Platform LSF Foundations

Platform LSF™
Version 8.0
June 2011

Contents

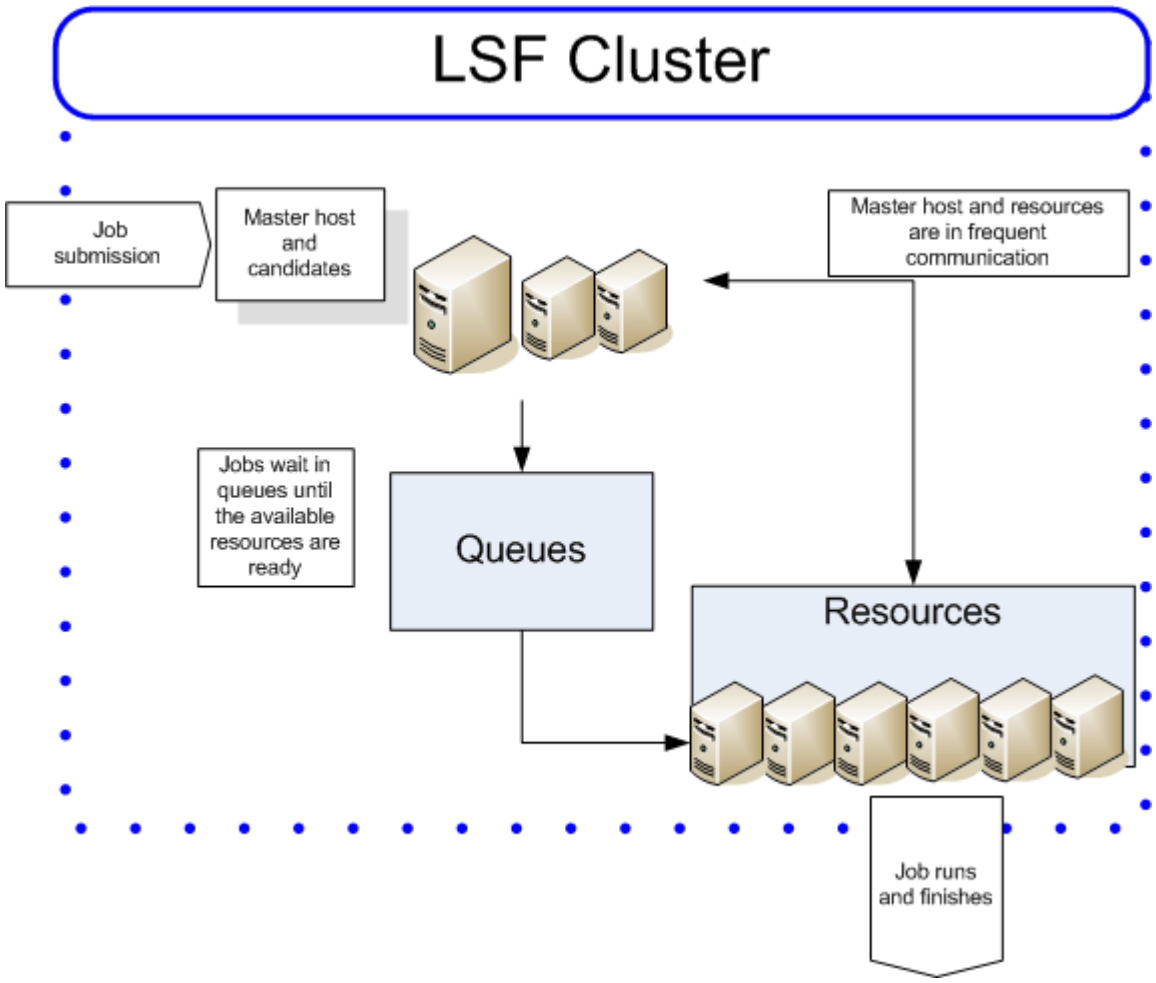
CHAPTER 1

Platform LSF: An Overview

Introduction to Platform LSF

The Platform LSF ("LSF", short for load sharing facility) software is leading enterprise-class software that distributes work across existing heterogeneous IT resources creating a shared, scalable, and fault-tolerant infrastructure, delivering faster, more reliable workload performance while reducing cost. LSF balances load and allocates resources, while providing access to those resources.

