes

Optional. Some information provided by Platform for the last patch that updated the file.

last patch fixes

Fixes included in the last patch that updated the file.

Build Version Information (-b)

With -b, displays information for patches with the specified build number only.

For each Platform product, if the product is using binaries from the specified build, displays product name and version followed by specific information about each binary type.

For each binary type, displays the following:

binary type

Binary type, build number and build date of the patch.

installed

Date the patch was installed.

notes

Optional. Some information provided by Platform for the build.

fixes

Fixes included in the patch.

files

Files included in the patch (not shown for a full distribution such as enhancement pack). Full path to the file installed by this patch.

Package Version Information (-c)

With -c, displays version information for a specified uninstalled package.

product

Displays Platform product name and version.

binary type

Binary type, build number and build date of the patch.

notes

Optional. Some information provided by Platform for the build.

fixes

Fixes included in the patch.

files

Files included in the patch (not shown for a full distribution such as enhancement pack). Relative path to the file.

pversions (Windows)

Windows version of the command: displays the version information for Platform products installed on a Windows host.

Synopsis

pversions [*product_name*]

pversions -h

pversions -V

Description

Displays the version and patch level of a Platform product installed on a Windows host, and the list of patches installed.

Options

product_name

Specify the Platform product for which you want version information. Specify one of the following:

- EGO—to see version information for Platform EGO
- Symphony—to see version information for Platform Symphony and Symphony Developer's Edition
- LSF—to see version information for Platform LSF

-h

Prints command usage to `stderr` and exits from the software.

-V

Prints product version to `stderr` and exits.

ssacct

displays accounting statistics about finished Session Scheduler jobs

Synopsis

ssacct [**-l**] *job_ID* [*task_ID* | **"***task_ID***[***index***]"**]

ssacct [**-l**] **"***job_ID* **[***index***]"** [*task_ID* | **"***task_ID***[***index***]"**]

ssacct [**-l**] **-f** *log_file* [*job_ID* [*task_ID* | **"***task_ID***[***index***]"**]]

ssacct [**-l**] **-f** *log_file* [**"***job_ID* **[***index***]"** [*task_ID* | **"***task_ID***[***index***]"**]]

ssacct [**-h**] | [**-V**]

Description

By default, displays accounting statistics for all finished jobs submitted by the user who invoked the command.

Options

-l

Long format. Displays additional accounting statistics.

-f *log_file*

Searches the specified job log file for accounting statistics. Specify either an absolute or relative path.

By default, `ssacct` searches for accounting files in SSCHED_ACCT_DIR in `lsb.params`. Use this option to parse a specific file in a different location. You can specify a log file name, or a job ID, or both a log file and a job ID. The following are correct:

```
ssacct -f log_file job_ID
```

```
ssacct -f log_file
```

```
ssacct job_ID
```

The specified file path can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

***job_ID* | "*job_ID*[*index*]"**

Displays information about the specified jobs or job arrays.

***task_ID* | "*task_ID*[*index*]"**

Displays information about the specified tasks or task arrays.

-h

Prints command usage to `stderr` and exits.

-V

Prints Session Scheduler release version to `stderr` and exits.

Output: default format

Statistics on all tasks in the session. The following fields are displayed:

- Total number of done tasks
- Total CPU time in seconds consumed
- Average CPU time in seconds consumed
- Maximum CPU time in seconds of a task
- Minimum CPU time in seconds of a task
- Total wait time in seconds
- Average wait time in seconds
- Maximum wait time in seconds
- Minimum wait time in seconds
- Average turnaround time (seconds/task)
- Maximum turnaround time (seconds/task)
- Minimum turnaround time (seconds/task)
- Average hog factor of a job (CPU time/turnaround time)
- Maximum hog factor of a task (CPU time/turnaround time)
- Minimum hog factor of a task (CPU time/turnaround time)

The total, average, minimum, and maximum statistics are on all specified tasks.

The wait time is the elapsed time from job submission to job dispatch.

The turnaround time is the elapsed time from job submission to job completion.

The hog factor is the amount of CPU time consumed by a job divided by its turnaround time.

Output: long format (-l)

In addition to the fields displayed by default in SUMMARY, `-l` displays the following fields:

CPU_T

CPU time in seconds used by the task

WAIT

Wall clock time in seconds between when the task was submitted to the Session Scheduler and when it has been dispatched to an execution host

TURNAROUND

Wall clock time in seconds between when the task was submitted to the Session Scheduler and when it has completed running

STATUS

Status that indicates the job was either successfully completed (done) or exited (exit)

HOG_FACTOR

Average hog factor, equal to CPU time /turnaround time

Examples: default format

ssacct 108 1[1]

```
Accounting information about tasks that are:
  - submitted by all users.
  - completed normally or exited.
  - executed on all hosts.
------------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done tasks:       1      Total number of exited tasks:     0
 Total CPU time consumed:       0.0      Average CPU time consumed:     0.0
 Maximum CPU time of a task:     0.0      Minimum CPU time of a task:     0.0
 Total wait time:     2.0
 Average wait time:    2.0
 Maximum wait time:    2.0      Minimum wait time:    2.0
 Average turnaround time:         3 (seconds/task)
 Maximum turnaround time:         3      Minimum turnaround time:         3
 Average hog factor of a task:  0.01 ( cpu time / turnaround time )
 Maximum hog factor of a task :  0.01      Minimum hog factor of a task:  0.01
```

