Using Platform LSF Session Scheduler

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

Copyright

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

Internal redistribution

Trademarks

Third-party license agreements

Contents

CHAPTER

1

Installing Platform LSF Session Scheduler

About Platform LSF Session Scheduler

While traditional Platform LSF job submission, scheduling, and dispatch methods such as job arrays or job chunking are well suited to a mix of long and short running jobs, or jobs with dependencies on each other, Session Scheduler is ideal for large volumes of independent jobs with short run times.

As clusters grow and the volume of workload increases, the need to delegate scheduling decisions increases. Session Scheduler improves throughput and performance of the LSF scheduler by enabling multiple tasks to be submitted as a single LSF job.

The LSF Session Scheduler implements a hierarchal, personal scheduling paradigm that provides very low-latency execution. With very low latency per job, Session Scheduler is ideal for executing very short jobs, whether they are a list of tasks, or job arrays with parametric execution.

The Session Scheduler provides users with the ability to run large collections of short duration tasks within the allocation of an LSF job using a job-level task scheduler that allocate resources for the job once, and reuses the allocated resources for each task.

Each Session Scheduler is dynamically scheduled in a similar manner to a parallel job. Each instance of the `ssched` command then manages its own workload within its assigned allocation. Work is submitted as a task array or a task definition file.

Session Scheduler satisfies the following goals for running a large volume of short jobs:

- Minimize the latency when scheduling short jobs
- Improve overall cluster utilization and system performance
- Allocate resources according to LSF policies
- Support existing LSF pre-execution, post-execution programs, job starters, resources limits, etc.
- Handle thousands of users and more than 50000 short jobs per user

Session Scheduler system requirements

Supported operating systems

Session Scheduler is delivered in the following distributions:

- `lsf7Update6_ssched_linux2.4-glibc2.2-x86_64.tar.Z`
- `lsf7Update6_ssched_linux2.4-glibc2.3-x86.tar.Z`
- `lsf7Update6_ssched_linux2.6-glibc2.3-x86.tar.Z`
- `lsf7Update6_ssched_linux2.6-glibc2.3-x86_64.tar.Z`

Required libraries

Note: These libraries may not be installed by default by all Linux distributions.

On Linux 2.4 (x86 and x86_64), the following external libraries are required:

- `libstdc++.so.5`
- `libpthread-0.60.so` or later
- `libgcc_s.so.1`

On Linux 2.6 (x86), the following external libraries are required:

- `libstdc++.so.5`
- `libpthread-2.3.4.so` or later

On Linux 2.6 (x86_64), the following external libraries are required:

- `libstdc++.so.5`
- `libpthread-2.3.4.so` or later
- `libxml2.so.2`

Compatible Linux distributions

Certified compatible distributions include:

- Red Hat Enterprise Linux AS 3 or later
- SUSE Linux Enterprise Server 10

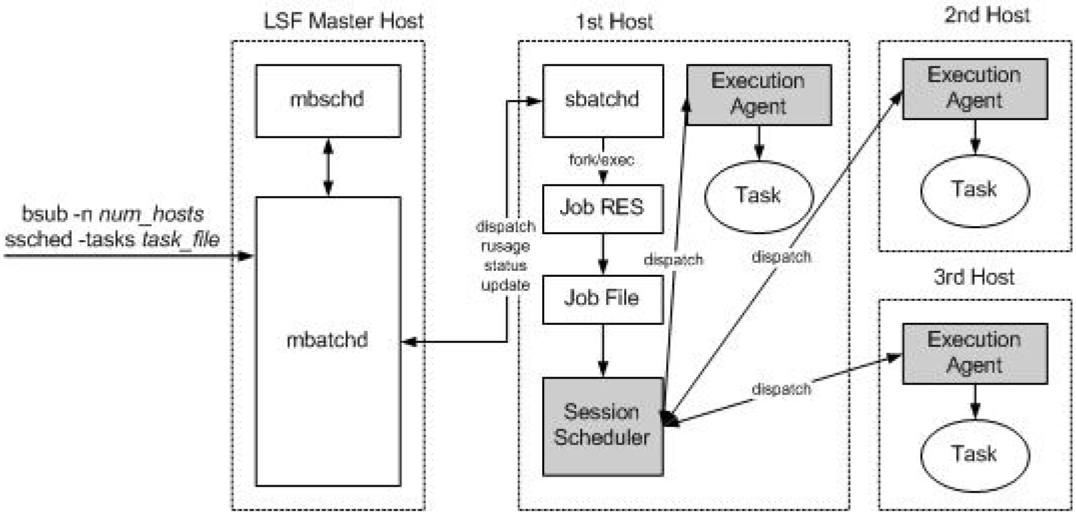
Platform LSF

Session Scheduler requires Platform LSF 7 Update 3 or later

Session Scheduler terminology

Job	A traditional LSF job that is individually scheduled and dispatched to `sbatchd` by `mbatchd` and `mbschd`
Task	Similar to a job, a unit of workload that describes an executable and its environment that runs on an execution node. Tasks are managed and dispatched by the Session Scheduler.
Job Session	An LSF job that is individually scheduled by `mbatchd`, but is not dispatched as an LSF job. Instead, a running Session Scheduler job session represents an allocation of nodes for running large collections of tasks
Scheduler	The component that accepts and dispatches tasks within the nodes allocated for a job session.

Session Scheduler architecture



Session Scheduler jobs are submitted, scheduled, and dispatched like normal LSF jobs.