LSF provides a resource management framework that takes your job requirements, finds the best resources to run the job, and monitors its progress. Jobs always run according to host load and site policies.



Cluster

A group of computers (hosts) running LSF that work together as a single unit, combining computing power, workload, and resources. A cluster provides a single-system image for a network of computing resources.

Hosts can be grouped into a cluster in a number of ways. A cluster could contain:

- All the hosts in a single administrative group
- All the hosts on a sub-network

- Hosts that have required hardware

Hosts

Your cluster's hosts perform different functions.

- Master host: An LSF server host that acts as the overall coordinator for the cluster, doing all job scheduling and dispatch.
- Server host: A host that submits and executes jobs.
- Client host: A host that only submits jobs and tasks.
- Execution host: A host that executes jobs and tasks.
- Submission host: A host from which jobs and tasks are submitted.

Job

A unit of work run in the LSF system. A job is a command submitted to LSF for execution. LSF schedules, controls, and tracks the job according to configured policies.

Jobs can be complex problems, simulation scenarios, extensive calculations, or anything that needs compute power.

Job slot

A job slot is a bucket into which a single unit of work is assigned in the LSF system.

Hosts can be configured with multiple job slots and you can dispatch jobs from queues until all the job slots are filled. You can correlate job slots with the total number of CPUs in the cluster.

Queue

A cluster-wide container for jobs. All jobs wait in queues until they are scheduled and dispatched to hosts.

Queues do not correspond to individual hosts; each queue can use all server hosts in the cluster, or a configured subset of the server hosts.

When you submit a job to a queue, you do not need to specify an execution host. LSF dispatches the job to the best available execution host in the cluster to run that job.

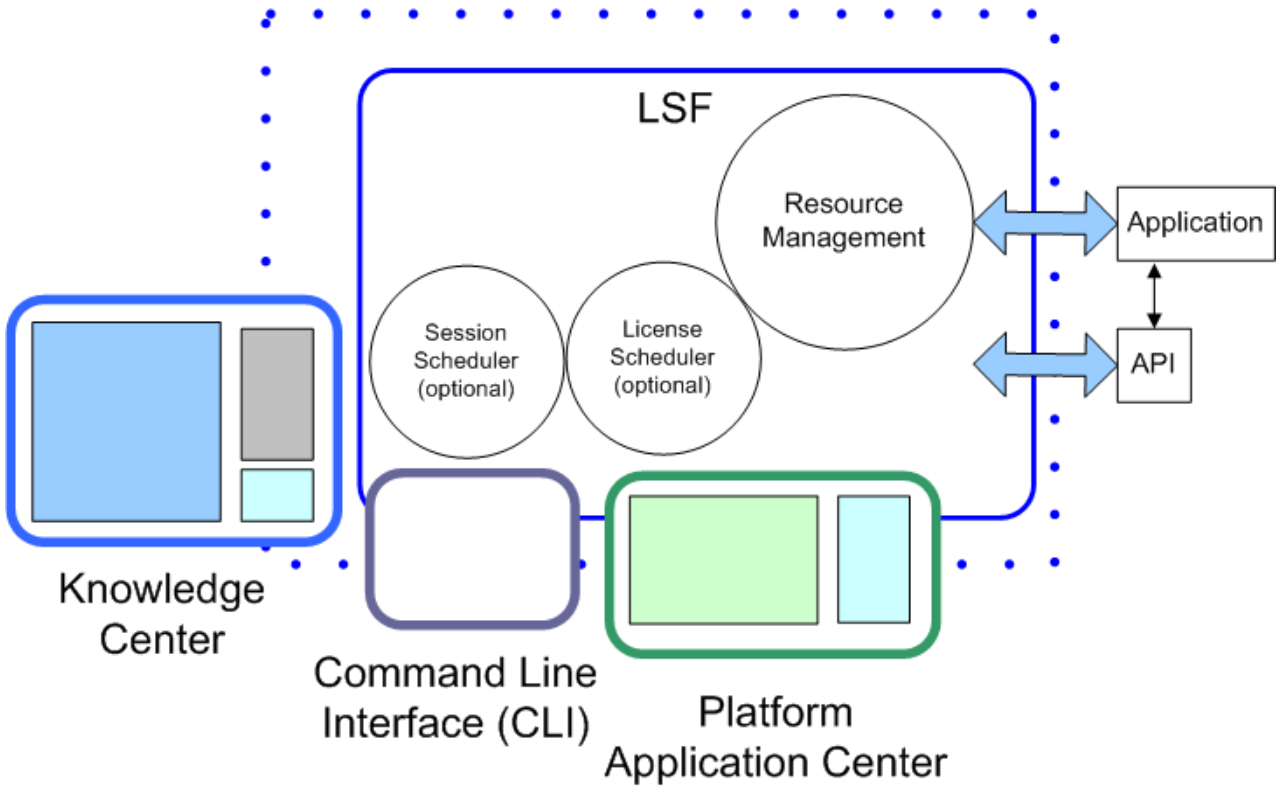
Queues implement different job scheduling and control policies.

