Inside Platform Lava

Version 1.0
September 2007
Comments to: doc@platform.com
## Contents

1. **Inside Your Cluster**
   - Cluster Characteristics ........................................... 6
   - Restarting and Reconfiguring Lava Daemons ..................... 8
   - Cluster Administrators ........................................... 12
   - Mail Notification ................................................. 13
   - Managing Users, Hosts, and Queues .............................. 14
   - Error and Event Logging .......................................... 23
   - Monitoring Your Cluster .......................................... 25

2. **Working with Resources and Resource Requirements**
   - Resource Classifications .......................................... 30
   - Configuring Your Own Resources .................................. 36
   - External Load Indices and ELIM .................................. 39
   - Configuring Resource Requirements ............................... 42
   - Resource Requirement Strings ................................... 43
   - Monitoring Resources ............................................. 46

3. **Configuring Job Controls**
   - Configuring Resource Usage Limits ............................... 50
   - Configuring Load Thresholds ..................................... 53
   - Configuring Job Control Actions ................................ 55
   - Configuring Pre-Execution and Post-Execution Commands ..... 58
   - Configuring Job Starters for Queues ............................. 60

Index .................................................................................. 63
CHAPTER 1

Inside Your Cluster

Contents

- “Cluster Characteristics” on page 6
- “Restarting and Reconfiguring Lava Daemons” on page 8
- “Cluster Administrators” on page 12
- “Mail Notification” on page 13
- “Managing Users, Hosts, and Queues” on page 14
- “Error and Event Logging” on page 23
- “Monitoring Your Cluster” on page 25
Cluster Characteristics

The Lava master host daemons

The Lava master host is a Lava server host that acts as the overall coordinator for the cluster. All Lava daemons run on the master host. The lim on the master host is the master LIM. The master host is installed on the front-end node (frontend-0).

- **master LIM**: LIM on the master host. The master host is installed on the front-end node.
- **mbatchd**: Master Batch Daemon running on the master host (the front-end node). Started by the slave batch daemon, sbatchd. Responsible for the overall state of jobs in the system. Receives job submission, and information query requests. Manages jobs held in queues. Dispatches jobs to hosts as determined by mbschd.
- **mbschd**: Master Batch Scheduler Daemon running on the master host. Works with mbatchd. Started by mbatchd. Makes scheduling decisions based on job requirements and policies.

Default Lava directories

The following directories are owned by the primary Lava administrator and are readable by all cluster users.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSF_CONFDIR</td>
<td>Lava configuration directory</td>
<td>/opt/lava/conf/</td>
</tr>
<tr>
<td>LSB_CONFDIR</td>
<td>Lava batch configuration directory</td>
<td>/opt/lava/conf/lsbatch/</td>
</tr>
<tr>
<td>LSB_SHAREDIR</td>
<td>Lava batch job history directory</td>
<td>/opt/lava/work/</td>
</tr>
<tr>
<td>LSF_LOGDIR</td>
<td>Server daemon error logs, one for each Lava daemon</td>
<td>/opt/lava/log/</td>
</tr>
</tbody>
</table>

Four important Lava configuration files

Lava configuration is administered through several configuration files, which you use to modify the behavior of your cluster. The four most important files you will work with are the following files, which are installed on the master host (front-end node):

- LSF_CONFDIR/lsf.conf
- LSF_CONFDIR/lsf.cluster.lava
- LSF_CONFDIR/lsf.shared
- LSB_CONFDIR/lava/configdir/lsb.queues

These files are created on the front-end node during the Lava installation.

All the files are owned by root. The files are readable by all cluster users. You can change ownership of all the files to the Lava administrator.

**lsf.conf**

The most important file in Lava. It contains the paths to the Lava configuration directories, log directories, libraries, and other global configuration information. A version of lsf.conf file is also installed on each compute host. It shows the location of the log directory and conf directory on the master host (front-end node).
lsf.cluster.lava

Defines the host name, model, and type of the master host (on the front-end node). It also defines the user name of the Lava administrator.

lsf.shared

This file is like a dictionary that defines all the keywords used by the Lava cluster. You can add your own keywords to specify the names of resources or host types.

Note that LSF_SERVERDIR is not a shared directory.

lsb.queues

Defines the Lava batch queues and their parameters for one Lava cluster.

Cluster name

The name of the cluster is lava. This name is part of the name of the /opt/lava/conf/lsf.cluster.lava file:

Lava hosts

- The Lava master host is configured in the Hosts section of LSF_CONFDIR/lsf.cluster.lava.
- The master host on the front-end node is dynamically configured as a Lava server host. This is indicated by 1 in the server column of the Hosts section of LSF_CONFDIR/lsf.cluster.lava.
- Host types installed in your cluster are listed in the Hosts section of LSF_CONFDIR/lsf.cluster.lava. The master host is configured by default. You can also add your compute hosts to this section.

Before you configure your resources, you must add your compute hosts to the Hosts section of LSF_CONFDIR/lsf.cluster.lava.
Restarting and Reconfiguring Lava Daemons

Restarting the whole cluster

Lava starts automatically in your Platform OCS cluster. If you need to restart your cluster, you must restart both the master host (on the front-end node) and all the compute hosts.

To change configuration on the master host, you do not need to restart the whole cluster. You can simply restart the Lava daemons on the master host. (See “Reconfiguring the cluster” on page 10.)

When you restart the cluster, you must restart Lava individually on pvfs-io and compute-pvfs hosts.

To restart the cluster:
1. Log on to the master host (on the front-end node) as root:
2. Restart Lava on the master host (front-end node):
   ```
   # /etc/init.d/lava stop
   # /etc/init.d/lava start
   ```
3. Restart Lava on the compute hosts:
   ```
   # cluster-fork /etc/init.d/lava stop
   # cluster-fork /etc/init.d/lava start
   ```
4. Restart Lava on individual hosts, such as pvfs-io and compute-pvfs hosts:
   ```
   # ssh hostname /etc/init.d/lava stop
   # ssh hostname /etc/init.d/lava start
   ```
   For example:
   ```
   # ssh compute-pvfs-0 /etc/init.d/lava stop
   # ssh compute-pvfs-0 /etc/init.d/lava start
   ```

Restarting the master host

If you need to restart the Lava daemons on the master host without restarting the whole cluster, run the following commands:

1. Run `badmin hshutdown` to shut down the slave batch daemon (`sbatchd`) on the master host. For example:
   ```
   # badmin hshutdown frontend-0
   ```
2. Restart `mbatchd`:
   ```
   # badmin reconfig
   ```
   This causes `mbatchd` and `mbschd` to exit. The `mbatchd` cannot be restarted, because `sbatchd` is shut down. All Lava services are temporarily unavailable, but existing jobs are not affected. When `mbatchd` is later started by `sbatchd`, its previous status is restored from the event log file, and job scheduling continues.
Reconfiguring the master host

If you have edited the configuration files, and do not need to recognize new hosts or remove hosts, you can reload the configuration files without restarting the cluster:

```bash
# badmin reconfig
```

For more information on changing your configuration, see “Reconfiguring the cluster” on page 10.

Controlling daemons

To control all daemons in the cluster, you must:

- Be logged on as root.
- Be able to run `rsh` or `cluster-fork` commands across all Lava hosts without having to enter a password.
  - `rsh` must be enabled.
  - The shell command specified by `LSF_RSH` in `lsf.conf` is used before `rsh` is tried.

The following is an overview of commands you use to control Lava daemons.

<table>
<thead>
<tr>
<th>Daemon</th>
<th>Action</th>
<th>Command</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbatchd</td>
<td>Start</td>
<td>badmin hstartup [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>Restart</td>
<td>badmin hrestart [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>Shut down</td>
<td>badmin hshutdown [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td>mbatchd</td>
<td>Restart</td>
<td>badmin reconfig</td>
<td>Must be root or the Lava administrator for these commands</td>
</tr>
<tr>
<td>mbschd</td>
<td>Shut down</td>
<td>1  badmin hshutdown</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2  badmin reconfig</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reconfigure</td>
<td>badmin reconfig</td>
<td></td>
</tr>
<tr>
<td>RES</td>
<td>Start</td>
<td>lsadmin resstartup [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>Shut down</td>
<td>lsadmin resshutdown [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>Restart</td>
<td>lsadmin resrestart [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td>LIM</td>
<td>Start</td>
<td>lsadmin limstartup [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>Shut down</td>
<td>lsadmin limshutdown [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>Restart</td>
<td>lsadmin limrestart [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>Restart all</td>
<td>lsadmin limshutdown [host_name ...</td>
<td>all]</td>
</tr>
<tr>
<td></td>
<td>in cluster</td>
<td>lsadmin reconfig</td>
<td></td>
</tr>
<tr>
<td>All Lava</td>
<td>Restart the</td>
<td>lava stop</td>
<td>Must be root</td>
</tr>
<tr>
<td>daemons</td>
<td>whole cluster</td>
<td>lava start</td>
<td></td>
</tr>
</tbody>
</table>

**sbatchd**

Restarting `sbatchd` on a host does not affect jobs that are running on that host. If `sbatchd` is shut down, the host is not available to run new jobs. Existing jobs running on that host continue, but the results are not sent to the user until `sbatchd` is restarted.
Restarting and Reconfiguring Lava Daemons

LIM and RES

Jobs running on the host are not affected by restarting the daemons.

If a daemon is not responding to network connections, `lsadmin` displays an error message with the host name. In this case, you must kill and restart the daemon manually.

If RES is shut down while remote interactive tasks are running on the host, the running tasks continue but no new tasks are accepted.

Reconfiguring the cluster

After changing Lava configuration files, you must tell Lava to reread the files to update the configuration. The commands you can use to reconfigure a cluster are:

- `lsadmin reconfig`
- `badmin reconfig`

The reconfiguration commands you use depend on which files you change in Lava. The following table is a quick reference.

<table>
<thead>
<tr>
<th>After making changes to ...</th>
<th>Use ...</th>
<th>Which ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts</td>
<td>badmin reconfig</td>
<td>reloads configuration files</td>
</tr>
<tr>
<td>lsb.hosts</td>
<td>badmin reconfig</td>
<td>reloads configuration files</td>
</tr>
<tr>
<td>lsb.modules</td>
<td>badmin reconfig</td>
<td>reloads configuration files</td>
</tr>
<tr>
<td>lsb.params</td>
<td>badmin reconfig</td>
<td>reloads configuration files</td>
</tr>
<tr>
<td>lsb.queues</td>
<td>badmin reconfig</td>
<td>reloads configuration files</td>
</tr>
<tr>
<td>lsf.cluster.lava</td>
<td>lsadmin reconfig</td>
<td>reconfigures LIM, reloads configuration files, and restarts mbatchd</td>
</tr>
<tr>
<td></td>
<td>badmin reconfig</td>
<td></td>
</tr>
<tr>
<td>lsf.conf</td>
<td>lsadmin reconfig AND badmin reconfig</td>
<td>reconfigures LIM and reloads configuration files, and restarts mbatchd</td>
</tr>
<tr>
<td>lsf.shared</td>
<td>lsadmin reconfig</td>
<td>reconfigures LIM, reloads configuration files, and restarts mbatchd</td>
</tr>
<tr>
<td>lsf.sudoers</td>
<td>badmin reconfig</td>
<td>reloads configuration files</td>
</tr>
<tr>
<td>lsf.task</td>
<td>lsadmin reconfig AND badmin reconfig</td>
<td>reconfigures LIM and reloads configuration files</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reconfiguring the cluster with lsadmin and badmin

1 Log on to the host as root or the Lava administrator.
2 Run `lsadmin reconfig` to reconfigure LIM:

```
# lsadmin reconfig
Checking configuration files ...
No errors found.
Do you really want to restart LIMs on all hosts? [y/n] y
Restart LIM on <compute-0-0> ...... done
Restart LIM on <compute-0-1> ...... done
Restart LIM on <compute-0-2> ...... done
```

The `lsadmin reconfig` command checks for configuration errors.
If no errors are found, you are asked to confirm that you want to restart lim on all
hosts, and lim is reconfigured. If fatal errors are found, reconfiguration is aborted.

3 Run `badmin reconfig` to reconfigure mbatchd:

```
# badmin reconfig
Checking configuration files ...
No errors found.
Do you want to reconfigure? [y/n] y
Reconfiguration initiated
```

The `badmin reconfig` command checks for configuration errors.
If no fatal errors are found, you are asked to confirm reconfiguration. If fatal errors
are found, reconfiguration is aborted.
Cluster Administrators

**Primary cluster administrator**

Required. The first cluster administrator is specified during installation as root. You can change the primary administrator in the `lsf.cluster.lava` file. The primary Lava administrator account owns the configuration and log files. The primary administrator has permission to perform cluster-wide operations, change configuration files, reconfigure the cluster, and control jobs submitted by all users.

**Cluster administrators**

Optional.

Cluster administrators can perform administrative operations on all jobs and queues in the cluster. Cluster administrators have the same cluster-wide operational privileges as the primary Lava administrator except that they do not have permission to change configuration files.

**Adding cluster administrators**

1. In the `ClusterAdmins` section of `LSF_CONFDIR/lsf.cluster.lava`, specify the list of cluster administrators following `ADMINISTRATORS`, separated by spaces. The first administrator in the list is the primary Lava administrator. All others are cluster administrators. You can specify user names and group names. For example:

   ```
   Begin ClusterAdmins
   ADMINISTRATORS = lavaadmin admin1 admin2
   End ClusterAdmins
   ```

2. Save your changes.
3. Run `lsadmin reconfig` to reconfigure LIM.
4. Run `badmin reconfig` to restart `mbatchd`. 
Mail Notification

When a batch job completes or exits, Lava by default sends a job report by electronic mail to the submitting user account. The report includes the following information:

- Standard output (stdout) of the job
- Standard error (stderr) of the job
- Lava job information such as CPU, process, and memory usage

The output from stdout and stderr are merged together in the order printed, as if the job was run interactively. The default standard input (stdin) file is the null device. The null device on UNIX is /dev/null.