When the Session Scheduler begins running, it starts a Session Scheduler execution agent on each host in its allocation.

The Session Scheduler then reads in the task definition file, which contains a list of tasks to run. Tasks are sent to an execution agent and run. When a task finishes, the next task in the list is dispatched to the available host. This continues until all tasks have been run.

Tasks submitted through Session Scheduler bypass the LSF `mbatchd` and `mbschd`. The LSF `mbatchd` is unaware of individual tasks.

Session Scheduler components

Session Scheduler comprises the following components.

Session Scheduler command (ssched)

The `ssched` command accepts and dispatches tasks within the nodes allocated for a job session. It reads the task definition file and sends tasks to the execution agents. `ssched` also logs errors, performs task accounting, and requeues tasks as necessary.

sservice and sschild

These components are the execution agents. They run on each remote host in the allocation. They set up the task execution environment, run the tasks, and enable task monitoring and resource usage collection.

Session Scheduler performance

Session Scheduler has been tested to support up to 50,000 tasks. Based on performance tests, the best maximum allocation size (specified by `bsub -n`) depends on the average runtime of the tasks. Here are some typical results:

Average Runtime (seconds)	Recommended maximum allocation size (slots)
0	12
5	64
15	256
30	512

Directions for future development

Future enhancements to Session Scheduler may include:

- Users can list the currently running tasks
- Individual tasks can be removed or killed
- Additional tasks can be dynamically added to a Session Scheduler job

Install Platform LSF Session Scheduler

- Use the `lsfinstall` script to install the Session Scheduler package when you install LSF.

CHAPTER

2

Using Platform LSF Session Scheduler

How Session Scheduler Runs Tasks

Once a Session Scheduler session job has been dispatched and starts running, Session Scheduler parses the task definition file specified on the `ssched` command. Each line of the task definition file is one task. Tasks run on the hosts in the allocation in any order. Dependencies between tasks are not supported.

Session Scheduler status is posted to the Session Scheduler session job through the LSF `bpost` command. Use `bread` or `bjobs -l` to view Session Scheduler status. The status includes the current number of pending, running and completed tasks. LSF administrators can configure how often the status is updated.

When all tasks are completed, the Session Scheduler exits normally.

`ssched` runs under the submission user account. Any processes it creates, either locally or remotely, also run under the submission user account. Session Scheduler does not require any privileges beyond those normally granted a user.

Session Scheduler job sessions

The Session Scheduler session job is compatible with all currently supported LSF job submission and execution parameters, including pre-execution, post-execution, job-starters, I/O redirection, queue and application profile configuration.

Run limits are interpreted and enforced as normal LSF parallel jobs. Application-level checkpointing is also supported. Job chunking is not relevant to Session Scheduler jobs since a single Session Scheduler session is generally long running and should not be chunked.

If the Session Scheduler session is killed (`bkill`) or requeued (`brequeue`), the Session Scheduler kills all running tasks, execution agents, and any other processes it has started, both local and remote. The session scheduler also cleans up any temporary files created and then exits. If the session scheduler is then requeued and restarted, all tasks are rerun.

If the Session Scheduler session is suspended (`bstop`), the Session Scheduler and all local and remote components will be stopped until the session is resumed (`bresume`).

Session Scheduler tasks

`ssched` and `sservice` and `sschild` execution agents ensure that the user submission environment variables are set correctly for each task. In order to minimize the load on the LSF, `mbatchd` does not have any knowledge of individual tasks.