Resources

Resources are the objects in your cluster that are available to run work. For example, resources include but are not limited to machines, CPU slots, and licenses.

Platform LSF cluster components

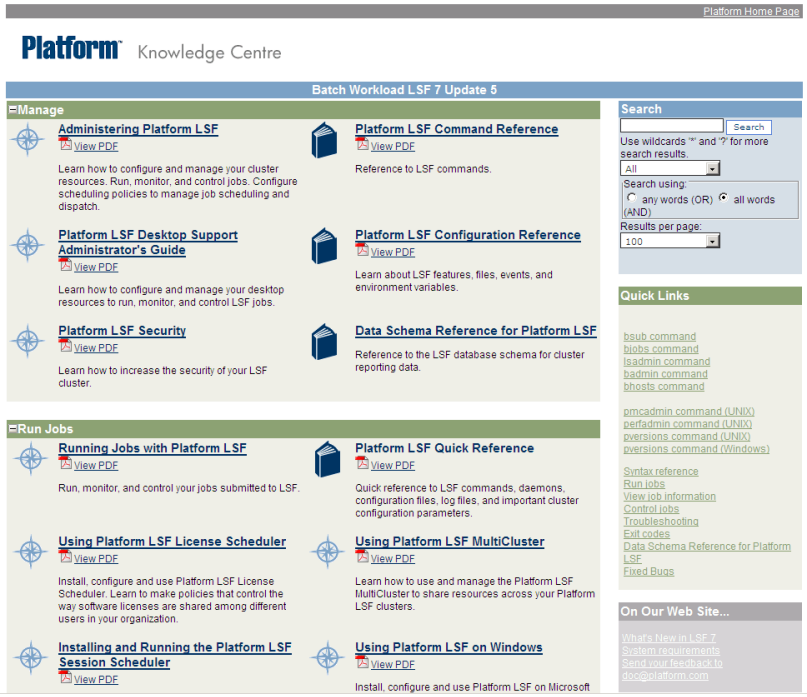
An LSF cluster manages resources, accepts and schedules workload, and monitors all events. LSF can be accessed by users and administrators by a command-line interface, an API, or through the Platform Application Center.



Platform LSF

- Core: The core of LSF includes daemons and functionality that schedules and runs jobs, as well as managing resources.
- License Scheduler: Platform LSF License Scheduler allows you to make policies that control the way software licenses are shared among different users in your organization. Platform LSF License Scheduler works with FlexNet™ products to control and monitor license usage.
- Session Scheduler: While traditional Platform LSF job submission, scheduling, and dispatch methods such as job arrays or job chunking are well suited to a mix of long and short running jobs, or jobs with dependencies on each other, Session Scheduler is ideal for large volumes of independent jobs with short run times.

Knowledge Center



The Knowledge Center is your access point to LSF documentation. It is provided with the LSF installation files and once extracted it can be accessed from any web browser. It can also be linked to directly from the Platform Application Center.

The Knowledge Center provides an overview of the organization of the product documentation. It also provides quick access to each document and links to some key resources, such as my.platform.com, your eSupport site.

In addition to links to all documents, the Knowledge Center provides full search capabilities within the documentation. You can perform keyword searches within a document or across the full documentation set.