Some batch jobs can create large amounts of output. To prevent large job output files from interfering with your mail system, you can use the LSB_MAILSIZE_LIMIT parameter in lsf.conf to limit the size of the email containing the job output information.

By default, LSB_MAILSIZE_LIMIT is not enabled—no limit is set on size of batch job output email.

If the size of the job output email exceeds LSB_MAILSIZE_LIMIT, the output is saved to a file under JOB_SPOOL_DIR, or the default job output directory if JOB_SPOOL_DIR is undefined. The email informs users where the job output is located.

If the -o option of bsub is used, the size of the job output is not checked against LSB_MAILSIZE_LIMIT.

Lava sets LSB_MAILSIZE to the approximate size in KB of the email containing job output information. LSB_MAILSIZE is not recognized by the Lava default mail program. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to explicitly set the maximum size in KB of the email containing the job information.

For more information on mail notification, see the Platform Lava man pages for information about the LSB_MAILSIZE environment variable and the LSB_MAILTO, LSB_MAILSIZE_LIMIT parameters in lsf.conf, and JOB_SPOOL_DIR in lsb.params.
Managing Users, Hosts, and Queues

Making your cluster available to users

To set up the Lava environment for your users, use the following two shell files:

- LSF_CONFDIR/cshrc.lsf (for csh, tcsh)
- LSF_CONFDIR/profile.lsf (for sh, ksh, or bash)

Make sure all Lava users include one of these files at the end of their own .cshrc or .profile file, or run one of these two files before using Lava.

For csh or tcsh

- Add cshrc.lsf to the end of the .cshrc file for all users:
  - Copy the cshrc.lsf file into .cshrc
  - OR
  - Add a line similar to the following to the end of .cshrc:
    ```
    source /opt/lava/conf/cshrc.lsf
    ```

For sh, ksh, or bash

- Add profile.lsf to the end of the .profile file for all users:
  - Copy the profile.lsf file into .profile
  - OR
  - Add a line similar to following to the end of .profile:
    ```
    /opt/lava/conf/profile.lsf
    ```

Controlling hosts

Hosts are opened and closed by root or a Lava Administrator issuing a command or through configured dispatch windows.

Closing a host

```bash
# badmin hclose compute-0-0
Close <compute-0-0> ...... done
```

If the command fails, it may be because the host is unreachable through network problems, or because the daemons on the host are not running.

Opening a host

```bash
# badmin hopen compute-0-0
Open <hostB> ...... done
```

Dispatch windows

A dispatch window specifies one or more time periods during which a host will receive new jobs.

1. Edit lsb.hosts.
2. Specify on or more time windows in the DISPATCH_WINDOW column.
   For example:
   ```
   Begin Host
   HOST_NAME     r1m      pg    ls     tmp    DISPATCH_WINDOW
   ... hostB 3.5/4.5  15/   12/15  0 (4:30-12:00)
   ... End Host
   ```
3. Reconfigure the cluster:
   a. Run lsadmin reconfig to reconfigure LIM.
   b. Run badmin reconfig to reconfigure mbatchd.
Chapter 1
Inside Your Cluster

Inside Platform Lava

4 Run `bhosts -l` to display the dispatch windows.
For information on dispatch windows for queues, see “Controlling when jobs run” on page 21.

Adding host types and host models to `lsf.shared`:
The `lsf.shared` file contains a list of host type and host model names for most operating systems. You can add to this list or customize the host type and host model names. A host type and host model name can be any alphanumeric string up to 29 characters long.

1 Log on as the Lava administrator to any host in the cluster.
2 Edit `lsf.shared`:
   a For a new host type, modify the `HostType` section:
      ```
      Begin HostType
      TYPENAME # Keyword
      DEFAULT
      LINUX86
      LINUX64
      End HostType
      ```
   b For a new host model, modify the `HostModel` section:
      ```
      Begin HostModel
      MODELNAME  CPUFACTOR   ARCHITECTURE # keyword
      # x86 (Solaris, NT, Linux): approximate values, based on SpecBench results
      # for Intel processors (Sparc/NT) and BogoMIPS results (Linux).
      Opteron848 60.0 (x15_3604_AMDOpterontmProcessor848)
      Intel_IA64 12.0 (ia64 IA64)
      End HostModel
      ```
   3 Save the changes to `lsf.shared`.
   4 Run `lsadmin reconfig` to reconfigure LIM.
   5 Run `badmin reconfig` to reconfigure `mbatchd`.

Registering service ports:
Lava uses dedicated UDP and TCP ports for communication. All hosts in the cluster must use the same port numbers to communicate with each other.
The service port numbers can be any numbers ranging from 1024 to 65535 that are not already used by other services. To make sure that the port numbers you supply are not already used by applications registered in your service database, check `/etc/services` or use the command `ypcat services`.
By default, port numbers for Lava services are defined automatically in the `lsf.conf` file during installation.
If you find any registration conflicts, change your service port numbers as follows:
1 Log on to any host as `root`.
2 Edit `lsf.conf` and add the following lines:
Matching host names and addresses

Lava needs to match host names with the corresponding Internet host addresses. Lava looks up host names and addresses the following ways:

- In the /etc/hosts file
- Sun Network Information Service/Yellow Pages (NIS or YP)
- Internet Domain Name Service (DNS).

  DNS is also known as the Berkeley Internet Name Domain (BIND) or named, which is the name of the BIND daemon.

Each host is configured to use one or more of these mechanisms. Each host has one or more network addresses; usually one for each network to which the host is directly connected. Each host can also have more than one name.

The first name configured for each address is called the official name. Other names for the same host are called aliases.

Lava uses the configured host naming system on each host to look up the official host name for any alias or host address. This means that you can use aliases as input to Lava, but Lava always displays the official name.

Host name services

The following rules apply:

- If your host has an /etc/resolv.conf file, your host is using DNS for name lookups
- If the command ypcat hosts prints out a list of host addresses and names, your system is looking up names in NIS
- Otherwise, host names are looked up in the /etc/hosts file

The man pages for the gethostbyname function, the ypbind and named daemons, the resolver functions, and the hosts, svc.conf, nsswitch.conf, and resolv.conf files explain host name lookups in more detail.

Hosts with multiple addresses

Hosts that have more than one network interface usually have one Internet address for each interface. Such hosts are called multi-homed hosts. Lava identifies hosts by name, so it needs to match each of these addresses with a single host name.
To match each address with a host name, the host name information must be configured so that all of the Internet addresses for a host resolve to the same name.

This can be done in one of the following ways:
- Modify the system hosts file (/etc/hosts) and the changes will affect the whole system
- Create a Lava hosts file (LSF_CONFDIR/hosts) and Lava will be the only application that resolves the addresses to the same host

**Multiple network interfaces**

Some system manufacturers recommend that each network interface, and therefore, each Internet address, be assigned a different host name. Each interface can then be directly accessed by name. This setup is often used to make sure NFS requests go to the nearest network interface on the file server, rather than going through a router to some other interface. This configuration can confuse Lava, because there is no way to determine that the two different names (or addresses) refer to the same host. Lava provides a workaround for this problem.

All host naming systems can be configured so that host address lookups always return the same name, while still allowing access to network interfaces by different names. Each host has an official name and a number of aliases, which are other names for the same host. By configuring all interfaces with the same official name but different aliases, you can refer to each interface by a different alias name while still providing a single official name for the host.

**Configuring the Lava hosts file**

If your Lava clusters include hosts that have more than one interface and are configured with more than one official host name, you must either modify the host name configuration, or create a private hosts file for Lava to use.

The Lava hosts file is stored in LSF_CONFDIR. The format of LSF_CONFDIR/hosts is the same as the format of /etc/hosts.

In the Lava hosts file, duplicate the system hosts database information, except make all entries for the host use the same official name. Configure all the other names for the host as aliases so that people can still refer to the host by any name.

**Example configurations**

If your /etc/hosts file contains:

```
AA.AA.AA.AA host-AA host # first interface
BB.BB.BB.BB host-BB # second interface
```

then the LSF_CONFDIR/hosts file should contain:

```
AA.AA.AA.AA host host-AA # first interface
BB.BB.BB.BB host host-BB # second interface
```

The following example is for a host with two interfaces, where the host does not have a unique official name:
Managing Users, Hosts, and Queues

Inside Platform Lava

# Address Official name Aliases
# Interface on network A
AA.AA.AA.AA host-AA.domain host.domain host-AA host
# Interface on network B
BB.BB.BB.BB host-BB.domain host-BB host

Looking up the address AA.AA.AA.AA finds the official name host-AA.domain. Looking up address BB.BB.BB.BB finds the name host-BB.domain. No information connects the two names, so there is no way for Lava to determine that both names, and both addresses, refer to the same host.

To resolve this case, you must configure these addresses using a unique host name. If you cannot make this change to the system file, you must create a Lava hosts file and configure these addresses using a unique host name in that file.

Here is the same example, with both addresses configured for the same official name:

# Address Official name Aliases
# Interface on network A
AA.AA.AA.AA host.domain host-AA.domain host-AA host
# Interface on network B
BB.BB.BB.BB host.domain host-BB.domain host-BB host

With this configuration, looking up either address returns host.domain as the official name for the host. Lava (and all other applications) can determine that all the addresses and host names refer to the same host. Individual interfaces can still be specified by using the host-AA and host-BB aliases.

Sun’s NIS uses the /etc/hosts file on the NIS master host as input, so the format for NIS entries is the same as for the /etc/hosts file.

Since Lava can resolve this case, you do not need to create a Lava hosts file.

DNS configuration

The configuration format is different for DNS. The same result can be produced by configuring two address (A) records for each Internet address. Following the previous example:

# name class type address
host.domain IN A AA.AA.AA.AA
host.domain IN A BB.BB.BB.BB
host-AA.domain IN A AA.AA.AA.AA
host-BB.domain IN A BB.BB.BB.BB

Looking up the official host name can return either address. Looking up the interface-specific names returns the correct address for each interface.

PTR records in DNS

Address-to-name lookups in DNS are handled using PTR records. The PTR records for both addresses should be configured to return the official name:

# address class type name
AA.AA.AA.AA.in-addr.arpa IN PTR host.domain
BB.BB.BB.BB.in-addr.arpa IN PTR host.domain
If it is not possible to change the system host name database, create the `hosts` file local to the Lava system, and configure entries for the multi-homed hosts only. Host names and addresses not found in the `hosts` file are looked up in the standard name system on your host.

### Controlling queues

Queues are controlled by a Lava Administrator or `root` issuing a command or through configured dispatch and run windows.

#### Adding a queue

1. Log on as the Lava administrator to the front-end host.
2. Edit `lsb.queues` to add the new queue definition.
   
   You can copy another queue definition from this file as a starting point; remember to change the `QUEUE_NAME` of the copied queue.
3. Save the changes to `lsb.queues`.
4. Run `badmin reconfig` to reconfigure `mbatchd`.
   
   Adding a queue does not affect pending or running jobs.

#### Removing a queue

If there are jobs in the queue, move pending and running jobs to another queue, then remove the queue. If you remove a queue that has jobs in it, the jobs are temporarily moved to a queue named `lost_and_found`. Jobs in the `lost_and_found` queue remain pending until the user or the Lava administrator uses the `bswitch` command to switch the jobs into regular queues. Jobs in other queues are not affected.

The following examples use queues named `night` and `idle`.

1. Log on as `root` or the Lava administrator to any host in the cluster.
2. Close the queue to prevent any new jobs from being submitted. For example:
   ```bash
   # badmin qclose night
   Queue <night> is closed
   ```
3. Move all pending and running jobs into another queue. In the following example, the `bswitch -q night` argument chooses jobs from the `night` queue, and the job ID number 0 specifies that all jobs should be switched:
   ```bash
   $ bjobs -u all -q night
   JOBID USER STAT QUEUE FROM_HOST EXEC_HOST JOB_NAME SUBMIT_TIME
   5308 user5 RUN night hostA hostD job5 Nov 21 18:16
   5310 user5 PEND night hostA hostC job10 Nov 21 18:17
   $ bswitch -q night idle 0
   Job <5308> is switched to queue <idle>
   Job <5310> is switched to queue <idle>
   ```
4. Edit `lsb.queues` and remove or comment out the definition for the queue you want to remove.
5. Save the changes to `lsb.queues`.
6. Run `badmin reconfig` to reconfigure `mbatchd`.

#### Closing a queue

Run `badmin qclose`:

```bash
# badmin qclose normal
Queue <normal> is closed
```
When a user tries to submit a job to a closed queue the following message is displayed:

```
$ bsub -q normal ...
normal: Queue has been closed
```

### Opening a queue

Run `badmin qopen`:

```
# badmin qopen normal
Queue <normal> is opened
```

### Inactivating a queue

Run `badmin qinact`:

```
# badmin qinact normal
Queue <normal> is inactivated
```

### Activating a queue

Run `badmin qact`:

```
# badmin qact normal
Queue <normal> is activated
```

### Configuring automatic job requeue

You can configure automatic job requeue to kill and requeue a job while it is running or when it is suspended.

To configure automatic job requeue, set `REQUEUE_EXIT_VALUES` in the queue definition (`lsb.queues`) and specify the exit codes that will cause the job to be requeued.

**Example**

Begin Queue

```
...  
REQUEUE_EXIT_VALUES = 99 100  
...  
End Queue
```

This configuration enables jobs that exit with 99 or 100 to be requeued.

To manually requeue a job, see the instructions in Running Jobs with Platform Lava.

### Configuring exclusive job requeue

Set `REQUEUE_EXIT_VALUES` in the queue definition (`lsb.queues`) and define the exit code using parentheses and the keyword `EXCLUDE`, as shown:

```
EXCLUDE(exit_code...)  
```

When a job exits with any of the specified exit codes, it will be requeued, but it will not be dispatched to the same host again.