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]
```

Session and task accounting

Jobs corresponding to the Session Scheduler session have one record in lsb.acct. This record represents the aggregate resource usage of all tasks in the allocation.

If task accounting is enabled with SSCHED_ACCT_DIR in lsb.params, Session Scheduler creates task accounting files for each Session Scheduler session job and appends an accounting record to the end of the file. This record follows a similar format to the LSF accounting file `lsb.acct` format, but with additional fields/

The accounting file is named *`jobID`*`.ssched.acct`. If no directory is specified, accounting records are not written.

The Session Scheduler accounting directory must be accessible and writable from all hosts in the cluster. Each Session Scheduler session (each `ssched` instance) creates one accounting file. Each file contains one accounting entry for each task. Each completed task index has one line in the file. Each line records the resource usage of one task.

Task accounting file format

Task accounting records have a similar format as the `lsb.acct` JOB_FINISH event record. See the *Platform LSF Configuration Reference* for more information about JOB_FINISH event fields.

Field	Description
Event type (%s)	TASK_FINISH
Version Number (%s)	7.06
Event Time (%d)	Time the event was logged (in seconds since the epoch)
jobId (%d)	ID for the job
userId (%d)	UNIX user ID of the submitter
options (%d)	Always 0
numProcessors (%d)	Always 1
submitTime (%d)	Task enqueue time
beginTime (%d)	Always 0
termTime (%d)	Always 0
startTime (%d)	Task start time
userName (%s)	User name of the submitter
queue (%s)	Always empty
resReq (%s)	Always empty
dependCond (%s)	Always empty
preExecCmd (%s)	Task pre-execution command
fromHost (%s)	Submission host name
cwd (%s)	Execution host current working directory (up to 4094 characters)
inFile (%s)	Task input file name (up to 4094 characters)
outFile (%s)	Task output file name (up to 4094 characters)
errFile (%s)	Task error output file name (up to 4094 characters)
jobFile (%s)	Task script file name
numAskedHosts (%d)	Always 0

Field	Description
askedHosts (%s)	Always empty
numExHosts (%d)	Always 1
execHosts (%s)	Name of task execution host
jStatus (%d)	64 indicates task completed normally. 32 indicates task exited abnormally
hostFactor (%f)	CPU factor of the task execution host
jobName (%s)	Always empty
command (%s)	Complete batch task command specified by the user (up to 4094 characters)
lsfRusage (%f)	All rusage fields contain resource usage information for the task
mailUser (%s)	Always empty
projectName (%s)	Always empty
exitStatus (%d)	UNIX exit status of the task
maxNumProcessors (%d)	Always 1
loginShell (%s)	Always empty
timeEvent (%s)	Always empty
idx (%d)	Session Job Index
maxRMem (%d)	Always 0
maxRSwap (%d)	Always 0
inFileSpool (%s)	Always empty
commandSpool (%s)	Always empty
rsvId (%s)	Always empty
sla (%s)	Always empty
exceptMask (%d)	Always 0
additionalInfo (%s)	Always empty
exitInfo (%d)	Always 0
warningAction (%s)	Always empty
warningTimePeriod (%d)	Always 0
chargedSAAP (%s)	Always empty
licenseProject (%s)	Always empty

Field	Description
options3 (%d)	Always 0
app (%s)	Always empty
taskID (%d)	Task ID
taskIdx (%d)	Task index
taskName (%s)	Task name
taskOptions (%d)	Bit mask of task options: • TASK_IN_FILE (0x01)—specify input file • TASK_OUT_FILE (0x02)—specify output file • TASK_ERR_FILE (0x04)—specify error file • TASK_PRE_EXEC (0x08)—specify pre-exec command • TASK_POST_EXEC (0x10)—specify post-exec command • TASK_NAME (0x20)—specify task name
taskExitReason (%d)	Task exit reason: • TASK_EXIT_NORMAL = 0— normal exit • TASK_EXIT_INIT = 1—generic task initialization failure • TASK_EXIT_PATH = 2—failed to initialize path • TASK_EXIT_NO_FILE = 3—failed to create task file • TASK_EXIT_PRE_EXEC = 4— task pre-exec failed • TASK_EXIT_NO_PROCESS = 5—fork failed • TASK_EXIT_XDR = 6—xdr communication error • TASK_EXIT_NOMEM = 7— no memory • TASK_EXIT_SYS = 8—system call failed • TASK_EXIT_TSCHILD_EXEC = 9—failed to run sschild • TASK_EXIT_RUNLIMIT = 10—task reaches run limit • TASK_EXIT_IO = 11—I/O failure • TASK_EXIT_RSRC_LIMIT = 12—set task resource limit failed

Running and monitoring Session Scheduler jobs

Create a Session Scheduler session and run tasks

1. Create task definition file.

   For example:

   ```
   cat my.tasks
   sleep 10
   hostname
   uname
   ls
   ```

2. Use `bsub` with the ssched application profile to submit a Session Scheduler job with the task definition.

   ```
   bsub -app ssched bsub_options ssched [task_options] [-tasks task_definition_file] [command [arguments]]
   ```

   For example:

   ```
   bsub -app ssched ssched -tasks my.tasks
   ```

When all tasks finish, Session Scheduler exits, all temporary files are deleted, the session job is cleaned from the system, and Session Scheduler output is captured and included in the standard LSF job e-mail.

You can also submit a Session Scheduler job without a task definition file to specify a single task.

Note:

The submission directory path can contain up to 4094 characters.

See the ssched command reference for detailed information about all task options.

Submit a Session Scheduler job as a parallel LSF job

1. Use the -n option of `bsub` to submit a Session Scheduler job as a parallel LSF job.

   ```
   bsub -app ssched -n num_hosts ssched [task_options] [-tasks task_definition_file] [command [arguments]]
   ```

   For example:

   ```
   bsub -app ssched -n 2 ssched -tasks my.tasks
   ```

Submit task array jobs

1. Use the -J option to submit a task array via the command line, and no task definition file is needed:

   ```
   -J task_name[index_list]
   ```

   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 (1) is assumed. The task array index starts at one (1).

All tasks in the array share the same option parameters. Each element of the array is distinguished by its array index.

See the ssched command reference for detailed information about all task options.

Submit tasks with automatic task requeue

1. Use the -Q option to specify requeue exit values for the tasks:

   ```
   -Q "exit_code ..."
   ```

   -Q enables automatic task requeue and sets the LSB_EXIT_REQUEUE environment variable. Use spaces to separate multiple exit codes. LSF does not save the output from the failed task, and does not notify the user that the task failed.

   If a job is killed by a signal, the exit value is 128+*signal_value*. Use the sum of 128 and the signal value as the exit code in the parameter. For example, if you want a task to rerun if it is killed with a signal 9 (SIGKILL), the exit value is 128+9=137.

   The SSCHED_REQUEUE_LIMIT setting limits the number of times a task can be requeued.

   See the ssched command reference for detailed information about all task options.

Monitor Session Scheduler jobs

1. Run `bjobs -ss` to get summary information for Session Scheduler jobs and tasks.

   ```
   JOBID OWNER JOB_NAME NTASKS PEND DONE  RUN EXIT
   1   lsfadmin job1   10     4    4    2    0
   2   lsfadmin job2   10    10    0    0    0
   3   lsfadmin job3   10    10    0    0    0
   ```

   Information displays about your session scheduler job, including Job ID, the owner, the job name, the number of total tasks, and the number of tasks in any of the following states: pend, run, done, exit.
2. Use `bjobs -l -ss` or `bread` to track the progress of the Session Scheduler job.

Kill a Session Scheduler session

1. Use `bkill` to kill the Session Scheduler session. All temporary files are deleted, and the session job is cleaned from the system.

Check your job submission

1. Use the -C option to sanity-check all parameters and the task definition file.

   `ssched` exits after the check is complete. An exit code of 0 indicates no errors were found. A non-zero exit code indicates errors. You can run `ssched -C` outside of LSF.

   See the ssched command reference for detailed information about all task options.

Example output of `ssched -C`:

ssched -C -tasks my.tasks