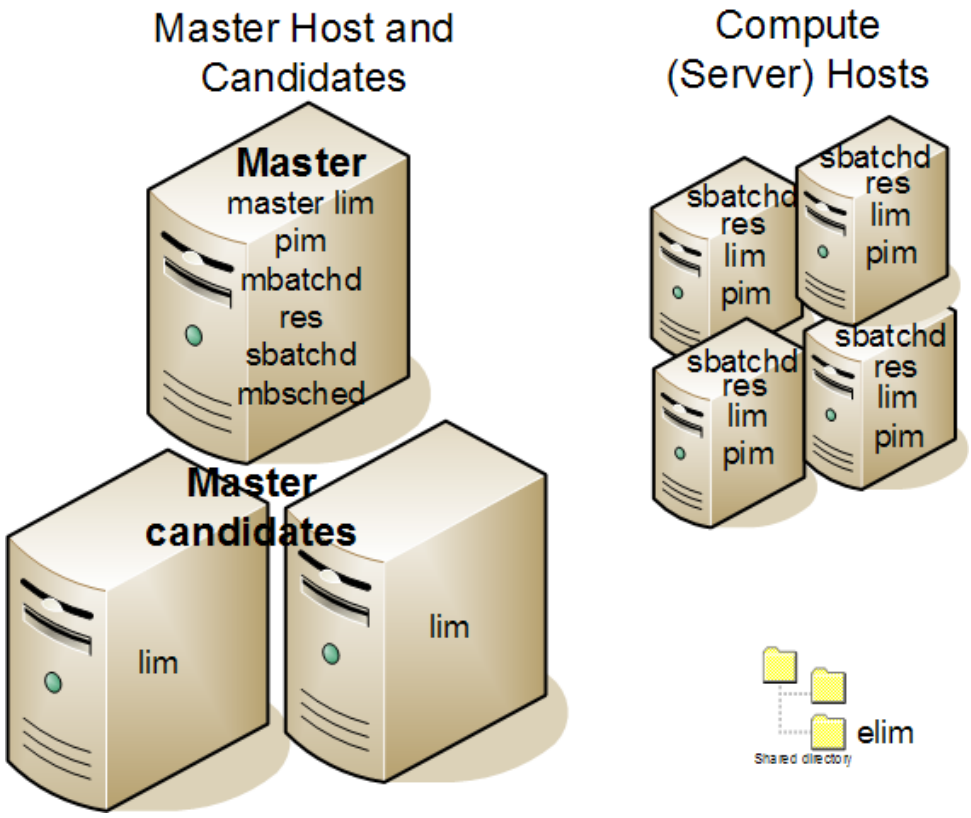
CHAPTER

2

Inside a Platform LSF Cluster

Platform LSF processes

There are multiple LSF processes running on each host in the cluster. The type and number of processes running depends on whether the host is a master host or a compute host.



Master host processes

LSF hosts run various processes, depending on their role in the cluster.

LSF daemon	Role
mbatchd	Job requests and dispatch
mbschd	Job scheduling
sbatchd	Job execution
res	Job execution
lim	Host information
pim	Job process information
elim	Dynamic load indices

mbatchd

Master Batch Daemon running on the master host. 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`.

Makes scheduling decisions based on job requirements, policies, and resource availability. Sends scheduling decisions to the `mbatchd`.

sbatchd

Slave Batch Daemon running on each server host including the master host. Receives the request to run the job from `mbatchd` and manages local execution of the job. Responsible for enforcing local policies and maintaining the state of jobs on the host.

`sbatchd` forks a child `sbatchd` for every job. The child `sbatchd` runs an instance of `res` to create the execution environment in which the job runs. The child `sbatchd` exits when the job is complete.

res

Remote Execution Server (RES) running on each server host. Accepts remote execution requests to provide transparent and secure remote execution of jobs and tasks.

lim

Load Information Manager (LIM) running on each server host. Collects host load and configuration information and forwards it to the master LIM running on the master host. Reports the information displayed by `lsload` and `lshosts`.

Static indices are reported when the LIM starts up or when the number of CPUs (`ncpus`) change.

Master lim

The LIM running on the master host. Receives load information from the LIMs running on hosts in the cluster.