**Example**

Begin Queue

```
...  
REQUEUE_EXIT_VALUES=30 EXCLUDE(20)  
HOSTS=hostA hostB hostC  
...  
End Queue
```

A job in this queue can be dispatched to `hostA, hostB, or hostC`. 
If a job running on hostA exits with value 30 and is requeued, it can be dispatched to hostA, hostB, or hostC. However, if a job running on hostA exits with value 20 and is requeued, it can only be dispatched to hostB or hostC. If the job runs on hostB and exits with a value of 20 again, it can only be dispatched on hostC. Finally, if the job runs on hostC and exits with a value of 20, it cannot be dispatched to any of the hosts, so it will pend forever.

Configuring automatic job rerun for a queue

Enable automatic job rerun if you want to requeue and rerun a job when the execution host goes down or when the Lava system fails while the job is running. Rerunnable jobs do not rerun if the job fails.

When a job is rerun or restarted, it is first returned to the queue from which it was dispatched with the same options as the original job. The priority of the job is set sufficiently high to ensure the job gets dispatched before other jobs in the queue. The job uses the same job ID number. It is executed when a suitable host is available, and an email message is sent to the job owner informing the user of the restart.

Automatic job rerun can be enabled at the job level, by the user, or at the queue level, by the Lava administrator. (To submit a rerunnable job, see the instructions in Running Jobs with Platform Lava.) To enable automatic job rerun at the queue level, set RERUNNABLE in lsb.queues to yes.

Example RERUNNABLE = yes

Controlling when jobs run

Dispatch and run windows are time windows that control when Lava jobs start and run.
- Dispatch windows can be defined in lsb.hosts. Dispatch and run windows can be defined in lsb.queues.
- Hosts can only have dispatch windows. Queues can have dispatch windows and run windows.
- Both windows affect job starting; only run windows affect the stopping of jobs.
- Dispatch windows define when hosts and queues are active and inactive. It does not control job submission.
- Run windows define when jobs can and cannot run. While a run window is closed, Lava cannot start any of the jobs placed in the queue, or finish any of the jobs already running.
- When a dispatch window closes, running jobs continue and finish, and no new jobs can be dispatched to the host or from the queue. When a run window closes, Lava suspends running jobs, but new jobs can still be submitted to the queue.
Dispatch windows

A dispatch window specifies one or more time periods during which batch jobs are dispatched to run on hosts. Jobs are not dispatched outside of configured windows. Dispatch windows do not affect job submission and running jobs (they are allowed to run until completion). By default, dispatch windows are not configured, queues are always Active.

To configure dispatch windows:
1. Edit lsb.queues
2. Create a DISPATCH_WINDOW keyword for the queue and specify one or more time windows. For example:

   ```
   Begin Queue
   QUEUE_NAME   = queue1
   PRIORITY     = 45
   DISPATCH_WINDOW = 4:30-12:00
   End Queue
   ```
3. Reconfigure the cluster using:
   a. lsadmin reconfig
   b. badmin reconfig
4. Run bqueues -l to display dispatch windows.

You can also configure dispatch windows for a host, by setting DISPATCH_WINDOW in lsb.hosts and specifying one or more time windows. If no host dispatch window is configured, the window is always open.

Run windows

A run window specifies one or more time periods during which jobs dispatched from a queue are allowed to run. When a run window closes, running jobs are suspended, and pending jobs remain pending. The suspended jobs are resumed when the window opens again. By default, run windows are not configured, queues are always Active and jobs can run until completion.

To configure a run window:
1. Edit lsb.queues.
2. Create a RUN_WINDOW keyword for the queue and specify one or more time windows. For example:

   ```
   Begin Queue
   QUEUE_NAME   = queue1
   PRIORITY     = 45
   RUN_WINDOW   = 4:30-12:00
   End Queue
   ```
3. Reconfigure the cluster:
   a. lsadmin reconfig
   b. badmin reconfig
4. Run bqueues -l to display the run windows.
Error and Event Logging

System directories and log files

Lava uses directories for temporary work files, log files, and transaction files and spooling.

Lava keeps track of all jobs in the system by maintaining a transaction log in the work subtree. The Lava log files are found in the directory /opt/lava/work/lava/logdir

This is not a shared directory. It is not shared with the compute hosts, only the master candidate hosts.

Current job states

Lava uses the lsb.events file to keep track of the state of all jobs. Each job is a transaction from job submission to job completion. Lava keeps track of everything associated with the job in the lsb.events file. By default, mbatchd automatically backs up and rewrites the lsb.events file after every 1000 batch job completions. This value is controlled by the MAX_JOB_NUM parameter in the lsb.params file.

Do not remove or modify the current lsb.events file. Removing or modifying the lsb.events file could cause batch jobs to be lost.

History

The events file is automatically trimmed and old job events are stored in lsb.event.n files. When mbatchd starts, it refers only to the lsb.events file, not the lsb.events.n files. The bhist command refers to the lsb.events.n files.

Job scripts

When a user issues a bsub command from a shell prompt, Lava collects all the commands issued on the bsub line and spools the data to mbatchd, which saves the bsub command script in the info directory for use at dispatch time or if the job is rerun. The info directory is managed by Lava and should not be modified by anyone.

Log directory permissions and ownership

Ensure that the LSF_LOGDIR directory is writable by root. The Lava administrator must own LSF_LOGDIR.
Managing error logs

Error logs maintain important information about Lava operations. When you see any abnormal behavior in Lava, you should first check the appropriate error logs to find out the cause of the problem.

Lava log files grow over time. These files should occasionally be cleared, either by hand or with automatic scripts.

Daemon error log

Lava log files are reopened each time a message is logged, so if you rename or remove a daemon log file, the daemons will automatically create a new log file.

The Lava daemons log messages when they detect problems or unusual situations.

The daemons can be configured to put these messages into files.

The error log file names for the Lava system daemons are:

- lim.log.host_name
- res.log.host_name
- pim.log.host_name
- sbatchd.log.host_name
- mbatchd.log.host_name
- mbschd.log.host_name

Lava daemons log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. Message logging is controlled by the parameter LSF_LOG_MASK in lsf.conf. Possible values for this parameter can be any log priority symbol that is defined in /usr/include/sys/syslog.h.

The default value for LSF_LOG_MASK is LOG_WARNING.

Error logging

If the optional LSF_LOGDIR parameter is defined in lsf.conf, error messages from Lava servers are logged to files in this directory.

If LSF_LOGDIR is defined, but the daemons cannot write to files there, the error log files are created in /tmp.

If LSF_LOGDIR is not defined, errors are logged to the system error logs (syslog). Look for the file /etc/syslog.conf, and read the man pages for syslog(3) and syslogd(1).
Monitoring Your Cluster

Viewing cluster information

Lava provides commands for users to get information about the cluster. Cluster information includes the cluster master host, cluster name, cluster resource definitions, and cluster administrator.

<table>
<thead>
<tr>
<th>To view the</th>
<th>Run...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version of Lava</td>
<td>lsid</td>
</tr>
<tr>
<td>Cluster name</td>
<td>lsid</td>
</tr>
<tr>
<td>Current master host</td>
<td>lsid</td>
</tr>
<tr>
<td>Cluster administrators</td>
<td>lsclusters</td>
</tr>
</tbody>
</table>

Use the `lsid` command to display the version of Lava, the name of your cluster, and the current master host:

```bash
# lsid
Platform Lava 1.0, August 30, 2007
Copyright 1992-2007 Platform Computing Corporation

My cluster name is lava
My master name is frontend-0.public
```

Restarting `sbatchd` on a host does not affect jobs that are running on that host.

If `sbatchd` is shut down, the host is not available to run new jobs. Existing jobs running on that host continue, but the results are not sent to the user until `sbatchd` is restarted.

Configuration errors

You can view configuration errors by using the following commands:

- `lsadmin ckconfig -v`
- `badmin ckconfig -v`

This reports all errors to your terminal.

Viewing host information

Lava uses some or all of the hosts in a cluster as execution hosts. The host list is configured by the Lava administrator. Use the `bhosts` command to view host information. Use the `lsload` command to view host load information.

<table>
<thead>
<tr>
<th>To view...</th>
<th>Run...</th>
</tr>
</thead>
<tbody>
<tr>
<td>All hosts in the cluster and their status</td>
<td><code>bhosts</code></td>
</tr>
<tr>
<td>Detailed server host information</td>
<td><code>bhosts -l</code> and <code>lshosts -l</code></td>
</tr>
<tr>
<td>Host load by host</td>
<td><code>lsload</code></td>
</tr>
<tr>
<td>Host architecture information</td>
<td><code>lshosts</code></td>
</tr>
<tr>
<td>Host history</td>
<td><code>badmin hhist</code></td>
</tr>
<tr>
<td>Host model and type information</td>
<td><code>lsinfo</code></td>
</tr>
<tr>
<td>Viewing job exit rate and load for hosts</td>
<td><code>bhosts -l</code> and <code>bhosts -x</code></td>
</tr>
</tbody>
</table>
Host states describe the ability of a host to accept and run batch jobs in terms of daemon states, load levels, and administrative controls. The `bhosts` and `lsload` commands display host states.

### bhosts
Displays the current status of the host about its ability to run batch jobs:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>Host is available to accept and run new batch jobs.</td>
</tr>
<tr>
<td>avail</td>
<td>Host is down, or LIM and <code>sbatchd</code> are unreachable.</td>
</tr>
<tr>
<td>reach</td>
<td>LIM is running but <code>sbatchd</code> is unreachable.</td>
</tr>
<tr>
<td>closed</td>
<td>Host will not accept new jobs. Use <code>bhosts -l</code> to display the reasons.</td>
</tr>
</tbody>
</table>

### bhosts -l
Displays the closed reasons. A closed host will not accept new batch jobs:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed_Adm</td>
<td>A Lava administrator or root explicitly closed the host using <code>badmin hclose</code>. Running jobs are not affected.</td>
</tr>
<tr>
<td>closed_BUSY</td>
<td>The value of a load index exceeded a threshold (configured in <code>lsb.hosts</code>, displayed by <code>bhosts -l</code>). Running jobs are not affected. Indices that exceed thresholds are identified with an asterisk (*).</td>
</tr>
<tr>
<td>closed_FULL</td>
<td>The configured maximum number of running jobs has been reached. Running jobs will not be affected.</td>
</tr>
<tr>
<td>closed_LIM</td>
<td><code>sbatchd</code> is running but LIM is unavailable.</td>
</tr>
<tr>
<td>closed_LOCK</td>
<td>A Lava administrator or root explicitly locked the host using <code>lsadmin limlock</code>. Running jobs are suspended (SSUSP). Use <code>lsadmin limunlock</code> to unlock LIM on the local host.</td>
</tr>
<tr>
<td>closed_WIND</td>
<td>Host is closed by a dispatch window defined in <code>lsb.hosts</code>. Running jobs are not affected.</td>
</tr>
</tbody>
</table>

### lsload
Displays the current state of the host about its ability to run batch jobs and remote tasks:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>Host is available to accept and run batch jobs and remote tasks.</td>
</tr>
<tr>
<td>-ok</td>
<td>LIM is running but RES is unreachable.</td>
</tr>
<tr>
<td>busy</td>
<td>Does not affect batch jobs, only used for remote task placement (i.e., <code>lsrun</code>). The value of a load index exceeded a threshold (configured in <code>lsf.cluster.lava</code>, displayed by <code>lshosts -l</code>). Indices that exceed thresholds are identified with an asterisk (*).</td>
</tr>
<tr>
<td>lockW</td>
<td>Does not affect batch jobs, only used for remote task placement (i.e., <code>lsrun</code>). Host is locked by a run window (configured in <code>lsf.cluster.lava</code>, displayed by <code>lshosts -l</code>).</td>
</tr>
<tr>
<td>lockU</td>
<td>Will not accept new batch jobs or remote tasks. A Lava administrator or root explicitly locked the host (i.e., <code>lsadmin limlock</code>). Running jobs are not affected.</td>
</tr>
<tr>
<td>avail</td>
<td>Host is down, or LIM is unavailable.</td>
</tr>
</tbody>
</table>

To view all hosts in the cluster

Run `bhosts` to display information about all hosts and their status.
To view detailed host information

Run `bhosts -l host_name` and `lshosts -l host_name` to display all information about each server host such as the CPU factor and the load thresholds to start, suspend, and resume jobs.

To view host load by host

The `lsload` command reports the current status and load levels of hosts in a cluster. The `lshosts -l` command shows the load thresholds.

The `lsmon` command provides a dynamic display of the load information. The Lava administrator can find unavailable or overloaded hosts with these tools.

Run `lsload` to see load levels for each host.

To view host architecture

A Lava cluster may consist of hosts of differing architectures and speeds. The `lshosts` command displays configuration information about hosts. All these parameters are defined by the Lava administrator in the Lava configuration files, or determined by the LIM directly from the system.

Host types represent binary compatible hosts; all hosts of the same type can run the same executable. Host models give the relative CPU performance of different processors.

To view host history

Run `badmin hhist` to view the history of a host such as when it is opened or closed.

To view host model and type

Run `lsinfo -m` to display information about host models that exist in the cluster.

Run `lim -t` to display the model of the current host. You must be the Lava administrator to use this command.

To view host dispatch windows

Use `bhosts -l` to display host dispatch windows.

Viewing queue information

The `bqueues` command displays information about queues. The `bqueues -l` option also gives current statistics about the jobs in a particular queue such as the total number of jobs in the queue, the number of running jobs, and the number of suspended jobs.

Queue states, displayed by `bqueues`, describe the ability of a queue to accept and start batch jobs using a combination of the following states:
- Open queues accept new jobs
- Closed queues do not accept new jobs
- Active queues start jobs on available hosts
- Inactive queues hold all jobs
Queue states can be changed by a Lava administrator or root.

In addition to the procedures listed here, see the `bqueues(1)` man page for more details.

**To view available queues**

Run `bqueues`. You can view the current status of a particular queue or all queues. The `bqueues` command also displays available queues in the cluster.

Use `bqueues -u user_name` to specify a user so that `bqueues` displays only the queues that accept jobs from these users.

The `bqueues -m host_name` option allows users to specify a host name so that `bqueues` displays only the queues that use these hosts to run jobs.