```
Error in tasks file line 1: -XXX 123 sleep 0

Unsupported option: -XXX

Error in tasks file line 2: -o my.out

A command must be specified
```

Only the `ssched` parameters are checked, not the `ssched` task command itself. The task command must exist and be executable. `ssched -C` cannot detect whether the task command exists or is executable. To check a task definitions file, remember to specify the -tasks option.

Enable recoverable Session Scheduler sessions

By default, Session Scheduler sessions are unrecoverable. In the event of a system crash, the session job must be resubmitted and all tasks are resubmitted and rerun.

However, the Session Scheduler supports application-level checkpoint/restart using Platform LSF's existing facilities. If the user specifies a checkpoint directory when submitting the session job, the job can be restarted using `brestart`. After a restart, only those tasks that have not yet completed are resubmitted and run.

1. To enable recoverable sessions, when submitting the session job:
   a) Provide a writable directory on a shared file system.
   b) Specify the `ssched` checkpoint method with the `bsub -k` option.

You do not need to call `bchkpnt`. The Session Scheduler automatically checkpoints itself after each task completes.

For example:

```
bsub -app ssched -k "/share/scratch method=ssched" -n 8
ssched -tasks simpton.tasks

Job <123> is submitted to default queue <normal>.

...

brestart /share/scratch 123
```

Resizable jobs

Enabling resizable jobs allows Session Scheduler to run jobs with minimum and maximum slots requested and dynamically use the number of slots available at any given time.

Session Scheduler automatically releases idle resources for all resizable jobs. The typical use case is a "long tail" scenario, where all short running tasks complete within one session except a few long running tasks. Those long running tasks occupy a small number of hosts, leaving most of the originally allocated resources idle. Session Scheduler automatically detects those idle resources, shuts down the execution agents running on those hosts and releases resources back to LSF.

When additional resources are added to Session Scheduler, it recognizes those resources and makes use of them to run tasks.

Resizable Job

A job whose job slot allocation can grow and shrink during its run time. The allocation change request may be triggered automatically or by the `bresize` command.

When users run the `bresize release` command to forcibly release resources from Session Scheduler, it recognizes those released resources and shuts down the execution agents running on those hosts.

Autoresizable job

A resizable job with a minimum and maximum slot request. LSF automatically schedules and allocates additional resources to satisfy job maximum request as the job runs.

Configuring resizable jobs

Session Scheduler jobs are resizable when the parameter **RESIZABLE_JOBS=Y** in the ssched application profile.

Session Scheduler jobs are autoresizable when the parameter **RESIZABLE_JOBS=AUTO** in the ssched application profile, or when **RESIZABLE_JOBS=Y** in the ssched application profile and jobs are submitted with the `bjobs` option -ar.

Session Scheduler jobs are not resizable when the parameter RESIZABLE_JOBS in the ssched application profile is commented out.

When the bresize release command is run on Session Scheduler jobs, the parameter **DJOB_RESIZE_GRACE_PERIOD=*seconds*** in the ssched application profile configures a time interval for Session Scheduler to react and take necessary actions such as shutting down execution agents.

Examples

Adding new resources

For an autoresizable job, new resources are added in when they become available. Consider the example where **RESIZABLE_JOBS=AUTO** in the ssched application profile:

```
bsub -app ssched -n 1,3 ssched -tasks ./my.tasks
```

The autoresizable job is requesting 3 hosts and only 1 is available. The job starts running on 1 host and pends on 2 more. When the additional hosts are free, the job allocation changes to 3 hosts automatically. Selected output from `bhist`:

```
Dispatched to <host1>;
```

```
Starting (Pid 24721);
```

```
"Tasks:PEND=3 RUN=0 DONE=0 EXIT=0"
```

```
"Tasks:PEND=2 RUN=1 DONE=0 EXIT=0"
```

```
"Tasks:PEND=2 RUN=1 DONE=0 EXIT=0"
```

```
Additional allocation on 1 Hosts/Processors <host2>
```

```
Resize notification acceptedExternal Message "Tasks:PEND=2 RUN=1 DONE=0
EXIT=0"
```

```
Additional allocation on 1 Hosts/Processors <host03>Resize notification
accepted
```

```
External Message "Tasks:PEND=1 RUN=2 DONE=0 EXIT=0"
```

```
External Message "Tasks:PEND=0 RUN=3 DONE=0 EXIT=0"
```

Releasing resources

Idle resources are released for both resizable and autoresizable jobs. For example, **RESIZABLE_JOBS=Y** in the ssched application profile and an autoresizable job is submitted:

bsub -ar -app ssched -n 1,3 ssched -tasks ./my.longtail

The autoresizable job is requesting 1 to 3 hosts and 3 are available so the job starts running on 3 right away. This longtail job has many short tasks and only a few long ones, and is soon finished running on 2 of the 3 hosts. Session Scheduler sees the 2 idle hosts and releases them from the allocation. Selected output from `bhist`:

```
Submitted from host <host11>, to Queue <normal>, CWD <$HOME>, 3 Processors
Requested
```

```
Dispatched to 3 Hosts/Processors <host01> <host02> <host03>;
```

```
"Tasks:PEND=3 RUN=0 DONE=0 EXIT=0"
```

```
"Tasks:PEND=0 RUN=3 DONE=0 EXIT=0"
```

After two tasks are done, Session Scheduler releases two hosts:

```
Release allocation on 2 Hosts/Processors <host02> <host03> by user or
administrator <user01>, Cancel pending allocation request;
```

```
Resize notification accepted;
```

```
"Tasks:PEND=0 RUN=1 DONE=2 EXIT=0"
```

Releasing resources using bresize

Resources can be released on demand for both resizable and autoresizable jobs. For example, **DJOB_RESIZE_GRACE_PERIOD=30** in **RESIZABLE_JOBS=Y** in the ssched application profile and a resizable job is submitted:

bsub -app ssched -n 1,3 ssched -tasks ./my.longtail

```
Job <5> is submitted to the default queue <normal>
```

The resizable job starts running on host02, host03, and host04. The following command releases host02 from the job:

bresize release "host02" 5

Session Scheduler shuts down the execution agent on host02. The job continues to run on host03 and host04.

Selected output from `bhist`:

```
Submitted from host <delpe07.lsf.platform.com>, to Queue <normal>, CWD <$HOME>,
3 Processors Requested;
```

```
Dispatched to 3 Hosts/Processors <host02> <host03> <host04>;
```

```
"Tasks:PEND=3 RUN=0 DONE=0 EXIT=0"
```

```
"Tasks:PEND=0 RUN=3 DONE=0 EXIT=0"
```

```
Release allocation on 1 Hosts/Processors <host02> by user or administrator
<user01>
```

```
Resize notification accepted
```

```
"Tasks:PEND=1 RUN=2 DONE=0 EXIT=0"
```

Troubleshooting

Use any of the following methods to troubleshoot your Session Scheduler jobs.

ssched environment variables

Before submitting the `ssched` command, You can set the following environment variables to enable additional debugging information:

SSCHED_DEBUG_LOG_MASK=[LOG_INFO | LOG_DEBUG | LOG_DEBUG1 | ...]	Controls the amount of logging
SSCHED_DEBUG_CLASS=ALL or SSCHED_DEBUG_CLASS=[LC_TRACE] [LC_FILE] [...]	• Filters out some log classes, or shows all log classes • By default, no log classes are shown
SSCHED_DEBUG_MODULES=ALL or SSCHED_DEBUG_MODULES=[ssched] [libvem.so] [sservice] [sschild]	• Enables logging on some or all components • By default, logging is disabled on all components • libvem.so controls logging by the `libvem.so` loaded by the SD, SSM and `ssched` • Enabling debugging of the Session Scheduler automatically enables logging by the `libvem.so` loaded by the Session Scheduler
SSCHED_DEBUG_REMOTE_HOSTS=ALL or SSCHED_DEBUG_REMOTE_HOSTS=[hostname1] [hostname2] [...]	• Enables logging on some/all hosts • By default, logging is disabled on all remote hosts
SSCHED_DEBUG_REMOTE_FILE=Y	• Directs logging to `/tmp/ssched/`*`job_ID`*`.`*`job_index`*`/` instead of `stderr` on each remote host • Useful if too much debugging info is slowing down the network connection • By default, debugging info is sent to `stderr`

ssched debug options

The `ssched` options -1, -2, and -3 are shortcuts for the following environment variables.

ssched -1	Is a shortcut for: • SSCHED_DEBUG_LOG_MASK=LOG_WARNING • SSCHED_DEBUG_CLASS=ALL • SSCHED_DEBUG_MODULES=ALL
ssched -2	Is a shortcut for: • SSCHED_DEBUG_LOG_MASK=LOG_INFO • SSCHED_DEBUG_CLASS=ALL • SSCHED_DEBUG_MODULES=ALL
ssched -3	Is a shortcut for:

- SSCHED_DEBUG_LOG_MASK=LOG_DEBUG
- SSCHED_DEBUG_CLASS=ALL
- SSCHED_DEBUG_MODULES=ALL

Example output of ssched -2:

```
Nov 20 14:35:01 2008 4546 6 7.05 SSCHED_UPDATE_SUMMARY_INTERVAL = 1

Nov 20 14:35:01 2008 4546 6 7.05 SSCHED_UPDATE_SUMMARY_BY_TASK = 0

Nov 20 14:35:01 2008 4546 6 7.05 SSCHED_REQUEUE_LIMIT = 1

Nov 20 14:35:01 2008 4546 6 7.05 SSCHED_RETRY_LIMIT = 1

Nov 20 14:35:01 2008 4546 6 7.05 SSCHED_MAX_TASKS = 10

Nov 20 14:35:01 2008 4546 6 7.05 SSCHED_MAX_RUNLIMIT = 600

Nov 20 14:35:01 2008 4546 6 7.05 SSCHED_ACCT_DIR = /home/user1/ssched

Nov 20 14:35:03 2008 4546 6 7.05 Task <1[1]> submitted. Command <sleep 0>;