Examples: long format (-l)

ssacct -l 108 1[1]

```
Accounting information about tasks that are:
  - submitted by all users.
  - completed normally or exited.
  - executed on all hosts.

------------------------------------------------------------------------------
Job <108>, Task <1>, User <user1>, Status <Done> Command <myjob>
Thu Nov  1 13:48:03 2008: Submitted from host <hostA>;
Thu Nov  1 13:48:05 2008: Dispatched to <hostA>, Execution CWD </home/user1/src>
Thu Nov  1 13:48:06 2008: Completed <done>.

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR
      0.03        2              3     done         0.0113
------------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done tasks:       1      Total number of exited tasks:     0
 Total CPU time consumed:        0.0      Average CPU time consumed:     0.0
 Maximum CPU time of a task:     0.0      Minimum CPU time of a task:     0.0
 Total wait time:     2.0
 Average wait time:    2.0
 Maximum wait time:    2.0      Minimum wait time:    2.0
 Average turnaround time:         3 (seconds/task)
 Maximum turnaround time:         3      Minimum turnaround time:         3
 Average hog factor of a task:  0.01 ( cpu time / turnaround time )
 Maximum hog factor of a task :  0.01      Minimum hog factor of a task:  0.01
```

Files

Reads *job_ID*.ssched.acct

See also

ssched, lsb.params

ssched

submit tasks through Platform LSF Session Scheduler

Synopsis

ssched [*options*] *command*

ssched [*options*] **-tasks** *task_definition_file*

ssched [*options*] **-tasks** *task_definition_file command*

ssched [**-h** | **-V**]

Description

Options can be specified on the ssched command line or on a line in a task definition file. If specified on the command line, the option applies to all tasks, whether specified on the command line or in a file. Options specified in a file apply only to the command on that line. Options in the task definition file override the same option specified on the command line.

ssched exit codes

Exit Code	Meaning
0	All tasks completed normally
1	An unspecified error occurred
3	All tasks completed, but some tasks have a non-zero exit code
4	Error parsing ssched command line parameters or tasks definition file. No tasks were run.
5	Exceeded the SSCHED_MAX_TASKS limit
6	License expired

Task Definition File Format

The task definition file is an ASCII file. Each line represents one task, or an array of tasks. Each line has the following format:

```
[task_options] command [arguments]
```

Command options

-1 | -2 | -3

Enables increasing amounts of debug output

-C

Sanity check all parameters and the task definition file. Exit immediately after the check is complete. An exit code of 0 indicates no errors were found. Any non-zero exit code indicates an error. `ssched -C` can be run outside of LSF.

-p

Do not delete the temporary working directory. This option is useful when diagnosing errors.

Task options

-E "*pre_exec_command* [*arguments* ...]"

Runs the specified pre-execution command on the execution host before actually running the task.

The task pre-execution behavior mimics the behavior of LSF job pre-execution. However, the task pre-execution command cannot run as root.

The standard input and output for the pre-execution command are directed to the same files as the job. The pre-execution command runs under the same user ID, environment, home, and working directory as the job. If the pre-execution command is not in the user's usual execution path (the $PATH variable), the full path name of the command must be specified.

-Ep "*post_exec_command* [*arguments* ...]"

Runs the specified post-execution command on the execution host after the task finishes.

The task post-execution behavior mimics the behavior of LSF job post-execution. However, the task post-execution command cannot run as root.

If the post-execution command is not in the user's usual execution path (the $PATH variable), the full path name of the command must be specified.

-e *error_file*

Specify a file path. Appends the standard error output of the job to the specified file.

If the parameter LSB_STDOUT_DIRECT in `lsf.conf` is set to Y or y, the standard error output of a task is written to the file you specify as the job runs. If LSB_STDOUT_DIRECT is not set, standard error output of a task is written to a temporary file and copied to the specified file after the task finishes.

You can use the special characters %J, %I, %T, %X in the name of the input file. %J is replaced by the job ID. %I is replaced by the job array index, %T is replaced with the task ID, and %X is replaced by the task array index.

If the current working directory is not accessible on the execution host after the job starts, Session Scheduler writes the standard error output file to `/tmp/`.

Note:

The file path can contain up to 4094 characters including the directory, file name, and expanded values for %J, %I, %T and %X

-i *input_file*

Gets the standard input for the job from specified file. Specify an absolute or relative path. The input file can be any type of file, though it is typically a shell script text file.

If -i is not specified, standard input defaults to `/dev/null`.

You can use the special characters %J, %I, %T, %X in the name of the input file. %J is replaced by the job ID. %I is replaced by the job array index, %T is replaced with the task ID, and %X is replaced by the task array index.

Note:

The file path can contain up to 4094 characters including the directory, file name, and expanded values for %J, %I, %T and %X

-J *task_name*[*index_list*]

Specifies the indices of the task array. The index list must be enclosed in square brackets. The index list is a comma-separated list whose elements have the syntax *start* [-*end*[:*step*]] where *start*, *end* and step are positive integers. If the step is omitted, a step of one is assumed. The task array index starts at one.