**To view detailed queue information**

To see the complete status and configuration for each queue, run `bqueues -l`. You can specify queue names on the command-line to select specific queues.

**To view the history of state changes in a queue**

Run `badmin qhist` to display the times when queues are opened, closed, activated, and inactivated.

**To view queue administrators**

Use `bqueues -l` for the queue.

**To view information about run windows**

Use `bqueues -r` to display information about queue run windows.

**To view queue dispatch windows**

Use `bqueues -d` to display queue dispatch windows.
CHAPTER 2

Working with Resources and Resource Requirements

Contents

- “Resource Classifications” on page 30
- “Configuring Your Own Resources” on page 36
- “External Load Indices and ELIM” on page 39
- “Configuring Resource Requirements” on page 42
- “Resource Requirement Strings” on page 43
- “Monitoring Resources” on page 46
Resource Classifications

The Lava system uses built-in and configured resources to track job resource requirements and schedule jobs according to the resources available on individual hosts.

**How resources are classified**

<table>
<thead>
<tr>
<th>By values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean resources</td>
<td>Resources that denote the availability of specific features.</td>
</tr>
<tr>
<td>Numerical resources</td>
<td>Resources that take numerical values such as all the load indices, number of processors on a host, or host CPU factor.</td>
</tr>
<tr>
<td>String resources</td>
<td>Resources that take string values such as host type, host model, host status.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>By the way values change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Resources</td>
</tr>
<tr>
<td>Static Resources</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>By definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site-Defined Resources</td>
</tr>
<tr>
<td>Built-In Resources</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>By scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host-Based Resources</td>
</tr>
<tr>
<td>Shared Resources</td>
</tr>
</tbody>
</table>

**Note:** Before you can specify resources or add your own configured resources, you must define your hosts in the Host section of lsf.cluster.lava. By default, the compute hosts are added to the cluster dynamically and are not defined in lsf.cluster.lava. Follow the example definitions in lsf.cluster.lava.
### Boolean resources

Boolean resources have a value of one (1) if they are defined for a host, and zero (0) if they are not defined for the host. Use Boolean resources to configure host attributes to be used in selecting hosts to run jobs. For example:

- Machines may have different types and versions of operating systems.
- Machines may play different roles in the system, such as file server or compute server.
- Some machines may have special-purpose devices needed by some applications.
- Certain software packages or licenses may be available only on some of the machines.

Specify a Boolean resource in a resource requirement selection string of a job, to select only hosts that can run the job.

Some examples of Boolean resources:

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>Describes</th>
<th>Meaning of Example Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>Role in cluster</td>
<td>Compute server</td>
</tr>
<tr>
<td>fs</td>
<td>Role in cluster</td>
<td>File server</td>
</tr>
<tr>
<td>linux64</td>
<td>Operating system</td>
<td>Linux operating system</td>
</tr>
<tr>
<td>frame</td>
<td>Available software</td>
<td>FrameMaker license</td>
</tr>
</tbody>
</table>

#### Use a boolean resource to specify a host

Usually, to indicate that a job must run on one of a number of specified hosts, you use the `bsub -m "hostA hostB ..."` option. By specifying a single host, you can force your job to wait until that host is available and then run on that host.

If you have applications that need specific resources, it is more flexible to create a new Boolean resource and configure that resource for the appropriate hosts in the cluster. This must be done by the Lava administrator. If you specify a host list using the `-m` option of `bsub`, you must change the host list every time you add a new host that supports the desired resources. By using a Boolean resource, the Lava administrator can add, move, or remove resources without forcing users to learn about changes to resource configuration.

### Load indices

Load indices are built-in resources that measure the availability of dynamic, non-shared resources on hosts in the Lava cluster.

Load indices built into the LIM are updated at fixed time intervals.

External load indices are defined and configured by the Lava administrator. An External Load Information Manager (ELIM) program collects the values of site-defined external load indices and updates LIM when new values are received.
Load indices collected by LIM

<table>
<thead>
<tr>
<th>Index</th>
<th>Measures</th>
<th>Units</th>
<th>Direction</th>
<th>Averaged over</th>
<th>Update Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>host status</td>
<td>string</td>
<td></td>
<td></td>
<td>15 seconds</td>
</tr>
<tr>
<td>r15s</td>
<td>run queue length</td>
<td>processes</td>
<td>increasing</td>
<td>15 seconds</td>
<td>15 seconds</td>
</tr>
<tr>
<td>r1m</td>
<td>run queue length</td>
<td>processes</td>
<td>increasing</td>
<td>1 minute</td>
<td>15 seconds</td>
</tr>
<tr>
<td>r15m</td>
<td>run queue length</td>
<td>processes</td>
<td>increasing</td>
<td>15 minutes</td>
<td>15 seconds</td>
</tr>
<tr>
<td>ut</td>
<td>CPU utilization</td>
<td>percent</td>
<td>increasing</td>
<td>1 minute</td>
<td>15 seconds</td>
</tr>
<tr>
<td>pg</td>
<td>paging activity</td>
<td>pages in + pages out per second</td>
<td>increasing</td>
<td>1 minute</td>
<td>15 seconds</td>
</tr>
<tr>
<td>ls</td>
<td>logins</td>
<td>users</td>
<td>increasing</td>
<td>N/A</td>
<td>30 seconds</td>
</tr>
<tr>
<td>it</td>
<td>idle time</td>
<td>minutes</td>
<td>decreasing</td>
<td>N/A</td>
<td>30 seconds</td>
</tr>
<tr>
<td>swp</td>
<td>available swap space</td>
<td>MB</td>
<td>decreasing</td>
<td>N/A</td>
<td>15 seconds</td>
</tr>
<tr>
<td>mem</td>
<td>available memory</td>
<td>MB</td>
<td>decreasing</td>
<td>N/A</td>
<td>15 seconds</td>
</tr>
<tr>
<td>tmp</td>
<td>available space in temporary file system</td>
<td>MB</td>
<td>decreasing</td>
<td>N/A</td>
<td>120 seconds</td>
</tr>
<tr>
<td>io</td>
<td>disk I/O (shown by lsl -l)</td>
<td>KB per second</td>
<td>increasing</td>
<td>1 minute</td>
<td>15 seconds</td>
</tr>
<tr>
<td>name</td>
<td>external load index configured by Lava administrator</td>
<td>site-defined</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Status**

The **status** index is a string indicating the current status of the host. This status applies to the LIM and RES.

The possible values for **status** are:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>The host is available to accept remote jobs. The LIM can select the host for remote execution.</td>
</tr>
<tr>
<td>-ok</td>
<td>When the status of a host is preceded by a dash (-), it means LIM is available but RES is not running on that host or is not responding.</td>
</tr>
<tr>
<td>busy</td>
<td>The host is overloaded (busy) because a load index exceeded a configured threshold. An asterisk (*) marks the offending index. LIM will not select the host for interactive jobs.</td>
</tr>
<tr>
<td>lockW</td>
<td>The host is locked by its run window. Use lshosts to display run windows.</td>
</tr>
<tr>
<td>lockU</td>
<td>The host is locked by a Lava administrator or root.</td>
</tr>
<tr>
<td>unavail</td>
<td>The host is down or the LIM on the host is not running or is not responding.</td>
</tr>
</tbody>
</table>

CPU run queue lengths

The **r15s**, **r1m**, and **r15m** load indices are the 15-second, 1-minute, and 15-minute average CPU run queue lengths. This is the average number of processes ready to use the CPU during the given interval.

Run queue length indices are not necessarily the same as the load averages printed by the `uptime(1)` command; `uptime` load averages on some platforms also include processes that are in short-term wait states (such as paging or disk I/O).
**Effective run queue length**

On multiprocessor systems, more than one process can execute at a time. Lava scales the run queue value on multiprocessor systems to make the CPU load of uniprocessors and multiprocessors comparable. The scaled value is called the effective run queue length. Use `lsload -E` to view the effective run queue length.

**Normalized run queue length**

Lava also adjusts the CPU run queue based on the relative speeds of the processors (the CPU factor). The normalized run queue length is adjusted for both number of processors and CPU speed. The host with the lowest normalized run queue length will run a CPU-intensive job the fastest. Use `lsload -N` to view the normalized CPU run queue lengths.

**CPU utilization**

The `ut` index measures CPU utilization, which is the percentage of time spent running system and user code. A host with no process running has a `ut` value of 0 percent; a host on which the CPU is completely loaded has a `ut` of 100 percent.

**Paging rate**

The `pg` index gives the virtual memory paging rate in pages per second. This index is closely tied to the amount of available RAM memory and the total size of the processes running on a host; if there is not enough RAM to satisfy all processes, the paging rate will be high. Paging rate is a good measure of how a machine will respond to interactive use; a machine that is paging heavily feels very slow.

**Interactive idle time**

The `it` index is the interactive idle time of the host, in minutes. Idle time is measured from the last input or output on a directly attached terminal or a network pseudo-terminal supporting a login session. This does not include activity directly through the X server such as CAD applications or `emacs` windows.

**Temporary directories**

The `tmp` index is the space available in MB on the file system that contains the temporary directory (`/tmp`).

**Swap space**

The `swp` index gives the currently available virtual memory (swap space) in MB. This represents the largest process that can be started on the host.

**Memory**

The `mem` index is an estimate of the real memory currently available to user processes. This represents the approximate size of the largest process that could be started on a host without causing the host to start paging.

LIM reports the amount of free memory available. Lava calculates free memory as a sum of physical free memory, cached memory, buffered memory, and an adjustment value. The command `vmstat` also reports free memory but displays these values separately. There may be a difference between the free memory reported by LIM and the free memory reported by `vmstat` because of virtual memory behavior variations among operating systems. You can write an ELIM that overrides the free memory values returned by LIM. (For information on ELIMs, see “External Load Indices and ELIM” on page 39.)

**I/O rate**

The `io` index measures I/O throughput to disks attached directly to this host, in KB per second. It does not include I/O to disks that are mounted from other hosts.
Static resources

Static resources are built-in resources that represent host information that does not change over time, such as the maximum RAM available to user processes or the number of processors in a machine. Most static resources are determined by the LIM at start-up time, or when Lava detects hardware configuration changes.

Static resources can be used to select appropriate hosts for particular jobs based on binary architecture, relative CPU speed, and system configuration.

The resources `ncpus`, `maxmem`, `maxswp`, and `maxtmp` are not static on hosts that support dynamic hardware reconfiguration.

<table>
<thead>
<tr>
<th>Index</th>
<th>Measures</th>
<th>Units</th>
<th>Determined by</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>host type</td>
<td>string</td>
<td>configuration</td>
</tr>
<tr>
<td>model</td>
<td>host model</td>
<td>string</td>
<td>configuration</td>
</tr>
<tr>
<td>hname</td>
<td>host name</td>
<td>string</td>
<td>configuration</td>
</tr>
<tr>
<td>cpuf</td>
<td>CPU factor</td>
<td>relative</td>
<td>configuration</td>
</tr>
<tr>
<td>server</td>
<td>host can run remote jobs</td>
<td>Boolean</td>
<td>configuration</td>
</tr>
<tr>
<td>rexpri</td>
<td>execution priority</td>
<td>nice(2) argument</td>
<td>configuration</td>
</tr>
<tr>
<td>ncpus</td>
<td>number of processors</td>
<td>processors</td>
<td>LIM</td>
</tr>
<tr>
<td>ndisks</td>
<td>number of local disks</td>
<td>disks</td>
<td>LIM</td>
</tr>
<tr>
<td>maxmem</td>
<td>maximum RAM</td>
<td>MB</td>
<td>LIM</td>
</tr>
<tr>
<td>maxswp</td>
<td>maximum swap space</td>
<td>MB</td>
<td>LIM</td>
</tr>
<tr>
<td>maxtmp</td>
<td>maximum space in /tmp</td>
<td>MB</td>
<td>LIM</td>
</tr>
</tbody>
</table>

CPU factor

The CPU factor (`cpuf`) is the speed of the host’s CPU relative to other hosts in the cluster. If one processor is twice the speed of another, its CPU factor should be twice as large. The CPU factors are defined by the Lava administrator. For multiprocessor hosts, the CPU factor is the speed of a single processor; Lava automatically scales the host CPU load to account for additional processors.

Shared resources

Shared resources are configured resources that are not tied to a specific host, but are associated with the entire cluster or a specific subset of hosts within the cluster. For example:
- Floating licenses for software packages
- Disk space on a file server that is mounted by several machines
- The physical network connecting the hosts

An application may use a shared resource by running on any host from which that resource is accessible. For example, in a cluster in which each host has a local disk but can also access a disk on a file server, the disk on the file server is a shared resource, and the local disk is a host-based resource. In contrast to host-based resources such as memory or swap space, a shared resource from one machine affects the availability of that resource as seen by other machines. One value for the entire cluster measures the utilization of the shared resource, but each host-based resource is measured separately.
Lava does not contain any built-in shared resources. All shared resources must be configured by the Lava administrator. A shared resource may be configured to be dynamic or static. In the above example, the total space on the shared disk may be static while the amount of space currently free is dynamic. A site may also configure the shared resource to report numeric, string, or Boolean values.

The following restrictions apply to the use of shared resources in Lava.

- A shared resource cannot be used as a load threshold in the Hosts section of the lsf.cluster.lava file.
- A shared resource cannot be used in the loadSched/ loadStop thresholds, or in the STOP_COND parameter in the queue definition in the lsb.queues file.

For information on loadSched, loadStop, and STOP_COND, see “Configuring Load Thresholds” on page 53.
Configuring Your Own Resources

Lava schedules jobs based on available resources. There are many resources built into Lava, but you can also add your own resources, and then use them the same way as built-in resources.

For maximum flexibility, you should characterize your resources clearly enough so that users have satisfactory choices. For example, if some of your machines are connected to both Ethernet and InfiniBand, while others are only connected to Ethernet, then you probably want to define a resource called `infini` and associate the `infini` resource with machines connected to InfiniBand. This way, users can specify resource `infini` if they want their jobs to run on machines connected to InfiniBand.