Nov 20 14:35:03 2008 4546 6 7.05 Task <1[2]> submitted. Command <sleep 0>;

Nov 20 14:35:03 2008 4546 6 7.05 Task <1[3]> submitted. Command <sleep 0>;

Nov 20 14:35:03 2008 4546 6 7.05 Task <1[4]> submitted. Command <sleep 0>;

Nov 20 14:35:03 2008 4546 6 7.05 Task <1[5]> submitted. Command <sleep 0>;

Nov 20 14:35:05 2008 4546 6 7.05 Task <1[1]> done successfully. The CPU time used is 0.030993
seconds;

Nov 20 14:35:05 2008 4546 6 7.05 Task <1[2]> done successfully. The CPU time used is 0.039992
seconds;

Nov 20 14:35:05 2008 4546 6 7.05 Task <1[3]> done successfully. The CPU time used is 0.033993
seconds;

Nov 20 14:35:05 2008 4546 6 7.05 Task <1[4]> done successfully. The CPU time used is 0.026994
seconds;

Nov 20 14:35:05 2008 4546 6 7.05 Task <1[5]> done successfully. The CPU time used is 0.036992
seconds;


Task Summary

Submitted:                    5

Done:                         5
```

Example output of ssched -2 with requeue

```
Nov 20 14:35:47 2008 4748 6 7.05 SSCHED_UPDATE_SUMMARY_INTERVAL = 1
Nov 20 14:35:47 2008 4748 6 7.05 SSCHED_UPDATE_SUMMARY_BY_TASK = 0
Nov 20 14:35:47 2008 4748 6 7.05 SSCHED_REQUEUE_LIMIT = 1
Nov 20 14:35:47 2008 4748 6 7.05 SSCHED_RETRY_LIMIT = 1
Nov 20 14:35:47 2008 4748 6 7.05 SSCHED_MAX_TASKS = 10
Nov 20 14:35:47 2008 4748 6 7.05 SSCHED_MAX_RUNLIMIT = 600
Nov 20 14:35:47 2008 4748 6 7.05 SSCHED_ACCT_DIR = /home/user1/ssched
Nov 20 14:35:49 2008 4748 6 7.05 Task <1> submitted. Command <exit 1>;
Nov 20 14:35:50 2008 4748 6 7.05 Task <1> exited with status 1.
Nov 20 14:35:50 2008 4748 6 7.05 Task <1> submitted. Command <exit 1>;
Nov 20 14:35:50 2008 4748 6 7.05 Task <1> exited with status 1.

Task Summary
Submitted:                  1
Requeued:                   1

Done:                       0
Exited:                     2
  Execution Errors: 2
  Dispatch Errors:  0
  Other Errors:     0

Task Error Summary

Execution Error
Task ID:                 1
Submit Time:             Tue Nov 20 14:35:49 2008
Start Time:              Tue Nov 20 14:35:50 2008
End Time:                Tue Nov 20 14:35:50 2008
Exit Code:               1
Exit Reason:             Normal exit
Exec Hosts:              ibm03
Exec Home:               /home/user1
Exec Dir:                /home/user1/src/lsf7ss/ssched/ssched
Command:                 exit 1
Action:                  Requeue exit value match; task will be requeued

Execution Error
Task ID:                 1
Submit Time:             Tue Nov 20 14:35:50 2008
Start Time:              Tue Nov 20 14:35:50 2008
End Time:                Tue Nov 20 14:35:50 2008
Exit Code:               1
Exit Reason:             Normal exit
Exec Hosts:              ibm03
Exec Home:               /home/user1
Exec Dir:                /home/user1/src/lsf7ss/ssched/ssched
Command:                 exit 1
Action:                  Task requeue limit reached; task will not be requeued
```

Example output of ssched -2 with retry

```
Nov 20 14:37:04 2008 5049 6 7.05 SSCHED_UPDATE_SUMMARY_INTERVAL = 1
Nov 20 14:37:04 2008 5049 6 7.05 SSCHED_UPDATE_SUMMARY_BY_TASK = 0
Nov 20 14:37:04 2008 5049 6 7.05 SSCHED_REQUEUE_LIMIT = 1
Nov 20 14:37:04 2008 5049 6 7.05 SSCHED_RETRY_LIMIT = 1
Nov 20 14:37:04 2008 5049 6 7.05 SSCHED_MAX_TASKS = 10
Nov 20 14:37:04 2008 5049 6 7.05 SSCHED_MAX_RUNLIMIT = 600
Nov 20 14:37:04 2008 5049 6 7.05 SSCHED_ACCT_DIR = /home/user1/ssched
Nov 20 14:37:06 2008 5049 6 7.05 Task <1> submitted. Command <sleep 0>;
Nov 20 14:37:08 2008 5049 6 7.05 Task <1> had a dispatch error.
Nov 20 14:37:08 2008 5049 6 7.05 Task <1> submitted. Command <sleep 0>;
Nov 20 14:37:08 2008 5049 6 7.05 Task <1> had a dispatch error.