All tasks in the array share the same option parameters. Each element of the array is distinguished by its array index.

-j "*starter* [*starter*] ['%USRCMD'] [*starter*]"

Task job starter. Creates a specific environment for submitted tasks prior to execution.

The job starter is any executable that can be used to start the task (that is, it can accept the task as an input argument). Optionally, additional strings can be specified.

By default, the user commands run after the job starter. A special string, % USRCMD, can be used to represent the position of the user's task in the job starter command line. The %USRCMD string may be followed by additional commands.

-o *output_file*

Specify a file path. Appends the standard output of the task to the specified file. The default is to output to the same `stdout` as the `ssched` command.

If only a file name is specified, LSF writes the output file to the current working directory. If the current working directory is not accessible on the execution host after the task starts, LSF writes the standard output file to `/tmp/`.

If the parameter LSB_STDOUT_DIRECT in `lsf.conf` is set to Y or y, the standard output of a task is written to the file you specify as the task runs. If LSB_STDOUT_DIRECT is not set, it is written to a temporary file and copied to the specified file after the task finishes.

You can use the special characters %J, %I, %T, %X in the name of the input file. %J is replaced by the job ID. %I is replaced by the job array index, %T is replaced with the task ID, and %X is replaced by the task array index.

Note:

The file path can contain up to 4094 characters including the directory, file name, and expanded values for %J, %I, %T and %X

-M *mem_limit*

Sets a per-process (soft) memory limit for all the processes that belong to the task (see getrlimit(2)).

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

You should only set a task level memory limit if it less than the job limit.

-Q "*exit_code* ..."

Task requeue exit values. Enables automatic task requeue and sets the LSB_EXIT_REQUEUE environment variable. Separate multiple exit codes with spaces. The output from the failed run is not saved, and the user is not notified by LSF.

-W [*minutes*:]*seconds*

Sets the run time limit of the task. If a task runs longer than the specified run limit, the task is sent a SIGKILL signal.

The run limit is in the form of [*minutes*:]*seconds*. The seconds can be specified as a number greater than 59. For example, three and a half minutes can either be specified as 3:30, or 210. The run limit you specify is the absolute run time.

-tasks *task_definition_file*

Specify tasks through a task definition file.

***command* [*argument*]**

The command can be anything that is provided to a UNIX Bourne shell (see sh(1)). The command is assumed to begin with the first word that is not part of a option. All arguments that follow command are provided as the arguments to the command.

The job command can be up to 4094 characters long.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

See also

ssacct, lsb.params

taskman

checks out a license token and manages interactive UNIX applications

Synopsis

taskman -Lp *project* **-R** "rusage[*token*=*number***[:duration**=*minutes* | *hours* h**]** [:*token*=*number*[**:duration**=*minutes* | *hours* h]]...] [**-N** *n_retries*] [**-v**] *command*

taskman [**-h** | **-V**]

Description

Runs the interactive UNIX application on behalf of the user. When it starts, the task manager connects to License Scheduler to request the application license tokens. When all the requested licenses are available, the task manager starts the application. While the application is running, the task manager monitors resource usage, CPU, and memory, and reports the usage to License Scheduler. When the application terminates, the task manager exits.

By default, a license is reserved for the duration of the task, so the application can check out the license at any time. Use the duration keyword if you want unused licenses to be reallocated if the application fails to check out the license before the reservation expires.

Options

command

Required. The command to start the job that requires the license.

-v

Verbose mode. Displays detailed messages about the status of configuration files.

-N *n_retries*

Specifies the maximum number of retry attempts taskman takes to connect to the daemon. If this option is not specified, taskman retries indefinitely.

-Lp *project*

Required. Specifies the interactive license project that is requesting tokens. The client must be known to LSF License Scheduler.

-R "rusage[token=*number* [:*duration=minutes* | *hours* h] [:*token=number* [:*duration=minutes* | *hours* h]] ...]

Required. Specifies the type and number of license tokens to request from License Scheduler. Optionally, specifies a time limit for the license reservation, expressed as an integer (the keyword h following the number indicates hours instead of minutes). You may specify multiple license types, with different duration values. Separate each requirement with a colon (:). Enclose the entire list in one set of square brackets.

-h

Prints command usage to `stderr` and exits.

-V

Prints the License Scheduler release version to `stderr` and exits.

tspeek

displays the `stdout` and `stderr` output of an unfinished Terminal Services job

Synopsis

tspeek *job_ID*

tspeek [**-h** | **-V**]

Description

Displays the standard output and standard error output that have been produced by one of your unfinished Terminal Services jobs, up to the time that this command is invoked.

This command is useful for monitoring the progress of a job and identifying errors. If errors are observed, valuable user time and system resources can be saved by terminating an erroneous job.

`tspeek` is supported on Windows and Linux. You cannot use `tspeek` to monitor job output from UNIX. `tspeek` on Linux requires `rdesktop`.

You can use `tspeek` from any Linux host where `rdesktop` is installed to view the output of a Terminal Services job. For example, if your job ID is 23245, run:

```
tspeek 23245
```

Options

job_ID

Operates on the specified Terminal Services job.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

tssub

tssub

submits a Terminal Services job to LSF

Synopsis

tssub [*bsub_options*] *command* [*arguments*]

tssub [**-h** | **-V**]

Description

Submits a Terminal Services job for batch execution and assigns it a unique numerical job ID.