Adding new resources to your cluster

To add host resources to your cluster, use the following steps:

1. Log on to any host in the cluster as the Lava administrator.
2. Define new resources in the `Resource` section of `lsf.shared`. Specify at least a name and a brief description, which will be displayed to a user by `lsinfo`. See “Configuring the lsf.shared resource section” on page 36.
3. For static Boolean resources for all hosts that have the new resources, add the resource name to the `RESOURCES` column in the `Host` section of `lsf.cluster.lava`.
4. For shared resources for all hosts that have the new resources, associate the resources with the hosts (you might also have a reason to configure non-shared resources in this section).
   See “Configuring the lsf.cluster.lava resourcemap section” on page 37.
5. Reconfigure your cluster.

Configuring the lsf.shared resource section

Configured resources are defined in the `Resource` section of `lsf.shared`. There is no distinction between shared and non-shared resources.

You must specify at least a name and description for the resource, using the keywords `RESOURCENAME` and `DESCRIPTION`.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following characters
  : . ( ) [ ] + - * / ! & | < > @ =
- A resource name cannot be any of the following reserved names:
  `cpu cpuf io login ls idle maxmem maxswp maxtmp type model status it mem ncpus ndisks pg r15m r15s r1m swap swp tmp ut`  
- Resource names are case sensitive
- Resource names can be up to 29 characters in length
- You can also specify:
  - The resource type (TYPE = Boolean | String | Numeric)
    The default is Boolean.
  - For dynamic resources, the update interval: `INTERVAL`, in seconds
For numeric resources, where a higher value indicates greater load:
\[ \text{INCREASING} = Y \]

For numeric shared resources, where Lava releases the resource when a job using the resource is suspended:
\[ \text{RELEASE} = Y \]

When the optional attributes are not specified, the resource is treated as static and Boolean.

**Example**

```
Begin Resource
RESOURCENAME TYPE INTERVAL INCREASING DESCRIPTION
mips Boolean () () (MIPS architecture)
dec Boolean () () (DECStation system)
scratch Numeric 30 N (Shared scratch space on server)
synopsys Numeric 30 N (Floating licenses for Synopsys)
verilog Numeric 30 N (Floating licenses for Verilog)
console String 30 N (User Logged in on console)
End Resource
```

**Configuring the lsf.cluster.lava resourcemap section**

Resources are associated with the hosts for which they are defined in the `ResourceMap` section of `lsf.cluster.lava`.

For each resource, you must specify the name and the hosts that have it.

Make sure that the hosts that have the resources you want to configure are defined in the `Host` section of `lsf.cluster.lava`. By default, the compute hosts are added to the cluster dynamically and are not defined in `lsf.cluster.lava`.

If the `ResourceMap` section is not defined, then any dynamic resources specified in `lsf.shared` are not tied to specific hosts, but are shared across all hosts in the cluster.

**Example**

A cluster consists of hosts `host1`, `host2`, and `host3`.

```
Begin ResourceMap
RESOURCENAME LOCATION
verilog (5@[all ~host1 ~host2])
synopsys (2@[host1 host2] 2@[others])
console (1@[host1] 1@[host2]1@[host3])
xyz (1@[default])
End ResourceMap
```

In this example:
- Five units of the `verilog` resource are defined on `host3` only (all hosts except `host1` and `host2`).
- Two units of the `synopsys` resource are shared between `host1` and `host2`. Two more units of the `synopsys` resource are defined on `host3` (shared among all the remaining hosts in the cluster).
- One unit of the `console` resource is defined on each host in the cluster (assigned explicitly). One unit of the `xyz` resource is defined on each host in the cluster (assigned with the keyword default).

**RESOURCENAME** The name of the resource, as defined in `lsf.shared`.
LOCATION  Defines the hosts that share the resource. For a static resource, you must define an initial value here as well. Do not define a value for a dynamic resource.

Possible states of a resource:
- Each host in the cluster has the resource
- The resource is shared by all hosts in the cluster
- There are multiple instances of a resource within the cluster, and each instance is shared by a unique subset of hosts.

Syntax

```
([resource_value]?[host_name... | all [-host_name]... | others | default]
...
```

- For static resources, you must include the resource value, which indicates the quantity of the resource. Do not specify the resource value for dynamic resources because information about dynamic resources is updated by ELIM.
- Type square brackets around the list of hosts, as shown. You can omit the parenthesis if you only specify one set of hosts.
- Each set of hosts within square brackets specifies an instance of the resource. The same host cannot be in more than one instance of a resource. All hosts within the instance share the quantity of the resource indicated by its value.
- The keyword all refers to all the server hosts in the cluster, collectively. Use the not operator (~) to exclude hosts.
- The keyword others refers to all hosts not otherwise listed in the instance.
- The keyword default refers to each host in the cluster, individually.
External Load Indices and ELIM

The Lava Load Information Manager (LIM) collects built-in load indices that reflect the load situations of CPU, memory, disk space, I/O, and interactive activities on individual hosts.

While built-in load indices might be sufficient for most jobs, you might have special workload or resource dependencies that require custom external load indices defined and configured by the Lava administrator. Load and shared resource information from external load indices are used the same as built in load indices for job scheduling and host selection.

You can write an External Load Information Manager (ELIM) program that collects the values of configured external load indices and updates LIM when new values are received.

An ELIM can be as simple as a small script or as complicated as a sophisticated C program. A well-defined protocol allows the ELIM to talk to LIM.

The ELIM executable must be located in LSF_SERVERDIR.

How Lava uses ELIM for external resource collection

The values of static external resources are specified through the lsf.cluster.lava configuration file. The values of all dynamic resources, regardless of whether they are shared or host-based, are collected through an ELIM.

When an ELIM is started

An ELIM is started in the following situations:

- On every host, if any dynamic resource is configured as host-based. For example, if the LOCATION field in the ResourceMap section of lsf.cluster.lava is ([default]), then every host will start an ELIM.
- On the master host, for any cluster-wide resources. For example, if the LOCATION field in the ResourceMap section of lsf.cluster.lava is ([all]), then an ELIM is started on the master host.
- On the first host specified for each instance, if multiple instances of the resource exist within the cluster. For example, if the LOCATION field in the ResourceMap section of lsf.cluster.lava is ([hostA hostB hostC] [hostD hostE hostF]), then an ELIM will be started on hostA and hostD to report the value of that resource for that set of hosts.

If the host reporting the value for an instance goes down, then an ELIM is started on the next available host in the instance. In above example, if hostA became unavailable, an ELIM is started on hostB. If the hostA becomes available again then the ELIM on hostB is shut down and the one on hostA is started.

There is only one ELIM on each host, regardless of the number of resources on which it reports. If only cluster-wide resources are to be collected, then an ELIM will only be started on the master host.

Environment variables

When LIM starts, the following environment variables are set for ELIM:

- LSF_MASTER: This variable is defined if the ELIM is being invoked on the master host; otherwise, it is undefined. This can be used to test whether the ELIM should report on cluster-wide resources that only need to be collected on the master host.
External Load Indices and ELIM

- LSF RESOURCES: This variable contains a list of resource names (separated by spaces) on which the ELIM is expected to report. A resource name is only put in the list if the host on which the ELIM is running shares an instance of that resource.

Writing an ELIM

The ELIM must be an executable program, either an interpreted script or compiled code.

ELIM output

The ELIM communicates with the LIM by periodically writing a load update string to its standard output. The load update string contains the number of indices followed by a list of name-value pairs in the following format:

number_indices [index_name index_value]...

For example,

3 tmp2 47.5 nio 344.0 licenses 5

This string reports three indices: tmp2, nio, and licenses, with values 47.5, 344.0, and 5 respectively. Index values must be numbers between -INFINIT_LOAD and INFINIT_LOAD as defined in the lsf.h header file.

If the ELIM is implemented as a C program, as part of initialization it should use setbuf(3) to establish unbuffered output to stdout.

The ELIM should ensure that the entire load update string is written successfully to stdout. This can be done by checking the return value of printf(3) if the ELIM is implemented as a C program or as the return code of /bin/echo(1) from a shell script. The ELIM should exit if it fails to write the load information.

Each LIM sends updated load information to the master every 15 seconds. Depending on how quickly your external load indices change, the ELIM should write the load update string at most once every 15 seconds. If the external load indices rarely change, the ELIM can write the new values only when a change is detected. The LIM continues to use the old values until new values are received.

ELIM location

The executable for the ELIM must be in LSF_SERVERDIR.

Use the following naming convention:

LSF_SERVERDIR/elim.application

For example, elim.license

If LIM expects some resources to be collected by an ELIM according to configuration, it invokes the ELIM automatically on startup. The ELIM runs with the same user ID and file access permission as the LIM.

Note that LSF_SERVERDIR is not a shared directory.

ELIM restart

The LIM restarts the ELIM if it exits. To prevent problems in case of a fatal error in the ELIM, it is restarted at most once every 90 seconds. When the LIM terminates, it sends a SIGTERM signal to the ELIM. The ELIM must exit upon receiving this signal.
## Debugging an ELIM

Set the parameter `LSF_ELIM_DEBUG=y` in the Parameters section of `lsf.cluster.lava` to log all load information received by LIM from the ELIM in the LIM log file.

Set the parameter `LSF_ELIM_BLOCKTIME=seconds` in the Parameters section of `lsf.cluster.lava` to configure how long LIM waits before restarting the ELIM.

Use the parameter `LSF_ELIM_RESTARTS=integer` in the Parameters section of `lsf.cluster.lava` to limit the number of times an ELIM can be restarted.

See the Platform Lava man pages for more details on these parameters.
Configuring Resource Requirements

Resource requirements define which hosts a job can run on. Each job has its resource requirements. Hosts that match the resource requirements are the candidate hosts. When Lava schedules a job, it uses the load index values of all the candidate hosts. The load values for each host are compared to the scheduling conditions. Jobs are only dispatched to a host if all load values are within the scheduling thresholds.

Default configuration

If a job has no resource requirements, Lava places it on a host of the same type as the submission host (type==local). However, if a job has string or Boolean resource requirements specified and the host type has not been specified, Lava places the job on any host (type==any) that satisfies the resource requirements.

When to configure resource requirements

To override the Lava defaults, specify resource requirements explicitly. Resource requirements can be set for queues, for individual applications, or for individual jobs. A resource requirement is an expression that contains resource names and operators.

Defining resource requirements for a queue

Each queue can define resource requirements that will be applied to all the jobs in the queue.

When resource requirements are specified for a queue, and no job-level resource requirement is specified, the queue-level resource requirements become the default resource requirements for the job.

Syntax

The condition for dispatching a job to a host can be specified through the queue-level RES_REQ parameter in the queue definition in lsb.queues.

Example

RES_REQ=select[\((\text{hname==hostA} \&\& \text{mem} > 50) | (\text{hname==hostB} \&\& \text{mem} > 100)\)\]

Using the hname resource in the resource requirement string allows you to set up different conditions for different hosts in the same queue.

To specify resource requirements for a specific job, see Running Jobs with Platform Lava.
Resource Requirement Strings

Most Lava commands accept a `-R res_req` argument to specify resource requirements.

A resource requirement string describes the resources a job needs. Lava uses resource requirements to select hosts for remote execution and job execution.

How queue and job resource requirements are resolved

If job-level resource requirements are specified together with queue-level resource requirements:

- In a `select` string, a host must satisfy both queue-level and job-level requirements for the job to be dispatched.
- An `order` section defined at the queue level is ignored if different `order` requirements are specified at the job level. The default `order` string is `r15s:pg`.

Resource requirement string sections

- A selection section (`select`). The selection section specifies the criteria for selecting hosts from the system.
- An ordering section (`order`). The ordering section indicates how the hosts that meet the selection criteria should be sorted.

Syntax

```
select [selection_string] order [order_string]
```

The square brackets must be typed as shown.

The section names are `select` and `order`.

If no section name is given, the entire string is treated as a selection string. The `select` keyword may be omitted if the selection string is the first string in the resource requirement.

Each section has a different syntax.

Selection string

The selection string specifies the characteristics a host must have to match the resource requirement. It is a logical expression built from a set of resource names. The selection string is evaluated for each host; if the result is non-zero, then that host is selected.

Syntax

The selection string can combine resource names with logical and arithmetic operators. Non-zero arithmetic values are treated as logical TRUE, and 0 as logical FALSE. Boolean resources have a value of 1 if they are defined for a host, and 0 if they are not defined for the host.

The resource names `swap`, `idle`, `login`, and `cpu` are accepted as aliases for `swp`, `it`, `ls`, and `r1m` respectively.

For `ut`, specify the percentage CPU utilization as an integer between 0-100.

For the string resources `type` and `model`, the special value `any` selects any value and `local` selects the same value as that of the local host. For example, `type==local` selects hosts of the same type as the host submitting the job. If a job can run on any type of host, include `type==any` in the resource requirements.
If no type is specified, the default for bsub is type=local unless a string or Boolean resource is specified, in which case it is type=any.