Forwards load information to `mbatchd`, which forwards this information to `mbschd` to support scheduling decisions. If the master LIM becomes unavailable, a LIM on a master candidate automatically takes over.

pim

Process Information Manager (PIM) running on each server host. Started by LIM, which periodically checks on PIM and restarts it if it dies.

Collects information about job processes running on the host such as CPU and memory used by the job, and reports the information to `sbatchd`.

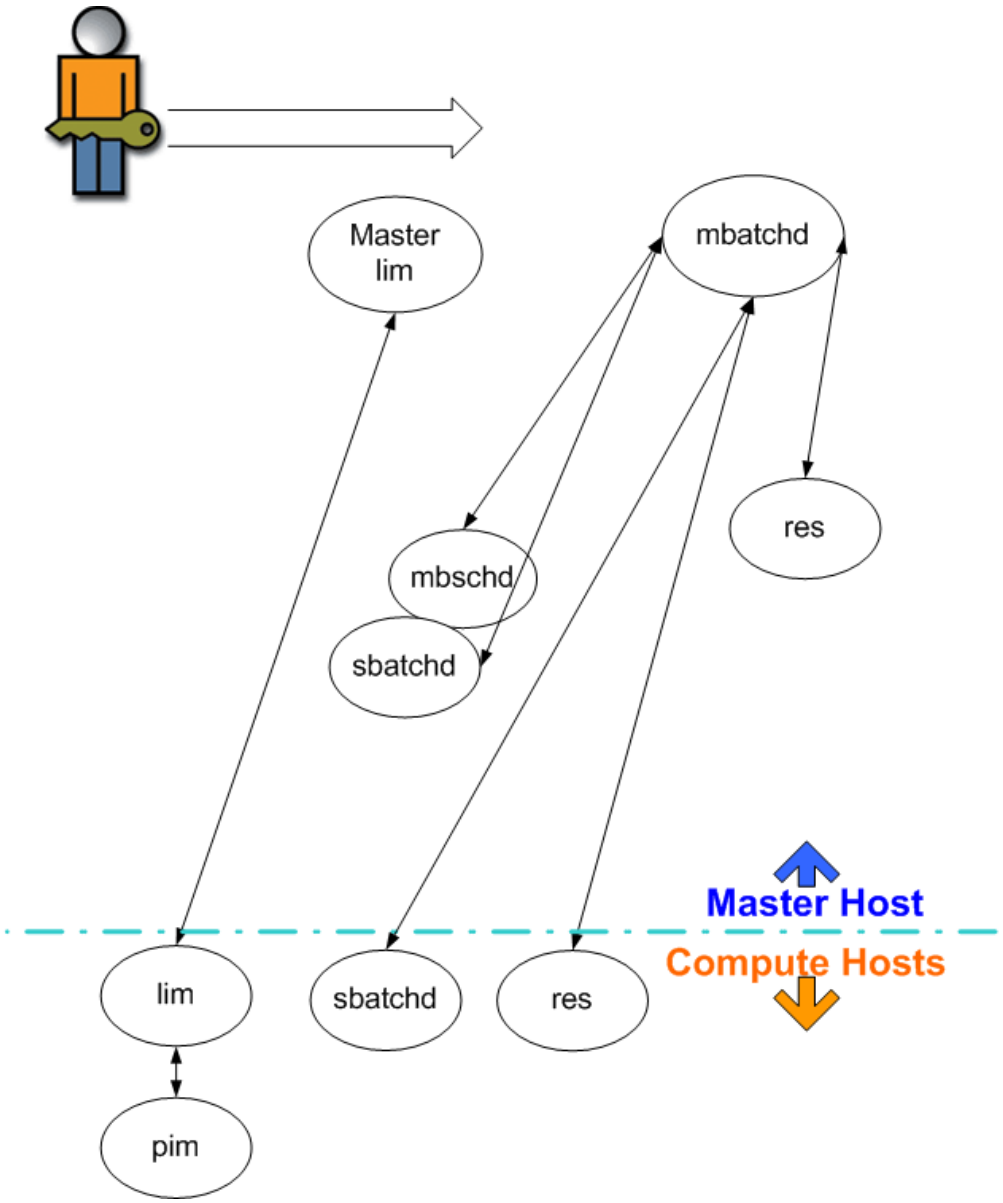
ELIM

External LIM (ELIM) is a site-definable executable that collects and tracks custom dynamic load indices. An ELIM can be a shell script or a compiled binary program, which returns the values of the dynamic

resources you define. The ELIM executable must be named `elim.`*anything* and located in `LSF_SERVERDIR`.