`tssub` is a wrapper around the `bsub` command that only submits jobs to hosts that have Microsoft Terminal Services installed. For `bsub` options, see the bsub command.

You submit Terminal Services job with `tssub` instead of `bsub`. If the terminal window is closed, the job remains running. You can reconnect to view the job with `tspeek`.

`tssub` is supported on Windows and Linux. You cannot use `tssub` to submit Terminal Services jobs from UNIX.

If the job is dispatched to a host in which Terminal Services is not installed or properly configured, the job is set to the PEND state and a pending reason is written in `sbatchd.log.`*`host_name`*.

If `tssub -I` is specified, a terminal display is visible on the submission host after the job has been started.

If the job is not a GUI job, LSF runs a command window and output is displayed in the command window when something is written to `stdout`.

Pre- and post-execution commands are executed within the terminal session. The job does not complete until post-execution commands complete.

If you use `bjobs -l` to monitor the job, you see a message similar to `"External Message 2 was posted from LSF\lsfadmin to message box 2"`. The body of the message contains the ID of the terminal session that was created.

Use tspeek to view job output.

`tssub` sets the LSB_TSJOB and LSF_LOGON_DESKTOP environment variables. These variables are then transferred to the execution host:

LSF_LOGON_DESKTOP

When LSF_LOGON_DESKTOP=1, jobs run in interactive foreground sessions. This allows GUIs to be displayed on the execution host. If this parameter is not defined, jobs run in the background.

LSB_TSJOB

When the LSB_TSJOB variable is defined to any value, it indicates to LSF that the job is a Terminal Services job.

Limitations

- You cannot use `bmod` to modify a job submitted as a Terminal Services job to become a non-Terminal Services job
- The `bsub` option -o *out_file* is not supported for `tssub`
- Only Windows `bsub` options are supported for `tssub`. For example, you cannot use the options -Ip, -Is, -L *login_shell* of `bsub` with `tssub`.

- Interactive `bsub` options (-I, -Ip, -Is) are not supported with `tssub` on Linux
- If user mapping is defined, the user who invokes `tspeek` must have the required privileges to access the session
- MultiCluster is not supported

Options

bsub_options

Only Windows `bsub` options are supported for `tssub`. For example, you cannot use the options -Ip, -Is, -L *login_shell* of `bsub` with `tssub`.

For `bsub` options, see the bsub command.

***command* [*argument*]**

The job can be specified by a command line argument command, or through the standard input if the command is not present on the command line. The *command* is assumed to begin with the first word that is not part of a `tssub` option. All arguments that follow *command* are provided as the arguments to the *command*.

The job command can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. If no job name is specified with `-J`, `bjobs`, `bhist` and `bacct` displays the command as the job name.

The commands are executed in the order in which they are given.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

bsub, tspeek

wgpasswd

changes a user's password for an entire Microsoft Windows workgroup

Synopsis

wgpasswd [*user_name*]

wgpasswd [**-h**]

Description

You must run this command on a host in a Windows workgroup. You must have administrative privileges to change another user's password.

Prompts for old and new passwords, then changes the password on every host in the workgroup.

By default, modifies your own user account.

Options

user_name

Specifies the account to modify. You must have administrative privileges to change another user's password.

-h

Prints command usage to `stderr` and exits.

Output

For each host in the workgroup, returns the status of the operation (SUCCESS or FAILED).

Files

Modifies the LSF password file.

wguser

modifies user accounts for an entire Microsoft Windows workgroup

Synopsis

wguser [**-r**] *user_name ...*

wguser [**-h**]

Description

Caution:

You must run this command on a host in a Microsoft Windows workgroup. You should have administrative privileges on every host in the workgroup.

Modifies accounts on every host in the workgroup that you have administrative privileges on.

By default, prompts for a default password to use for all of the accounts, and then creates the specified user accounts on each host, if they do not already exist.

Use -r to remove accounts from the workgroup.

Options

-r

Removes the specified user accounts from each host, if they exist.

***user_name* ...**

Required. Specifies the accounts to add or remove.

-h

Prints command usage to `stderr` and exits.

Output

For each host in the workgroup, returns the result of the operation (SUCCESS or FAILED).

Index

A

ABS_RUNLIMIT
    lsb.params file
        bsub -W 233
absolute path
    lsfinstall options 269
Active status
    bqueues 150
ACTIVE WINDOW
    bsla 193
Active\
    Missed status
        bsla 193
    Ontime status
        bsla 193
ADJUST
    bhpart 69, 162
ADMIN
    blinfo output 113
    blparams output 117
    lsclusters 266
ADMIN ACTION COMMENT
    bhosts -l 66
    bqueues -l 160
ADMINISTRATORS
    bqueues -l 158
advance reservations
    bacct -U 13
ALLOCATION
    blinfo output 111
APP_NAME
    bclusters 45
Application Profile
    bhist -l 56
    bjobs -l ouput 83
APPLICATION_NAME
    bapp 34
ARRAY_SPEC
    bjobs -A 88