Task Summary
Submitted:                  1
Done:                       0
Exited:                     1
  Execution Errors: 0
  Dispatch Errors:  1
  Other Errors:     0

Task Error Summary

Dispatch Error
Task ID:                 1
Submit Time:             Tue Nov 20 14:37:06 2008
Failure Reason:          Pre-execution command failed
Command:                 sleep 0
Pre-Exec:                exit 1
Start time:              Tue Nov 20 14:37:07 2008
Execution host:          ibm03
Action:                  Task will be retried

Dispatch Error
Task ID:                 1
Submit Time:             Tue Nov 20 14:37:08 2008
Failure Reason:          Pre-execution command failed
Command:                 sleep 0
Pre-Exec:                exit 1
Start time:              Tue Nov 20 14:37:08 2008
Execution host:          ibm03
Action:                  Task retry limit reached; task will not be retried
```

Note:

The "Task Summary" and "Summary of Errors" sections are sent to `stdout`. All other output is sent to `stderr`.

Send SIGUSR1 signal

After the tasks have been submitted to the Session Scheduler and started, users can enable additional debugging by Session Scheduler components by sending a SIGUSR1 signal.

To enable additional debugging by the `ssched` and `libvem` components, send a SIGUSR1 to the ssched_real process. This enables the following:

- SSCHED_DEBUG_LOG_MASK=LOG_DEBUG
- SSCHED_DEBUG_CLASS=ALL
- SSCHED_DEBUG_MODULES=ALL

The additional log messages are sent to `stderr`.

To enable additional debugging by the `sservice` and `sschild` components, send a SIGUSR1 on the remote host to the `sservice` process. This enables the following:

- SSCHED_DEBUG_LOG_MASK=LOG_DEBUG
- SSCHED_DEBUG_CLASS=ALL
- SSCHED_DEBUG_MODULES=ALL
- SSCHED_DEBUG_REMOTE_HOSTS=ALL
- SSCHED_DEBUG_REMOTE_FILE=Y

The debug messages are saved to a file in `/tmp/ssched/`. You are responsible for deleting this file when it is no longer needed.

Send SIGUSR2 signal

If a SIGUSR1 signal is sent, SIGUSR2 restores debugging to its original level.

Known issues and limitations

General issues

- The Session Scheduler caches host info from LIM. If the host factor of a host is changed after the Session Scheduler starts, the Session Scheduler will not see the updated host factor. The host factor is used in the task accounting log.
- Session Scheduler does not support per task memory or swap utilization tracking from `ssacct`. Run `bacct` to see aggregate memory and swap utilization.
- When specifying a multiline command line as a `ssched` command line parameter, you must enclose the command in quotes. A multiline command line is any command containing a semi-colon (`;`). For example:

  ```
  ssched -o my.out "hostname; ls"
  ```

  When specifying a multiline command line as a parameter in a task definition file, you must NOT use quotes. For example:

  ```
  cat my.tasks
  ```

  ```
  -o my.out hostname; ls
  ```
- If you submit a shell script containing multiple `ssched` commands, `bjobs -l` only shows the task summary for the currently running `ssched` instance. Enable task accounting and

examine the accounting file to see information for tasks from all `ssched` instances in the shell script.

- Submitting a large number of tasks as part of one session may cause a slight delay between when the Session Scheduler starts and when tasks are dispatched to execution agents. The Session Scheduler must parse and submit each task before it begins dispatching any tasks. Parsing 50,000 tasks can take up to 2 minutes before dispatching starts.
- After all tasks have completed, the Session Scheduler will take some time to terminate all execution agents and to clean up temporary files. A minimum of 20 seconds is normal, longer for larger allocations.
- Session Scheduler handles the following signals: SIGINT, SIGTERM, SIGUSR1, SIGSTOP, SIGTSTP, and SIGCONT. All other signals cause `ssched` to exit immediately. No summary is output and task accounting information is not saved. The signals Session Scheduler handles will be expanded in future releases.

CHAPTER

3

Platform LSF Session Scheduler Reference

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

Summary (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.

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
```