**Operators**

These operators can be used in selection strings. The operators are listed in order of decreasing precedence.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>Negative of a</td>
</tr>
<tr>
<td>!a</td>
<td>Logical not: 1 if a==0, 0 otherwise</td>
</tr>
<tr>
<td>a*b</td>
<td>Multiply a and b</td>
</tr>
<tr>
<td>a/b</td>
<td>Divide a by b</td>
</tr>
<tr>
<td>a + b</td>
<td>Add a and b</td>
</tr>
<tr>
<td>a - b</td>
<td>Subtract b from a</td>
</tr>
<tr>
<td>a &gt; b</td>
<td>1 if a is greater than b, 0 otherwise</td>
</tr>
<tr>
<td>a &lt; b</td>
<td>1 if a is less than b, 0 otherwise</td>
</tr>
<tr>
<td>a &gt;= b</td>
<td>1 if a is greater than or equal to b, 0 otherwise</td>
</tr>
<tr>
<td>a &lt;= b</td>
<td>1 if a is less than or equal to b, 0 otherwise</td>
</tr>
<tr>
<td>a == b</td>
<td>1 if a is equal to b, 0 otherwise</td>
</tr>
<tr>
<td>a != b</td>
<td>1 if a is not equal to b, 0 otherwise</td>
</tr>
<tr>
<td>a &amp;&amp; b</td>
<td>Logical AND: 1 if both a and b are non-zero, 0 otherwise</td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```bash
select[(((2*r1s + 3*r1m + r15m) / 6 < 1.0) && !fs && (cpuf > 4.0))]
```

**Specifying shared resources with the keyword “defined”**

A shared resource may be used in the resource requirement string of any Lava command. For example, when submitting a Lava job that requires a certain amount of shared scratch space, you might submit the job as follows:

```bash
$ bsub -R "avail_scratch > 200 && swap > 50" myjob
```

The above assumes that all hosts in the cluster have access to the shared scratch space. The job will only be scheduled if the value of the "avail_scratch" resource is more than 200 MB and will go to a host with at least 50 MB of available swap space.

It is possible for a system to be configured so that only some hosts within the Lava cluster have access to the scratch space. To exclude hosts that cannot access a shared resource, the defined(resource_name) function must be specified in the resource requirement string.

For example:

```bash
$ bsub -R "defined(avail_scratch) && avail_scratch > 100 && swap > 100" myjob
```

would exclude any hosts that cannot access the scratch resource. The Lava administrator configures which hosts do and do not have access to a particular shared resource.
**Order string**

The order string allows the selected hosts to be sorted according to the values of resources. The values of \( r_{15s}, r_{1m}, \) and \( r_{15m} \) used for sorting are the normalized load indices returned by `lsload -N`.

The order string is used for host sorting and selection. The ordering begins with the rightmost index in the order string and proceeds from right to left. The hosts are sorted into order based on each load index, and if more hosts are available than were requested, the LIM drops the least desirable hosts according to that index. The remaining hosts are then sorted by the next index.

After the hosts are sorted by the leftmost index in the order string, the final phase of sorting orders the hosts according to their status, with hosts that are currently not available for load sharing (not in the `ok` state) listed at the end.

Because the hosts are sorted again for each load index, only the host status and the leftmost index in the order string actually affect the order in which hosts are listed. The other indices are only used to drop undesirable hosts from the list.

When sorting is done on each index, the direction in which the hosts are sorted (increasing vs. decreasing values) is determined by the default order returned by `lsinfo` for that index. This direction is chosen such that after sorting, by default, the hosts are ordered from best to worst on that index.

**Syntax**

\[ [-\]resource\_name [:\[-\]resource\_name]... \]

You can specify any built-in or external load index.

When an index name is preceded by a minus sign `-'`, the sorting order is reversed so that hosts are ordered from worst to best on that index.

**Default**

The default sorting order is \( r_{15s}:pg \) (except for `lslogin(1):ls:r_{1m}`).

**Example**

\( swp:r_{1m}:tmp:r_{15s} \)
Monitoring Resources

```
lsinfo
Use lsinfo to list the resources available in your cluster. The lsinfo command lists all the resource names and their descriptions:

```
$ lsinfo

```
RESOURCE_NAME   TYPE   ORDER   DESCRIPTION
r15s             Numeric Inc 15-second CPU run queue length
r1m              Numeric Inc 1-minute CPU run queue length (alias: cpu)
r15m             Numeric Inc 15-minute CPU run queue length
ut               Numeric Inc 1-minute CPU utilization (0.0 to 1.0)
pg               Numeric Inc Paging rate (pages/second)
io               Numeric Inc Disk IO rate (Kbytes/second)
ls                Numeric Inc Number of login sessions (alias: login)
it               Numeric Dec Idle time (minutes) (alias: idle)
tmp              Numeric Dec Disk space in /tmp (Mbytes)
swp              Numeric Dec Available swap space (Mbytes) (alias: swap)
mem              Numeric Dec Available memory (Mbytes)
ncpus             Numeric Dec Number of CPUs
ndisks            Numeric Dec Number of local disks
maxmem            Numeric Dec Maximum memory (Mbytes)
maxswp            Numeric Dec Maximum swap space (Mbytes)
maxtmp            Numeric Dec Maximum /tmp space (Mbytes)
cpuf              Numeric Dec CPU factor
server            Boolean N/A Lava server host
cserver           Boolean N/A Compute server
cserver           Boolean N/A File server
type              String N/A Host type
model             String N/A Host model
status             String N/A Host status
hname             String N/A Host name

```

```
TYPE_NAME
HPPA
SGI6
ALPHA
SUNSOL
RS6K
NTX86

```

```
MODEL_NAME CPU_FACTOR
DEC3000  10.00
R10K     14.00
PENT200  6.00
IBM350   7.00
SunSparc 6.00
HP735    9.00
HP715    5.00

```

```
lshosts
Use lshosts to get a list of the resources defined on a specific host:

```
$ lshosts hostA

```
HOST_NAME   type   model   cpuf   ncpus  maxmem  maxswp  server RESOURCES
hostA       SOL732  Ultra2  20.2   2     256M   679M    Yes ()
```
### Viewing host load by resource

Use `lshosts -s` to view host load by shared resource:

```bash
$ lshosts -s

<table>
<thead>
<tr>
<th>RESOURCE</th>
<th>VALUE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tot_lic</td>
<td>5</td>
<td>host1, host2</td>
</tr>
<tr>
<td>tot_scratch</td>
<td>500</td>
<td>host1, host2</td>
</tr>
</tbody>
</table>
```

The above output indicates that five licenses are available, and that the shared scratch directory currently contains 500 MB of space.

The `VALUE` field indicates the amount of that resource. The `LOCATION` column shows the hosts that share this resource. The `lshosts -s` command displays static shared resources. The `lsload -s` command displays dynamic shared resources.

### Viewing shared resources for hosts

Run `bhosts -s` to view shared resources for hosts. For example:

```bash
$ bhosts -s

<table>
<thead>
<tr>
<th>RESOURCE</th>
<th>TOTAL</th>
<th>RESERVED</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tot_lic</td>
<td>5</td>
<td>0.0</td>
<td>hostA, hostB</td>
</tr>
<tr>
<td>tot_scratch</td>
<td>00</td>
<td>0.0</td>
<td>hostA, hostB</td>
</tr>
<tr>
<td>avail_lic</td>
<td>2</td>
<td>3.0</td>
<td>hostA, hostB</td>
</tr>
<tr>
<td>avail_scratch</td>
<td>100</td>
<td>400.0</td>
<td>hostA, hostB</td>
</tr>
</tbody>
</table>
```

The `TOTAL` column displays the value of the resource. For dynamic resources, the `RESERVED` column displays the amount that has been reserved by running jobs.

### Viewing load on a host

Use `bhosts -l` to check the load levels on the host, and adjust the suspending conditions of the host or queue if necessary. The `bhosts -l` command gives the most recent load values used for the scheduling of jobs. A dash (-) in the output indicates that the particular threshold is not defined.

```bash
$ bhosts -l hostB

HOST: hostB

<table>
<thead>
<tr>
<th>STATUS</th>
<th>CPUF</th>
<th>JL/U</th>
<th>MAX NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>20.00</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

CURRENT LOAD USED FOR SCHEDULING:

<table>
<thead>
<tr>
<th>r15s</th>
<th>r1m</th>
<th>r15m</th>
<th>ut</th>
<th>pg</th>
<th>io</th>
<th>ls</th>
<th>t</th>
<th>tmp</th>
<th>swp</th>
<th>mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>0.3</td>
<td>0.8</td>
<td>0.9</td>
<td>61%</td>
<td>3.8</td>
<td>72</td>
<td>26</td>
<td>0</td>
<td>6M</td>
<td>253M</td>
</tr>
<tr>
<td>Reserved</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0%</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0M</td>
<td>0M</td>
</tr>
</tbody>
</table>

LOAD THRESHOLD USED FOR SCHEDULING:

<table>
<thead>
<tr>
<th>r15s</th>
<th>r1m</th>
<th>r15m</th>
<th>ut</th>
<th>pg</th>
<th>io</th>
<th>ls</th>
<th>it</th>
<th>tmp</th>
<th>swp</th>
<th>mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadSched</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>loadStop</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Viewing information about load indices

The `lsinfo -l` command displays all information available about load indices in the system. You can also specify load indices on the command line to display information about selected indices:

$ lsinfo -l swp
RESOURCES_NAME: swp
DESCRIPTION: Available swap space (Mbytes) (alias: swap)
TYPE ORDER INTERVAL BUILTIN DYNAMIC RELEASE
Numeric Dec 60 Yes Yes NO

Viewing resource requirements for a queue

Use `bqueues -l` to view resource requirements (RES_REQ) defined for the queue.

Viewing resource requirements for a job

Use `bjobs -l` to view resource requirements defined for the job:

After a job is finished, use `bhist -l` to view resource requirements defined for the job:
CHAPTER

3

Configuring Job Controls

Contents

◆ “Configuring Resource Usage Limits” on page 50
◆ “Configuring Load Thresholds” on page 53
◆ “Configuring Job Control Actions” on page 55
◆ “Configuring Pre-Execution and Post-Execution Commands” on page 58
◆ “Configuring Job Starters for Queues” on page 60
Configuring Resource Usage Limits

Resource usage limits control how much resource can be consumed by running jobs. Jobs that use more than the specified amount of a resource are signalled or have their priority lowered.

Limits can be specified either for a queue by the Lava administrator (lsb.queues) or for a job at submission time.

Limits specified at the queue level are hard limits, while those specified with job submission are soft limits.

Summary of resource usage limits

<table>
<thead>
<tr>
<th>Limit</th>
<th>Job syntax (bsub)</th>
<th>Queue syntax (lsb.queues)</th>
<th>Format/Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core file size limit</td>
<td>-C core_limit</td>
<td>CORELIMIT=limit</td>
<td>integer KB</td>
</tr>
<tr>
<td>CPU time limit</td>
<td>-c cpu_limit</td>
<td>CPULIMIT=[default]</td>
<td>[hours]:[minutes]/[host_name]/host_model</td>
</tr>
<tr>
<td>Data segment size limit</td>
<td>-D data_limit</td>
<td>DATALIMIT=[default]</td>
<td>integer KB</td>
</tr>
<tr>
<td>File size limit</td>
<td>-F file_limit</td>
<td>FILELIMIT=limit</td>
<td>integer KB</td>
</tr>
<tr>
<td>Memory limit</td>
<td>-M mem_limit</td>
<td>MEMLIMIT=[default]</td>
<td>integer KB</td>
</tr>
<tr>
<td>Run time limit</td>
<td>-W run_limit</td>
<td>RUNLIMIT=[default]</td>
<td>[hours]:[minutes]/[host_name]/host_model</td>
</tr>
<tr>
<td>Stack segment size limit</td>
<td>-S stack_limit</td>
<td>STACKLIMIT=limit</td>
<td>integer KB</td>
</tr>
<tr>
<td>Virtual memory limit</td>
<td>-v swap_limit</td>
<td>SWAPLIMIT=limit</td>
<td>integer KB</td>
</tr>
</tbody>
</table>

How resource usage limits are prioritized

If no limit is specified at job submission, then the following apply to all jobs submitted to the queue:

<table>
<thead>
<tr>
<th>If ...</th>
<th>Then ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both default and maximum limits are defined</td>
<td>The default is enforced</td>
</tr>
<tr>
<td>Only a maximum is defined</td>
<td>The maximum is enforced</td>
</tr>
<tr>
<td>No limit is specified in the queue or at job submission</td>
<td>No limits are enforced</td>
</tr>
</tbody>
</table>

Incorrect resource usage limits

Incorrect limits are ignored and a warning message is displayed when the cluster is reconfigured or restarted. A warning message is also logged to the mbatchd log file when Lava is started.

If no limit is specified at job submission, then the following apply to all jobs submitted to the queue:
Resource usage limits specified at job submission must be less than the maximum specified in `lsb.queues`.

### Specifying resource usage limits

Queues can enforce resource usage limits on running jobs. Lava supports most of the limits that the underlying operating system supports. In addition, Lava also supports a few limits that the underlying operating system does not support.

Specify queue-level resource usage limits using parameters in `lsb.queues`.

Limits configured in `lsb.queues` apply to all jobs submitted to the queue. Job-level resource usage limits specified at job submission override the queue definitions.

**Maximum value only**

Specify only a maximum value for the resource.

For example, to specify a maximum run limit, use one value for the `RUNLIMIT` parameter in `lsb.queues`:

```
RUNLIMIT = 10
```

The maximum run limit for the queue is 10 minutes. Jobs cannot run for more than 10 minutes. Jobs in the RUN state for longer than 10 minutes are killed by Lava.

If only one run limit is specified, jobs that are submitted with `bsub -W` with a run limit that exceeds the maximum run limit will not be allowed to run. Jobs submitted without `bsub -W` will be allowed to run but will be killed when they are in the RUN state for longer than the specified maximum run limit.

For example, in `lsb.queues`:

```
RUNLIMIT = 10
```

The maximum run limit for the queue is 10 minutes. Jobs cannot run for more than 10 minutes.

If you specify two limits, the first one is the default (soft) limit for jobs in the queue and the second one is the maximum (hard) limit. Both the default and the maximum limits must be positive integers. The default limit must be less than the maximum limit. The default limit is ignored if it is greater than the maximum limit.

Use the default limit to avoid having to specify resource usage limits in the `bsub` command.

For example, to specify a default and a maximum run limit, use two values for the `RUNLIMIT` parameter in `lsb.queues`:

```
RUNLIMIT = 10 15
```

- The first number is the default run limit applied to all jobs in the queue that are submitted without a job-specific run limit (without `bsub -W`).