B

BACKFILL
    bqueues -l 156
BIND_JOB
    bapp -l 36
bjdepinfo 70
bld
    License Scheduler daemon 96, 104
Blocks in
    lsacct 253
Blocks out
    lsacct 253
BUILTIN
    lsinfo -l 286
bulk jobs
    killing 92

C

CHECKPOINT
    bqueues -l 160
Checkpoint directory
    bhist -l 57
    bjobs -l 83
Checkpoint period
    bhist -l 57
    bjobs -l 83
CHKPNT_DIR
    bapp -l 37
CHKPNT_INITPERIOD
    bapp -l 37
CHKPNT_METHOD
    bapp -l 37
CHKPNT_PERIOD
    bapp -l 37
CHKPNTDIR

bqueues -l 160
CHKPNTPERIOD
bqueues -l 160
chunk jobs
bmig 134
bsub restrictions 205
bswitch 241
CHUNK_JOB_SIZE
bapp -l 38
bqueues -l 159
Closed status
bqueues 150
CLUSTER
bclusters 44
blusers output 130
CLUSTER_NAME
blinfo output 112
lsclusters 265
Command
bhist -l output 56
bjobs -l 83
Command line
lsacct -l output 253
COMPL_TIME
bacct -l 12
Completion time
lsacct -l 253
compound resource requirements
syntax 227
CONNECT_TIME column
glbtasks -g output 128
CONSUMABLE
lsinfo -l 286
CONSUMER
bsla 192
CORELIMIT
bapp -l 36
bqueues -l 154
CPU time
bjobs -l 86
lsacct 252
CPU_T
bacct -b 11
ssacct -l 338
CPU_TIME
bhpart 69, 161
cpuf
lshosts 280
CPUF
bhosts -l 65
CPULIMIT
bapp -l 35
bqueues -l 152
CPUSET_OS
brlainfo 174
CREATOR
bacct -U 13
CURRENT LOAD
bhosts -l 65
CWD
bacct -l 12
bjobs -l 83
lsacct -l 253

D

DATALIMIT
bapp -l 36
bqueues -l 153
DEFAULT HOST SPECIFICATION
bqueues -l 157
Default queue indication
bqueues -l 151
DEFAULT_EXTSCHED
lsb.queues file
bsub -ext 215
DEFAULT_HOST_SPEC
lsb.params file
bsub -c 212
lsb.queues file
bsub -c 212
DEFAULT_QUEUE
lsb.params file
bsub command 206
DEMAND
blstat output 123
Description
bapp -l 35
bqueues -l 151
DESCRIPTION
bclusters 45
blinfo output 113
DISPAT_TIME
bacct -l 12
DISPATCH_WINDOWS
bhosts -l 65

bqueues -l 157
display job dependencies 70
DISPLAYS
blusers output 131
DISTRIBUTION
blinfo output 111
DISTRIBUTION_POLICY_VIOLATION_ACTION
blinfo output 113
blparams output 117
DONE
bjobs -A 88
bjobs -l 84
dual-core CPUs
license needed in lshosts -l 282
DYNAMIC
lsinfo -l 286
dynamic slave host
lsfinstall -s option 270
slave.config file variables 269

E

EGO_RES_REQ
bsla 192
ENABLE_DEFAULT_EGO_SLA
lsb.params file
bsub -sla 230
ENABLE_ONE_UG_LIMITS 80
ERR_FILE
bacct -l 13
ESTIMATED FINISH TIME
bsla 193
EXCEPTION LOAD AND THRESHOLD
bhosts -l 66
EXCEPTION STATUS
bjobs -l 13, 87
EXCLUSIVE
bqueues -l 156
EXEC_HOST
bjobs 82
EXEC_ON
bacct -b 11
Execution host
lsacct -l 253
EXIT
bjobs -A 88
bjobs -l 84
Exit status
lsacct -l 254
exited jobs
bhist -e 52
external_index
lsload 290

F

FAIRSHARE
bqueues -l 156
bqueues -r 162
FAIRSHARE_QUEUES
bqueues -l 157
FEATURE
blinfo output 111, 112
blstat output 122
blusers output 129
FILELIMIT
bapp -l 36
bqueues -l 153
FINISH
bjgroup
default output 74
bsla 194
finished jobs
bhist -d 52
FLEX_NAME
blinfo output 112
FREE
blstat output 123
FREE CPU LIST
brlainfo 175
FREECPUS
brlainfo 174
FROM
bacct -b 11
FROM_HOST
bjobs 82

G

GOAL
bsla 193
GROUP
blinfo output 112
GROUP_NAME
bjgroup
default output 73

job slots (-N) output 74

H

HOG_FACTOR
bacct -l 12
ssacct -l 338
HOST
blusers output 130
HOST column
glbtasks -g output 128
HOST_NAME
bhosts 62
lshosts 280
lsload 288
lsmon 300
HOST_PARTITION_NAME
bhpart 68
HOSTNAME
brlainfo 174
hosts
lost_and_found 62, 83
HOSTS
bhpart 68
blimits 107
blinfo output 113
blparams output 117
bqueues -l 158
lsclusters 266