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 2007: Submitted from host <hostA>;
Thu Nov  1 13:48:05 2007: Dispatched to <hostA>, Execution CWD </home/user1/src
Thu Nov  1 13:48:06 2007: 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]
```

Options list

Command options

-1 | **-2** | **-3**

-C

-p

Task options

-E "*`pre_exec_command [argument ...]`***"**

-Ep "*`post_exec_command [argument ...]`***"**

-e *`err_file`*

-i *`input_file`*

-J *`task_name`***[***`index_list`***]**

-j "*`starter [starter]`* `[`**%USRCMD**`]` *`[starter]`***"**

-M *`mem_limit`*

-o *`out_file`*

-Q "*`exit_code ...`***"**

-W `[`*`minutes`*`:]`*`seconds`*

-h

-V

Option descriptions

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

Environment variables

By default, all environment variables that are set as part of the session are available in each tasks's execution environment.

See the *Platform LSF Configuration Reference* for more information about LSF environment variables.

Variables for the execution host of each task

The following environment variables are reset according to the execution host of each task:

- EGO_SERVERDIR
- LSB_TRAPSIGS
- LSF_SERVERDIR
- HOSTTYPE
- LSB_HOSTS
- LSF_BINDIR
- EGO_BINDIR
- PWD
- HOME
- LSB_ERRORFILE
- LSB_OUTPUTFILE
- TMPDIR
- LSF_LIBDIR
- EGO_LIBDIR
- LSB_MCPU_HOSTS
- PATH (prepend LSF_BINDIR)
- LD_LIBRARY_PATH (prepend LSF_LIBDIR and EGO_LIBDIR)

Environment variables NOT available in the task environment

- LSB_JOBRES_PID
- LSB_EEXEC_REAL_UID
- LS_EXEC_T
- LSB_INTERACTIVE
- LSB_CHKFILENAME
- SPOOLDIR
- LSB_ACCT_FILE
- LSB_EEXEC_REAL_GID
- LSB_CHKPNT_DIR
- LSB_CHKPNT_PERIOD
- LSB_JOB_STARTER
- LSB_EXIT_REQUEUE
- LSB_DJOB_RU_INTERVAL
- LSB_DJOB_HB_INTERVAL
- LSB_DJOB_HOSTFILE
- LSB_JOBEXIT_INFO
- LSB_JOBPEND
- LSB_EXECHOSTS

Environment variables corresponding to the session job

- LSB_JOBID
- LSB_JOBINDEX
- LSB_JOBINDEX_STEP
- LSB_JOBINDEX_END
- LSB_JOBPID
- LSB_JOBNAME
- LSB_JOBFILENAME

Environment variables set individually for each task

- LSB_TASKID—The current task ID
- LSB_TASKINDEX—The current task index

lsb.params

The `lsb.params` file defines parameters used by the Session Scheduler system. This file contains only one section, named Parameters. The file is optional. If not present, the LSF-defined defaults are assumed.

Note:

This chapter describes only the subset of parameters used by Session Scheduler for Platform LSF 7. See the *Platform LSF Configuration Reference* for all `lsb.params` parameters.

This file is installed by default in `LSB_CONFDIR/`*cluster_name*`/configdir`.

Changing lsb.params configuration

After making any changes to `lsb.params`, run `badmin reconfig` to reconfigure mbatchd.

Parameters Section

This section and all the keywords in this section are optional. If keywords are not present, the default values are assumed.

Session Scheduler Parameters

- SSCHED_ACCT_DIR
- SSCHED_MAX_RUNLIMIT
- SSCHED_MAX_TASKS
- SSCHED_REQUEUE_LIMIT
- SSCHED_RETRY_LIMIT
- SSCHED_UPDATE_SUMMARY_BY_TASK
- SSCHED_UPDATE_SUMMARY_INTERVAL

SSCHED_ACCT_DIR

Syntax

SSCHED_ACCT_DIR=*directory*

Description

A universally accessible and writable directory that will store Session Scheduler task accounting files. Each Session Scheduler session (each `ssched` instance) creates one accounting file. Each file contains one accounting entry for each task. The accounting file is named *`job_ID`*`.ssched.acct`. If no directory is specified, accounting records are not written.

Valid values

Specify any string up to 4096 characters long

Default

Not defined. No task accounting file is created.

SSCHED_MAX_RUNLIMIT

Syntax

SSCHED_MAX_RUNLIMIT=*seconds*

Description

Maximum run time for a task. Users can override this value with a lower value. Specify a value greater than or equal to zero (0).

Recommended value

For very short-running tasks, a reasonable value is twice the typical runtime. Because LSF does not release slots allocated to the session until all tasks are completed and `ssched` exits, you should avoid setting a large value for SSCHED_MAX_RUNLIMIT.

Valid values

Specify a positive integer between 0 and 2147483645

Default

600 seconds (10 minutes)

SSCHED_MAX_TASKS

Syntax

SSCHED_MAX_TASKS=*integer*

Description

Maximum number of tasks that can be submitted to Session Scheduler. Session Scheduler exits if this limit is reached. Specify a value greater than or equal to zero (0).

Valid values

Specify a positive integer between 0 and 2147483645

Default

50000 tasks

SSCHED_REQUEUE_LIMIT

Syntax

SSCHED_REQUEUE_LIMIT=*integer*

Description

Number of times Session Scheduler tries to requeue a task as a result of the REQUEUE_EXIT_VALUES (`ssched -Q`) setting. SSCHED_REQUEUE_LIMIT=0 means never requeue. Specify a value greater than or equal to zero (0).

Valid values

Specify a positive integer between 0 and 2147483645

Default

3 requeue attempts

SSCHED_RETRY_LIMIT

Syntax

SSCHED_RETRY_LIMIT=*integer*

Description

Number of times Session Scheduler tries to retry a task that fails during dispatch or setup. SSCHED_RETRY_LIMIT=0 means never retry. Specify a value greater than or equal to zero (0).

Valid values

Specify a positive integer between 0 and 2147483645

Default

3 retry attempts

SSCHED_UPDATE_SUMMARY_BY_TASK

Syntax

SSCHED_UPDATE_SUMMARY_INTERVAL=*integer*

Description

Update the Session Scheduler task summary via `bpost` after the specified number of tasks finish. Specify a value greater than or equal to zero (0).

If both SSCHED_UPDATE_SUMMARY_INTERVAL and SSCHED_UPDATE_SUMMARY_BY_TASK are set to zero (0), `bpost` is not run.

Valid values

Specify a positive integer between 0 and 2147483645

Default

0

See also

SSCHED_UPDATE_SUMMARY_INTERVAL

SSCHED_UPDATE_SUMMARY_INTERVAL

Syntax

SSCHED_UPDATE_SUMMARY_INTERVAL=*seconds*

Description

Update the Session Scheduler task summary via `bpost` after the specified number of seconds. Specify a value greater than or equal to zero (0).

If both SSCHED_UPDATE_SUMMARY_INTERVAL and SSCHED_UPDATE_SUMMARY_BY_TASK are set to zero (0), `bpost` is not run.

Valid values

Specify a positive integer between 0 and 2147483645

Default

60 seconds

See also

SSCHED_UPDATE_SUMMARY_BY_TASK

See Also

ssacct, ssched

Index