Platform LSF cluster communications paths

The communication paths between the daemons in the cluster are as shown below:



Fault tolerance

Platform LSF has a robust architecture designed with fault tolerance in mind. Every component in the system has a recovery operation—vital components are monitored by another component and can automatically recover from a failure.

LSF is designed to continue operating even if some of the hosts in the cluster are unavailable. One host in the cluster acts as the master, but if the master host becomes unavailable another master host candidate takes over. LSF is available as long as there is one available master host candidate in the cluster.

LSF can tolerate the failure of any host or group of hosts in the cluster. When a host becomes unavailable, all jobs running on that host are either requeued or lost, depending on whether the job was marked as rerunnable. No other pending or running jobs are affected.

How failover works

Fault tolerance in LSF depends on the event log file, `lsb.events`, which is kept on the primary file server. Every event in the system is logged in this file, including all job submissions and job and host status changes. If the master host becomes unavailable, a new master is chosen from the master candidate list, and `sbatchd` on the new master starts a new `mbatchd`. The new `mbatchd` reads the `lsb.events` file to recover the state of the system.

For sites not wanting to rely solely on a central file server for recovery information, LSF can be configured to maintain a duplicate event log by keeping a replica of `lsb.events`. The replica is stored on the file server, and used if the primary copy is unavailable. When using LSF's duplicate event log function, the primary event log is stored locally on the first master host, and re-synchronized with the replicated copy when the host recovers.

Host failover

The LSF master host is chosen dynamically. If the current master host becomes unavailable, another host takes over automatically. The failover master host is selected from the list defined in LSF_MASTER_LIST in `lsf.conf` (specified in `install.config` at installation). The first available host in the list acts as the master.

Running jobs are managed by `sbatchd` on each server host. When the new `mbatchd` starts, it polls the `sbatchd` on each host and finds the current status of its jobs. If `sbatchd` fails but the host is still running, jobs running on the host are not lost. When `sbatchd` is restarted it regains control of all jobs running on the host.

Job failover

Jobs can be submitted as *rerunnable*, so that they automatically run again from the beginning or as *checkpointable*, so that they start again from a checkpoint on another host if they are lost because of a host failure.

If all of the hosts in a cluster go down, all running jobs are lost. When a master candidate host comes back up and takes over as master, it reads the `lsb.events` file to get the state of all batch jobs. Jobs that were running when the systems went down are assumed to have exited unless they were marked as rerunnable, and email is sent to the submitting user. Pending jobs remain in their queues, and are scheduled as hosts become available.

Partitioned cluster

If the cluster is partitioned by a network failure, a master LIM takes over on each side of the partition as long as there is a master host candidate on each side of the partition. Interactive load-sharing remains available as long as each host still has access to the LSF executables.