<table>
<thead>
<tr>
<th>If ...</th>
<th>Then ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>The default limit is incorrect</td>
<td>The default is ignored and the maximum limit is enforced</td>
</tr>
<tr>
<td>Both default and maximum limits are specified and the maximum is incorrect</td>
<td>The maximum is ignored and the resource has no maximum limit—only a default limit</td>
</tr>
<tr>
<td>Both default and maximum limits are incorrect</td>
<td>The default and maximum are ignored and no limit is enforced</td>
</tr>
</tbody>
</table>

| Resource usage limits specified at job submission must be less than the maximum specified in `lsb.queues`. | |

---

**Default and maximum values**

Specifying only a maximum value for the resource.

For example, to specify a maximum run limit, use one value for the `RUNLIMIT` parameter in `lsb.queues`:

```
RUNLIMIT = 10
```

The maximum run limit for the queue is 10 minutes. Jobs cannot run for more than 10 minutes. Jobs in the RUN state for longer than 10 minutes are killed by Lava.

If only one run limit is specified, jobs that are submitted with `bsub -W` with a run limit that exceeds the maximum run limit will not be allowed to run. Jobs submitted without `bsub -W` will be allowed to run but will be killed when they are in the RUN state for longer than the specified maximum run limit.

For example, in `lsb.queues`:

```
RUNLIMIT = 10
```

The maximum run limit for the queue is 10 minutes. Jobs cannot run for more than 10 minutes.

If you specify two limits, the first one is the default (soft) limit for jobs in the queue and the second one is the maximum (hard) limit. Both the default and the maximum limits must be positive integers. The default limit must be less than the maximum limit. The default limit is ignored if it is greater than the maximum limit.

Use the default limit to avoid having to specify resource usage limits in the `bsub` command.

For example, to specify a default and a maximum run limit, use two values for the `RUNLIMIT` parameter in `lsb.queues`:

```
RUNLIMIT = 10 15
```

- The first number is the default run limit applied to all jobs in the queue that are submitted without a job-specific run limit (without `bsub -W`).
The second number is the maximum run limit applied to all jobs in the queue that are submitted with a job-specific run limit (with `bsub -W`). The default run limit must be less than the maximum run limit.

You can specify both default and maximum values for the following resource usage limits in `lsb.queues`:

- `CPULIMIT`
- `DATALIMIT`
- `MEMLIMIT`
- `PROCESSLIMIT`
- `RUNLIMIT`

**Host specification with two limits**

If default and maximum limits are specified for CPU time limits or run time limits, only one host specification is permitted. For example, the following CPU limits are correct (and have an identical effect):

- `CPULIMIT = 400/hostA 600`
- `CPULIMIT = 400 600/hostA`

The following CPU limit is incorrect:

`CPULIMIT = 400/hostA 600/hostB`

To specify resource usage limits for a job, see Running Jobs with Platform Lava.

**Setting the CPU time and run time limits**

To set the CPU time limit and run time limit for jobs in a platform-independent way, Lava scales the limits by the CPU factor of the hosts involved. When a job is dispatched to a host for execution, the limits are then normalized according to the CPU factor of the execution host.

Whenever a normalized CPU time or 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.

**Normalization host**

If no host or host model is given with the CPU time or run time, Lava selects a host as follows (in order of preference):

- The default CPU time normalization host if defined at the queue level (`DEFAULT_HOST_SPEC` in `lsb.queues`)
- The default CPU time normalization host if defined at the cluster level (`DEFAULT_HOST_SPEC` in `lsb.params`)
- The submission host

**Example**

`CPULIMIT=10/hostA`

If `hostA` has a CPU factor of 2, and `hostB` has a CPU factor of 1 (`hostB` is slower than `hostA`), this specifies an actual time limit of 10 minutes on `hostA`, or on any other host that has a CPU factor of 2. However, if `hostB` is the execution host, the actual time limit on `hostB` is 20 minutes (10 * 2 / 1).
Configuring Load Thresholds

You can configure load thresholds to schedule jobs in queues. Load thresholds specify a load index value. There are two types of load thresholds:

- **loadSched**
  The scheduling threshold which determines the load condition for dispatching pending jobs.

- **loadStop**
  The condition that determines when running jobs should be suspended.

Thresholds can be configured for each queue, for each host, or a combination of both. The value of a load index may either increase or decrease with load, depending on the meaning of the specific load index. Therefore, when comparing the host load conditions with the threshold values, you need to use either greater than (>) or less than (<), depending on the load index.

The queue definition (lsb.queues) can contain thresholds for 0 or more of the load indices. Any load index that does not have a configured threshold has no effect on job scheduling.

**Syntax**

Each load index is configured on a separate line with the format:

```
load_index = loadSched/loadStop
```

Specify the name of the load index, for example, r1m for the 1-minute CPU run queue length or pg for the paging rate. loadSched is the scheduling threshold for this load index. loadStop is the suspending threshold. The loadSched condition must be satisfied by a host before a job is dispatched to it and also before a job suspended on a host can be resumed. If the loadStop condition is satisfied, a job is suspended.

The loadSched and loadStop thresholds permit the specification of conditions using simple AND/OR logic. For example, the specification:

```
MEM=100/10
SWAP=200/30
```

translates into a loadSched condition of `mem>=100 && swap>=200` and a loadStop condition of `mem < 10 || swap < 30`.

**Theory**

- The r15s, r1m, and r15m CPU run queue length conditions are compared to the effective queue length as reported by lsswap -E, which is normalized for multiprocessor hosts. Thresholds for these parameters should be set at appropriate levels for single processor hosts.

- Configure load thresholds consistently across queues. If a low priority queue has higher suspension thresholds than a high priority queue, then jobs in the higher priority queue will be suspended before jobs in the low priority queue.
Configuring Load Thresholds

**Configuring suspending conditions**

The condition for suspending a job can be specified using the queue-level STOP_COND parameter. It is defined by a resource requirement string. Only the select section of the resource requirement string is considered when stopping a job. All other sections are ignored.

This parameter provides similar but more flexible functionality for loadStop.

If loadStop thresholds have been specified, then a job will be suspended if either the STOP_COND is TRUE or the loadStop thresholds are exceeded.

**Example**

This queue will suspend a job based on the idle time for desktop machines and based on availability of swap and memory on compute servers. Assume `cs` is a Boolean resource defined in the `lsf.shared` file and configured in the `lsf.cluster.lava` file to indicate that a host is a compute server:

```
Begin Queue

STOP_COND = select[((!cs && it < 5) || (cs && mem < 15 && swap < 50))]

End Queue
```
Configuring Job Control Actions

After a job is started, it can be killed, suspended, or resumed by the system, a Lava user, or Lava administrator. Lava job control actions cause the status of a job to change. Several situations may require overriding or augmenting the default actions for job control. For example:

- Notifying users when their jobs are suspended, resumed, or terminated
- An application holds resources (for example, licenses) that are not freed by suspending the job. The administrator can set up an action to be performed that causes the license to be released before the job is suspended and re-acquired when the job is resumed.
- The administrator wants the job checkpointed before it is:
  - Suspended when a run window closes
  - Killed when the RUNLIMIT is reached

To override the default actions for the SUSPEND, RESUME, and TERMINATE job controls, specify the JOB_CONTROLS parameter in the queue definition in `lsb.queues`.

Configuring job controls

The JOB_CONTROLS parameter in `lsb.queues` has the following format:

```
Begin Queue

JOB_CONTROLS = SUSPEND[signal | CHKPNT | command] \n                RESUME[signal | command] \n                TERMINATE[signal | CHKPNT | command]

End Queue
```

When Lava needs to suspend, resume, or terminate a job, it invokes one of the following actions as specified by SUSPEND, RESUME, and TERMINATE.

- **signal**: A UNIX signal name (for example, SIGTSTP or SIGTERM). The specified signal is sent to the job.
  
  To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the `kill -l` command.

- **CHKPNT**: Checkpoint the job. Only valid for SUSPEND and TERMINATE actions.
  - If the SUSPEND action is CHKPNT, the job is checkpointed and then stopped.
  - If the TERMINATE action is CHKPNT, then the job is checkpointed and killed.

- **command**: A `/bin/sh` command line. Do not quote the command line inside an action definition.
  
  See the Platform Lava man pages for information about the job control parameters in the `lsb.queues` file.
**TERMINATE job actions**

Use caution when configuring TERMINATE job actions that do more than just kill a job. For example, resource usage limits that terminate jobs change the job state to SSUSP while Lava waits for the job to end. If the job is not killed by the TERMINATE action, it remains suspended indefinitely.

**TERMINATE_WHEN parameter**

In certain situations you may want to terminate the job instead of calling the default SUSPEND action. For example, you may want to kill jobs if the run window of the queue is closed. Use the TERMINATE_WHEN parameter in `lsb.queues` to configure the queue to invoke the TERMINATE action instead of SUSPEND.

See the Platform Lava man pages for information about the TERMINATE_WHEN parameter in the `lsb.queues` file.

**Syntax**

TERMINATE_WHEN = WINDOW | LOAD

**Example**

The following defines a night queue that will kill jobs if the run window closes.

```
Begin Queue
  NAME = night
  RUN_WINDOW = 20:00-08:00
  TERMINATE_WHEN = WINDOW
  JOB_CONTROLS = TERMINATE[ kill -KILL $LSB_JOBPIDS;
    echo "job $LSB_JOBID killed by queue run window" |
    mail $USER ]
End Queue
```

**Using a command as a job control action**

The following apply to a job control action that is a command:

- The command line for the action is run with `/bin/sh -c` so you can use shell features in the command.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action.
- The following additional environment variables are set:
  - `LSB_JOBPGIDS` — a list of current process group IDs of the job
  - `LSB_JOBPIDS` — a list of current process IDs of the job
- For the SUSPEND action command, the following environment variable is also set:
  - `LSB_SUSP_REASONS` — an integer representing a bitmap of suspending reasons as defined in `lsbatch.h`.

The suspending reason can allow the command to take different actions based on the reason for suspending the job.

- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly or not. You should make sure the command line is correct. If you want to see the output from the command line for testing purposes, redirect the output to a file inside the command line.
**LSB_SIGSTOP parameter**

Use LSB_SIGSTOP in lsf.conf to configure the SIGSTOP signal sent by the default SUSPEND action.

If LSB_SIGSTOP is set to anything other than SIGSTOP, the SIGTSTP signal that is normally sent by the SUSPEND action is not sent. For example, if LSB_SIGSTOP=SIGKILL, the three default signals sent by the TERMINATE action (SIGINT, SIGTERM, and SIGKILL) are sent 10 seconds apart.

See the Platform Lava man pages for information about LSB_SIGSTOP in the lsf.conf file.

**Avoiding signal and action deadlock**

Do not configure a job control to contain the signal or command that is the same as the action associated with that job control. This will cause a deadlock between the signal and the action.

For example, the bkill command uses the TERMINATE action, so a deadlock results when the TERMINATE action itself contains the bkill command.

Any of the following job control specifications will cause a deadlock:

- JOB_CONTROLS=TERMINATE[bkill]
- JOB_CONTROLS=TERMINATE[brequeue]
- JOB_CONTROLS=RESUME[bresume]
- JOB_CONTROLS=SUSPEND[bstop]
Configuring Pre-Execution and Post-Execution Commands

Pre- and post-execution commands can be configured at the job level or on a per-queue basis.

Job-level commands

Job-level pre-execution commands require no configuration. Use the `bsub -E` option to specify an arbitrary command to run before the job starts.

Queue-level commands

Use the PRE_EXEC and POST_EXEC keywords in the queue definition (`lsb.queues`) to specify pre- and post-execution commands.

The following points should be considered when setting up pre- and post-execution commands at the queue level:

- If the pre-execution command exits with a non-zero exit code, then it is considered to have failed and the job is requeued to the head of the queue. This feature can be used to implement customized scheduling by having the pre-execution command fail if conditions for dispatching the job are not met.
- Other environment variables set for the job are also set for the pre- and post-execution commands.
- When a job is dispatched from a queue that has a pre-execution command, Lava will remember the post-execution command defined for the queue from which the job is dispatched. If the job is later switched to another queue or the post-execution command of the queue is changed, Lava will still run the original post-execution command for this job.
- When the post-execution command is run, the environment variable, `LSB_JOBEXIT_STAT`, is set to the exit status of the job. See the man page for the `wait(2)` command for the format of this exit status.
- The post-execution command is also run if a job is requeued because the job’s execution environment fails to be set up, or if the job exits with one of the queue’s `REQUEUE_EXIT_VALUES`. (See “Configuring automatic job requeue” on page 20.)
  
  The `LSB_JOBPEND` environment variable is set if the job is requeued. If the job’s execution environment could not be set up, `LSB_JOBEXIT_STAT` is set to 0. See “Automatic Job Requeue” on page 329 for more information.
- If both queue and job-level pre-execution commands are specified, the job-level pre-execution is run after the queue-level pre-execution command.

The entire contents of the configuration line of the pre- and post-execution commands are run under `/bin/sh -c`, so shell features can be used in the command.

For example, the following is valid:

```
PRE_EXEC = /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
POST_EXEC = /usr/share/lsf/misc/testq_post | grep -v "Hey!"
```

The pre- and post-execution commands are run in `/tmp`. Standard input and standard output and error are set to `/dev/null`. The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.
The PATH environment variable is set to:

```
PATH='/bin /usr/bin /sbin /usr/sbin'
```

Begin Queue

```
QUEUE_NAME = priority
PRIORITY = 43
NICE = 10
PRE_EXEC = /usr/share/lsf/pri_preexec
POST_EXEC = /usr/share/lsf/pri_postexec
```

End Queue

**LSB_PRE_POST_EXEC_USER parameter**

By default, both the pre- and post-execution commands are run as the job submission user. Use the `LSB_PRE_POST_EXEC_USER` parameter in `lsf.sudoers` to specify a different user ID for queue-level pre- and post-execution commands.

**Example**

For example, if the pre- or post-execution commands perform privileged operations that require root permission, specify:

```
LSB_PRE_POST_EXEC_USER=root
```
Configuring Job Starters for Queues

Lava administrators can define a job starter for an individual queue to create a specific environment for jobs to run in. A queue-level job starter specifies an executable that performs any necessary setup, and then runs the job when the setup is complete. The JOB_STARTER parameter in lsb.queues specifies the command or script that is the job starter for the queue.