I

idle job exception
bacct -l -x 13
bjobs -l 87
bqueues -l 156
IDLE_FACTOR
bjobs -l 86
IGNORE_DEADLINE
bqueues -l 156
lsb.applications file
bsub -W 233
Inact_Adm status
bqueues -l 151
Inact_Win status
bqueues -l 151
Inactive status
blsa 193
bqueues 150
bqueues -l 151
Initial checkpoint period
bhist -l 57
bjobs -l 83
INPUT_FILE
bacct -l 12
install.config file
required variables 269
interactive jobs
submitting 207
INTERVAL
lsinfo -l 286
Interval for a host to accept two jobs
bqueues -l 152
INUSE
blstat output 123
Involuntary con sw
lsacct 253
io
bqueues -l 154
lsload 290
it
bqueues -l 155
lsload 289
lsmon 301

J

JL/H
bqueues 151
JL/P
bqueues 150
busers 246
JL/U
bhosts 63
bqueues 150
JLIMIT
bjgroup
default output 74
JOB CONTROLS
bqueues -l 160
job dependencies
displaying 70
JOB EXCEPTION PARAMETERS
bqueues -l 155
job exceptions
bacct 16

job exit codes
bhist -l 53
job migration
absolute job priority scheduling 134
job requeue
absolute job priority scheduling 165
JOB STATUS
bjobs -l 84
JOB_CONTROLS
lsb.queues file
bsub -t 231
JOB_DEP_LAST_SUB
lsb.params file
bsub -w 234
JOB_FLOW
bclusters 43
JOB_INCLUDE_POSTPROC
bapp -l 37
JOB_NAME
bacct -b 12
bjobs 83, 89
JOB_POSTPROC_TIMEOUT
bapp -l 38
JOB_SPOOL_DIR
lsb.params file
bsub -i 218
bsub -Zs 237
JOB_STARTER
bapp -l 38
bqueues -l 159
JOBID
bacct -l 12
bjobs 82
bjobs -A 88
blusers output 130
JOBS
blimits 108
JSDL
jsdl_strict option 219
submit a job using bsub 219

K

kill
job array 94

L

LIC_COLLECT
lsf.licensescheduler file 102
LIC_SERVERS
blinfo output 112
LICENSE CLASS NEEDED
lshosts -l 282
LICENSES_ENABLED
lshosts -l 282
LIMITS
blinfo output 113, 123
LM_REMOVE_INTERVAL
blinfo output 113
blparams output 117
LM_STAT_INTERVAL
blinfo output 114
blparams output 117
LOAD THRESHOLD
bhosts -l 66
LOAD_THRESHOLDS
lshosts -l 282
loadSched
bhosts -l 64
bjobs -l 84
loadStop
bhosts -l 64
bjobs -l 84
LOCAL_QUEUE
bclusters 43
LOCATION
bhosts -s 66
lshosts -s 282
lsload -s 290
lost_and_found host 62, 83
lost_and_found queue 107
bqueues 82
lost_and_found queue name
bqueues 149
ls
bqueues -l 154
lsload 289
lsmon 301
LS_ENABLE_MAX_PREEMPT
blinfo output 114
blparams output 118
LS_MAX_TASKMAN_PREEMPT
blinfo output 114
blparams output 118
LS_MAX_TASKMAN_SESSIONS
blinfo output 114
blparams output 118

LSF_DESERVE
bltstat output 122
LSF_ENABLE_EXTSCHEDULER
bsub 215
LSF_FREE
bltstat output 122
LSF_LIC_SCHED_HOSTS
blinfo output 114
blparams output 119
LSF_LIC_SCHED_PREEMPT_REQUEUE
blinfo output 114
blparams output 119
LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE
blinfo output 114
blparams output 119
LSF_LIC_SCHED_PREEMPT_STOP
blinfo output 114
blparams output 119
LSF_LICENSE_FILE
blinfo output 114
blparams output 119
LSF_USE
bltstat output 122
lsfinstall command
location 269
lsrcp 306

M

MANDATORY_EXTSCHED
lsb.queues file
bsub -ext 215
MASTER_HOST
lsclusters 266
MAX
bhosts 63
bqueues 150
busers 247
MAX_HOST_IDLE_TIME
bsla 193
Maximum slot reservation time
bqueues -l 159
maxmem
lshosts 281
maxswp
lshosts 281
maxtmp
lshosts -l 281
mem
bqueues -l 155
lsload 290
lsmon 301
MEM
bacct -l 12
bjobs -l 86
blimits 108
MEMLIMIT
bapp -l 35
bqueues -l 153
MEMLIMIT_TYPE bapp -l 36
Messages rcvd
lsacct 253
Messages sent
lsacct 253
MIG
bapp -l 37
migrated jobs
absolute job priority scheduling 134
Migration threshold
bhist -l 57
bjobs -l 83
bqueues -l 152
model
lshosts 280
MPEND
busers 247
multi-core CPUs
ncpus in lshosts 280
multiple resource requirement strings
bhist -l 52