Queue-level job starters have no effect on interactive jobs, unless the interactive job is submitted to a queue as an interactive batch job.

Configuring a queue-level job starter

Use the JOB_STARTER parameter in lsb.queues to specify a queue-level job starter in the queue definition. All jobs submitted to this queue are run using the job starter. The jobs are called by the specified job starter process rather than initiated by the batch daemon process.

For example:

```
Begin Queue

JOB_STARTER = xterm -e

End Queue
```

All jobs submitted to this queue are run under an xterm terminal emulator.

JOB_STARTER parameter

The JOB_STARTER parameter in the queue definition (lsb.queues) has the following format:

```
JOB_STARTER = starter [starter] [%USRCMD] [starter]
```

The string starter is the command or script that is used to start the job. It can be any executable that can accept a job as an input argument. Optionally, additional strings can be specified.

When starting a job, Lava runs the JOB_STARTER command, and passes the shell script containing the job commands as the argument to the job starter. The job starter is expected to do some processing and then run the shell script containing the job commands. The command is run under /bin/sh -c and can contain any valid Bourne shell syntax.

%USRCMD string

The special string %USRCMD indicates the position of the job starter command in the job command line. By default, the user commands run after the job starter, so the %USRCMD string is not usually required. For example, these two job starters both give the same results:

```
JOB_STARTER = /bin/csh -c
JOB_STARTER = /bin/csh -c %USRCMD
```

You can also enclose the %USRCMD string in quotes or follow it with additional commands. For example:

```
JOB_STARTER = /bin/csh -c "%USRCMD;sleep 10"
```

If a user submits the following job to the queue with this job starter:
$ bsub myjob arguments
the command that actually runs is:
$ /bin/csh -c "myjob arguments; sleep 10"
Symbols
%USRCMD string in job starters 60
/etc/hosts file 17
/etc/syslog.conf file 24
/usr/include/sys/syslog.h file 24
~ (tilde), not operator, host-based resources 38

A
administrators
  cluster administrator description 12
  displaying queue administrators 28
  primary Lava administrator 12
ADMINISTRATORS parameter in lsf.cluster.lava 12
aliases
  for resource names 43
  using as host names 16

B
badmin command
  hopen 14
  hrestart 9
  hshutdown 9
  hstartup 9
  mbdrestart 9
  qact 20
  qclose 19
  qinact 20
  qopen 20
  reconfig 9
batch jobs
  email about jobs, options 13
  input and output 13
Berkeley Internet Name Domain (BIND), host naming 16
BIND (Berkeley Internet Name Domain), host naming 16
Boolean resources 30, 31
bsub command
  email job notification 13
  input and output 13
built-in, resources 30
busy host status
  lsload command 26
  status load index 32

C
ceiling resource usage limit 51
ceiling run limit 51
closed host status, bhosts command 26, 27
closed_Adm condition, output of bhosts -l 26
closed_Busy condition, output of bhosts -l 26
closed_Full condition, output of bhosts -l 26
closed_LIM condition, output of bhosts -l 26
closed_Lock condition, output of bhosts -l 26
closed_Wind condition, output of bhosts -l 26
cluster administrators
  description 12
  viewing 25
cluster name 7
  viewing 25
cluster-fork command, for running the same command on multiple hosts 9
clusters
  configuration file quick reference 10
  reconfiguring, commands 10
  viewing information 25
commands
  running under user ID 59
  using in job control actions 56
configuration
  adding and removing, queues 25
  viewing, errors 25
configuration files, reconfiguration quick reference 10
CPU
  factors
    static resource 34
    time normalization 52
    normalizing 52
    utilization, ut load index 33
cpuf static resource 34
custom resources
  adding 36
  configuring 36
  description 36
  resource types 30

D
Daemon Error Log 24
daemons
  error logs 24
  restarting
    mbatchd 10
    sbatchd 9
  shutting down, sbatchd 9
  TCP service ports 15
  ypbind 16
deadlock, avoiding signal and job action 57
default
  LSF_LOGDIR 24
  resource usage limits 51
Index

DEFAULT_HOST_SPEC parameter, in lsb.queues 52
directories, log, permissions and ownership 23
disks, I/O rate 33
dispatch windows, hosts 14
DISPATCH_WINDOW, queues 22
Domain Name Service (DNS), host naming 16
dynamic, resources 30
E
effective run queue length, built-in resources 33
ELIM (external LIM)
custom resources 39
debugging 41
email
job options 13
limiting the size of job email 13
error logs 24
Lava daemons 24
log directory, LSF_LOGDIR 24
log files 24
LSF_LOG_MASK parameter 24
managing log files 24
errors, viewing in reconfiguration 25
/etc/hosts file
host naming 16
name lookup 16
/etc/syslog.conf file 24
examples, /etc/hosts file entries 17
execution, priority 34
external, load indices, using ELIM 39
F
files
/etc/hosts
example host entries 17
host naming 16
name lookup 16
hosts, configuring 17
lsf.conf
configuring TCP service ports 15
daemon service ports 15
lsf.shared, adding a custom host types and models 15
resolv.conf 16
free memory 33
G
gethostbyname function (host naming) 16
H
hard resource limits, description 50
hard resource usage limits, example 51
hname static resource 34
hopen badmin command 14
host entries, examples 17
host model, select string 43
host model static resource 34
host models, adding custom names in lsf.shared 15
host name static resource 34
host names
/etc/hosts file 16
aliases 16
matching with Internet addresses 16
resolv.conf file 16
resolver function 16
using DNS 16
host selection 43
host status
busy 26, 32
closed 26, 27
index 32
lockU and lockW 26, 32
-ok 26, 32
-ok 26, 32
unavail 26, 32
unreach 26
viewing 26
host type
resource requirements 42
select string 43
host type static resource 34
host types 7
adding custom names in lsf.shared 15
host-based resources 30
hosts
associating resources with 37
closing 14
controlling 14
dispatch windows 14
displaying 26
file 16
multiple network interfaces 16
official name 16
opening 14
viewing
detailed information 27
history 27
hosts and host status 26
load by host 27, 47
load by resource 47
model and type information 27
shared resources 47
status of closed servers 27
hosts file
configuring 17
example host entries 17
host naming 16
hrestart badmin command 9
hshutdown badmin command 9
hstartup badmin command 9
I
idle time, built-in load index 33
interfaces, network 16
Internet addresses, matching with host names 16
Internet Domain Name Service (DNS), host naming 16
io load index 33
it load index 33
J
job control actions
CHKPNT 55
terminating 56
using commands in 56
job email
bsub options 13
limiting size with LSB_MAILSIZE_LIMIT 13
job starters
queue-level, configuring 60
specifying command or script 60
user commands 60
JOB_CONTROLS parameter in lsb.queues 55
JOB_STARTER parameter in lsb.queues 60
job-level
pre-execution commands, configuring 58
resource requirements 42
jobs
CHKPNT 55
email notification, options 13
enabling rerun 21
submitting, resources 31
suspending at queue level 54
L
Lava version, viewing 25
LIM (Load Information Manager)
configuring number of restarts 41
configuring wait time 41
logging load information 41
lim.log.host_name file 24
limits, See resource allocation limits or resource usage limits
load average 32
load indices
built-in, summary 31
io 33
it 33
mem 33
pg 33
r15m 32
r15s 32
r1m 32
swp 33
tmp 33
ut 33
viewing 25, 48
load levels
viewing by resource 47
viewing for cluster 25
viewing for hosts 27
load thresholds
configuring 53
description 53
queue level 53
lockU and lockW host status
lsload command 26
status load index 32
log files
directory permissions and ownership 23
lim.log.host_name 24
maintaining 24
managing 24
mbatchd.log.host_name 24
mbsched.log.host_name 24
pim.log.host_name 24
res.log.host_name 24
sbatchd.log.host_name 24
lost_and_found queue 19
lsadmin command
limlock 26
limunlock 26
lsb.params file, controlling lsb.events file rewrites 23
lsb.queues file
adding a queue 19
normalization host 52
resource usage limits 51
LSB_JOBEXIT_STAT environment variable 58
LSB_JOBPEND environment variable 58
LSB_JOBPGIDS environment variable 56
LSB_JOBPIDS environment variable 56
LSB_MAILSIZE_LIMIT parameter in lsf.conf 13
LSB_MBD_PORT parameter in lsf.conf 16
LSB_SBD_PORT parameter in lsf.conf 16
LSB_SIGSTOP parameter in lsf.conf 57
LSB_SUSP_REASON environment variable 56
lsf.cluster.lava file, configuring cluster administrators 12
lsf.conf file
configuring TCP service ports 15
daemon service ports 15
limiting the size of job email 13
managing error logs 24
lsf.shared file, adding a custom host type and model 15
LSF_LIM_PORT parameter in lsf.conf 16
LSF_LOG_MASK parameter in lsf.conf 24
LSF_LOGDIR parameter in lsf.conf 24
LSF_RES_PORT parameter in lsf.conf 16
LSF_RSH parameter in lsf.conf, controlling daemons 9
M
mail
job options 13
limiting the size of job email 13
master host 7
viewing current 25
MAX_JOB_NUM parameter in lsb.params 23
maximum
resource usage limit 51
run limit 51
maxmem static resource 34
maxswp static resource 34
maxtmp static resource 34
mbatchd (master batch daemon), restarting 10
mbatchd.log.host_name file 24
mbdrestart badmin command 9
mbsched.log.host_name file 24
mem load index 33
memory, available 33
model static resource 34
multi-homed hosts 16
multiprocessor hosts, configuring queue-level load thresholds 53
Index

N
name lookup using /etc/hosts file 16
ncpus static resource, reported by LIM 34
ndisks static resource 34
network, interfaces 16
NIS (Network Information Service)
  host name lookup in Lava 16
  ypcat hosts.byname 16
normalization
  CPU time limit 52
  host 52
  run time limit 52
normalized run queue length, description 33
not operator (~), host-based resources 38
numerical resources 30

O
official host name 16
  - ok host status
    lsload command 26
    status load index 32
  ok host status
    bhosts command 26
    lsload command 26
    status load index 32
operators
  not (~), host-based resources 38
  resource requirements 44
  selection strings 44
order string 45
ownership of log directory 23

P
paging rate
  description 33
  load index 33
paths
  /etc/hosts file
    host naming 16
    name lookup 16
periodic tasks 24
permissions, log directory 23
pim.log.host_name file 24
ports, registering daemon services 15
post-execution commands, running under user ID 59
pre-execution commands, running under user ID 59

Q
qact badmin command 20
qclose badmin command 19
qinact badmin command 20
qopen badmin command 20
queue administrators, displaying 28
QUEUE_NAME parameter in lsb.queues 19
queue-level
  job starter 60
  pre- and post-execution commands, configuring 58
  resource limits 51
  resource requirements 42
  run limits 52
  queue-level resource limits, defaults 51
queues
  adding and removing 25
  configuring
    job control actions 55
    suspending conditions 54
  displaying queue administrators 28
  lost_and_found 19
  REQUEUE_EXIT_VALUES parameter 58
  setting rerun level 21
  specifying suspending conditions 54
  viewing
    available 28
    detailed queue information 28
    history 28
    status 28
  viewing administrator of 25

R
- R res_req command argument 43
  r15m load index, built-in resources 32
  r15s load index, built-in resources 32
  r1m load index, built-in resources 32
  reconfig badmin command 9
  remote jobs, execution priority 34
  REQUEUE_EXIT_VALUES parameter in lsb.queues 20
  RERUNNABLE parameter in lsb.queues 21
  res.log.host_name file 24
  resolv.conf file 16
  resolver function 16
  resource names
    aliases 43
    description 36
  resource requirements
    description 42
    host type 42
    operators 44
    ordering hosts 43, 45
    select string 43
    selecting hosts 43
  resource usage limits
    ceiling 51
    conflicting 50
    default 51
    hard 51
    maximum 51
    priority 50
    soft 51
  ResourceMap section in lsf.cluster.cluster_name 37
  resources
    adding 36
    adding custom 36
    and job submission 31
    associating with hosts 37
    Boolean 31
    built-in 31
    configuring custom 36
    custom 36
    shared 31, 47
    types 30
Index

viewing
available 25, 46
host load 47
shared 25
rexpri static resource 34
rsh command, lsfrestart 9
run limits
ceiling 51
configuring 50
maximum 51
run queue
effective 33
normalized 33
run time, normalization 52
RUN_WINDOW, queues 22

S
sample /etc/hosts file entries 17
sbatchd (slave batch daemon)
   restarting 9
   shutting down 9
sbatchd.log.host_name file 24
scheduling, threshold, queue-level resource requirements 53
selection string 43
selection strings, operators 44
server hosts, viewing detailed information 27
server status closed 27
service ports (TCP and UDP), registering 15
shared resources
description 31
viewing 47
signals
   avoiding job action deadlock 57
   configuring SIGSTOP 57
SIGSTOP signal, configuring 57
soft resource limits
description 50
example 51
static resources
   See also individual resource names
description 34
STATUS, bhosts 26

status
closed in bhosts 27
load index 32
viewing
   hosts 26
   queues 28
string resources 30
Sun Network Information Service/ Yellow Pages. See NIS
suspending conditions, configuring 54
swap space, load index 33
swp load index, description 33
syslog.h file 24

T
TCP service port numbers, registering for Lava 15
TERMINATE_WHEN parameter in lsb.queues, changing default SUSPEND action 56
tilde (~), not operator, host-based resources 38
time normalization, CPU factors 52
tmp, load index, description 33
/usr/include/sys/syslog.h file 24
type static resource 34

U
UDP service port numbers, registering for Lava 15
unavail host status
   bhosts command 26
   lsload command 26
status load index, status load index 32
unreach host status, bhosts command 26
/usr/include/sys/syslog.h file 24
%USRCMD string in job starters 60

ut load index, built-in resource 33

V
viewing, configuration errors 25
virtual memory, load index 33
vmstat 33

Y
ypbind daemon 16
ypcat hosts.byname 16