N

NAME
blimits 107
blinfo output 111
ncores
lshosts 281
lshosts -l 282
NCPU/NODE NSTATIC_CPUSETS
brlainfo 174
ncpus
lshosts 280
NCPUS
bacct -U 13
brlainfo 174

ndisks
lshosts -l 281
NFREECPUS ON EACH NODE
brlainfo 175
NICE
bqueues -l 151
NJOBS
bapp 34
bjgroup
default output 73
job slots (-N) output 74
bjobs -A 88
bqueues 151
bsla 194
busers 247
NJOBS bhosts 63
NLICS
blusers output 130
NNODES
brlainfo 174
NO_INTERACTIVE
bqueues -l 156
NON_LSF_DESERVE
blstat output 122
NON_LSF_FREE
blstat output 123
NON_LSF_USE
blstat output 122
NON_SHARED
blinfo output 113
blstat output 123
NON-SHARED_DISTRIBUTION
blinfo output 111
nprocs
lshosts 281
lshosts -l 282
NQS DESTINATION QUEUES
bqueues -l 158
NSTATIC_CPUSETS
brlainfo 175
NTASKS
blusers output 130
NTHREAD
bjobs -l 86
nthreads
lshosts 281
lshosts -l 282
NUM_RECALLED_HOSTS
bsla 193

O

ONLY_INTERACTIVE
bqueues -l 156
Open status
bqueues 150
OPTIMUM NUMBER OF RUNNING JOBS
bsla 193
ORDER
lsinfo -l 286
OTHERS
blstat output 122
blusers output 131
OUTPUT_FILE
bacct -l 12
overrun job exception
bacct -l -x 13
bjobs -l 87
bqueues -l 155
OWN
blstat output 123
OWNER
bjgroup
default output 74
job slots (-N) output 75
bjobs -A 88
OWNERSHIP
blinfo output 112

P

Page faults
lsacct 252
PARALLEL_SCHED_BY_SLOT
lsb.params file
bsub -n 223
PEND
bapp 35
bhist 56
bjgroup
default output 73
job slots (-N) output 74
bjobs -A 88
bjobs -l 84
bqueues 151
bsla 194

busers 247
pending jobs
bhist -p 53
PENDING REASONS
bjobs -l 84
pg
bqueues -l 154
lsload 289
lsmon 301
PGID
bjobs -l 87
PID
lsacct -l 253
PIDs
bjobs -l 87
PMC configuration
automatic startup
removing (pmcremoverc command) 330
setting (pmcsetrc command) 331
POLICY
glbtasks -g output 128
PORT
blinfo output 114
blparams output 119
POST_EXEC
bapp -l 37
bqueues -l 159
lsb.applications file
bsub -Ep 214
lsb.queues file
bsub -Ep 214
Post-execute Command
bjobs -l 84
PRE_EXEC
bapp -l 37
bqueues -l 159
lsb.applications file
bsub -E 213
lsb.queues file
bsub -E 213
pre- and post-execution processing
job level
job state post_done 236
job state post_err 236
PREEMPTABLE
bqueues 160
PREEMPTION parameter
bqueues 160
PREEMPTIVE
bqueues 160
bqueues -l 160
PRIO
bqueues 149
PRIORITY
bhpart 69, 161
bsla 192
PROCESSLIMIT
bapp -l 36
bqueues -l 153
PROCLIMIT
bapp -l 36
bqueues -l 152
lsb.queues file
bsub -n 223
Project
bhist -l 56
bjobs -l 83
PROJECT
blinfo output 111
blstat output 123
blusers output 130
PROJECT_NAME
bacct -l 12
PROJECT/GROUP
blstat output 124
PROJECTS
blimits 108
PSUSP
bhist 56
bjobs -A 88
bjobs -l 84

Q

QUEUE
bacct -b 11
bjobs 82
QUEUE_NAME
bqueues 149
queues
lost_and_found 82, 107
QUEUES
blimits 107

R

r15m
bqueues -l 154
lsload 289
lsmon 300
r15s
bqueues -l 154
lsload 289
lsmon 300
r1m
bqueues -l 154
lsload 289
lsmon 300
RECALLED_HOSTS_TIMEOUT
bsla 193
RECEIVE_JOBS_FROM
bqueues -l 159
RELEASE
lsinfo -l 286
REMOTE
bclusters 43
remote shell
lsrcp 306
REMOTE_CLUSTER
bclusters 44, 45
reporting configuration
automatic startup
removing (perfremovrc command) 326
setting (perfsetrc command) 327
REQUEUE_EXIT_VALUES
bapp -l 38
bqueues -l 159
requeued jobs
absolute job priority scheduling 165
required install.config and slave.config variables 269
RERUNNABLE
bapp -l 38
bqueues -l 160
RES_REQ
bapp -l 38
bqueues -l 159
RESERVE
blstat output 123
RESERVED
bhosts -s 66
bhpart 69, 161
resizable job information
bacct 19
bhist -l 53
RESOURCE
bhosts -s 66
blusers output 130
lshosts -s 282
lsload -s 290
RESOURCE LIMITS
bapp -l 35
bjobs -l 87
bqueues -l 152
resource requirements
compound
syntax 227
RESOURCE USAGE
bjobs -l 86
Resource usage of tasks selected
lsacct 252
RESOURCE_FLOW
bclusters 44
RESOURCES
lshosts 281
RESUME_COND
bqueues -l 159
RESUME_CONTROL
bapp -l 38
rexpri
lshosts -l 282
rsh command
badmin hstartup all 27
lsadmin limstartup all 257
lsadmin resstartup all 259
lsfrestart lsfshutdown lsfstartup 273–275
lsrcp 306
RSV
bapp -l 35
bhosts 64
bjgroup
job slots (-N) output 75
bqueues -l 152
busers 247
RSV_HOSTS
bacct -U 13
RSVID
bacct -U 13
RUN
bapp 35
bhist 56

bhosts 63
bjgroup
default output 73
job slots (-N) output 75
bjobs -A 88
bjobs -l 84
bqueues 151
bsla 194
busers 247
RUN column
glbtasks -g output 128
RUN_TIME
bhpart 69, 161
RUN_WINDOWS
bqueues -l 157
lshosts -l 282
RUNLIMIT
bapp -l 36
bqueues -l 153
running jobs
bhist -r 53
RUNTIME
bjobs -l 85
runtime estimate exceeded
bhist -l 53
runtime_est_exceeded job exception
bacct -l -x 13
RUSAGE
blusers output 130

S

Schedule delay for a new job
bqueues -l 152
SCHEDULING PARAMETERS
bqueues -l 154
SCHEDULING POLICIES
bqueues -l 156
bqueues -r 162
secure shell 306
SEND_JOBS_TO
bqueues -l 159
server
lshosts 281
SERVERS
lsclusters 266
SERVICE CLASS NAME
bsla 192
SERVICE_DOMAIN
blinfo output 111, 112
blstat output 122
blusers output 129, 130
Session Scheduler
job summary 88
SHARE
blstat output 123
SHARE_INFO_FOR
blstat output 124
SHARES
bhpart 68, 161
blinfo output 112
SLA
bjgroup
default output 74
job slots (-N) output 75
SLA THROUGHPUT
bsla 193
slave.config file
required variables 269
SLOT_POOL
bqueues -l 161
SLOT_SHARE
bqueues -l 160
SLOTS
blimits 108
ssh command
badmin hstartup all 27
lsadmin limstartup all 257
lsadmin resstartup all 259
lsfrestart lsfshutdown lsfstartup 273–275
lsrcp 306
SSUSP
bapp -l 35
bhist 56
bhosts 64
bjgroup
default output 74
job slots (-N) output 75
bjobs -A 88
bjobs -l 84
bqueues -l 151
bsla 194
busers 247
STACKLIMIT
bapp -l 36
bqueues -l 153

start and end time
bhist 58
START_TIME
blusers output 130
STARTED
bhpart 69, 161
Starting time
lsacct -l 253
STAT
bjobs 82
STATIC CPUSETS
brlainfo 175
status
lsload 288
lsmon 300
STATUS
bacct -l 12
bclusters 44
bhosts 62
bhosts -l 64
bqueues 150
bqueues -l 151
bsla 193
lsclusters 265
ssacct -l 338
STOP_COND
bqueues -l 159
SUBMIT_TIME
bacct -b 11
bjobs 83
SUSP
bapp 35
bqueues 151
SUSPEND_CONTROL
bapp -l 38
suspended jobs
bhist -s 53
SUSPENDING REASONS
bjobs -l 84
SWAP
bacct -l 12
bjobs -l 86
SWAPLIMIT
bapp -l 36
bqueues -l 153
Swaps
lsacct 253
swp
bqueues -l 155
lsload 290
lsmon 301
SWP
blimits 108

T

TASK_NAME
glbtasks -g output 128
TERMINATE_CONTROL
bapp -l 38
lsb.applications file
bsub -t 231
termination reasons
bacct 14
THREADLIMIT
bapp -l 36
bqueues -l 153
THROUGHPUT
bsla 193
time interval format
bhist 58
Time range of ended tasks
lsacct 252
Time range of started tasks
lsacct 252
time windows
syntax 178
TIME_WINDOW
bacct -U 13
tmp
bqueues -l 155
lsload 289
lsmon 301
TMP
blimits 108
TOTAL
bhist 56
bhosts -s 66
blinfo output 111
TOTAL column
glbtasks -g output 128
Total number of tasks
lsacct 252
TOTAL_FREE
blstat output 122
TOTAL_INUSE

blstat output 122
TOTAL_RESERVE
blstat output 122
Turnaround
lsacct 253
TURNAROUND
bacct -b 11
ssacct -l 338
type
lshosts 280
TYPE
bacct -U 13
lsinfo -l 286

U

U/UID
bacct -b 11
underrun job exception
bacct -l -x 13
bjobs -l 87
bqueues -l 156
unfinished jobs
bhist -a 52
UNKNOWN
blusers output 130
UNKWN
bhist 56
bjobs -l 85
USER
bacct -U 13
bjobs 82
blusers output 130
glbtasks -g output 128
user and host name
lsacct -l 253
USER GROUP
bsla 193
USER_SHARES
bqueues -l 157
USER/GROUP
bhpart 68, 161
busers 246
USERS
blimits 107
bqueues -l 158
USUSP
bapp -l 35
bhist 56
bhosts 64
bjgroup
default output 74
job slots (-N) output 75
bjobs -A 88
bjobs -l 84
bqueues -l 152
bsla 194
busers 247
ut
bqueues -l 154
lsload 289
lsmon 300

V

VALUE
lshosts -s 282
lsload -s 290
Voluntary cont sw
lsacct 253

W

WAIT
bacct -b 11
bjobs -l 85
ssacct -l 338
windows
time 178

X

X-window job
interactive 207

Z

ZOMBI
bjobs -l 85