Partitioned network

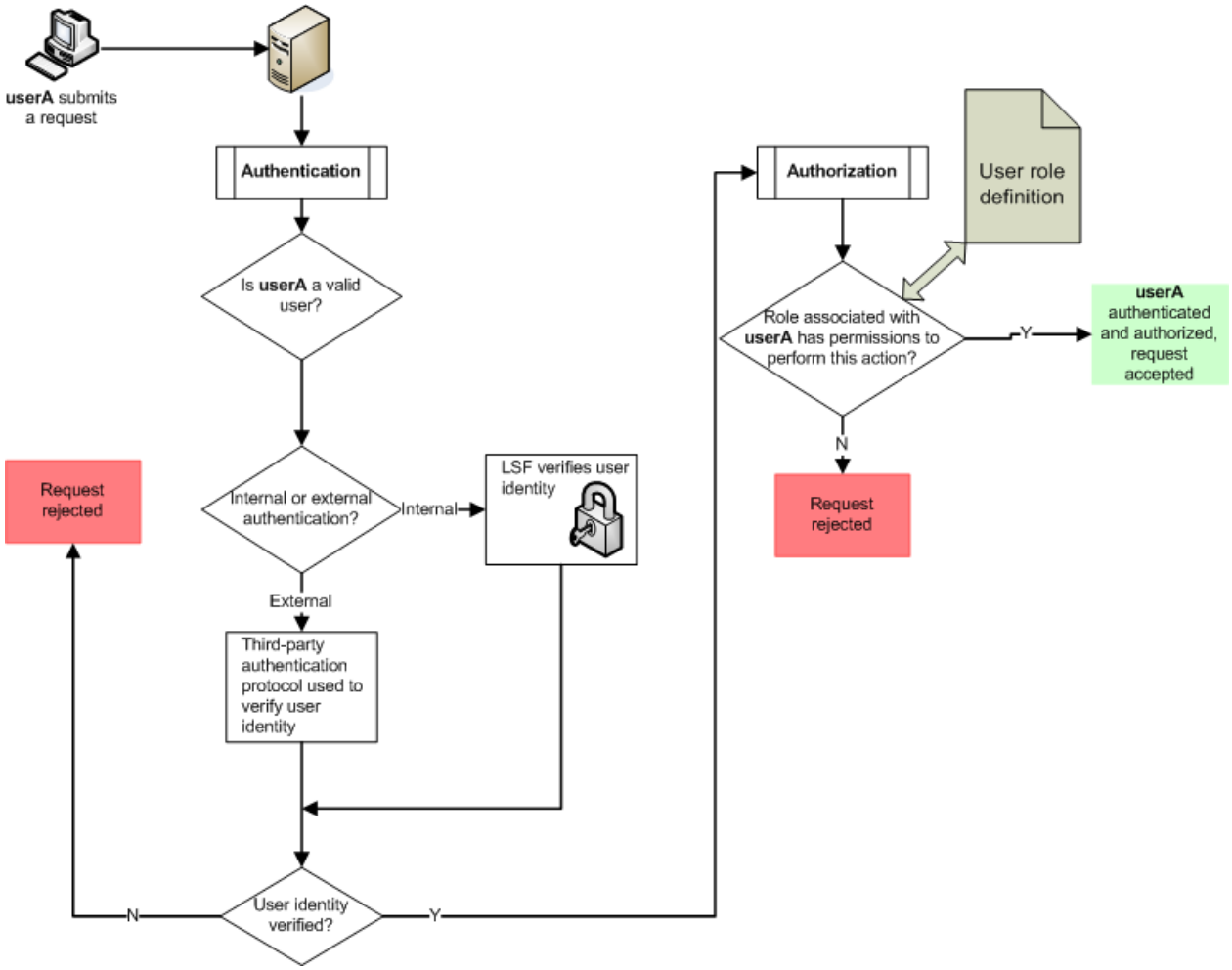
If the network is partitioned, only one of the partitions can access `lsb.events`, so batch services are only available on one side of the partition. A lock file is used to make sure that only one `mbatchd` is running in the cluster.

Job exception handling

You can configure hosts and queues so that LSF detects exceptional conditions while jobs are running, and takes appropriate action automatically. You can customize what exceptions are detected and the corresponding actions. For example, you can set LSF to restart a job automatically if it exits with a specific error code.

Security

Platform LSF security model



Out of the box, the LSF security model keeps track of user accounts internally. A user account defined in LSF includes a password to provide authentication and an assigned role to provide authorization, such as administrator.

Platform LSF user roles

LSF, without EGO enabled, supports the following roles:

- LSF user: Has permission to submit jobs to the LSF cluster and view the states of jobs and the cluster.
- Primary LSF administrator: Has permission to perform clusterwide operations, change configuration files, reconfigure the cluster, and control jobs submitted by all users.

  Configuration files such as `lsb.params` and `lsb.hosts` configure all aspects of LSF.
- LSF administrator: Has permission to perform operations that affect other LSF users.

- Cluster administrator: Can perform administrative operations on all jobs and queues in the cluster. May not have permission to change LSF configuration files.
- Queue administrator: Has administrative permissions limited to a specified queue.
- Hostgroup administrator: Has administrative permissions limited to a specified host group.
- Usergroup administrator: Has administrative permissions limited to a specified user group.

Platform LSF user roles with Platform EGO enabled

LSF, with EGO enabled, supports the following roles:

- Cluster Administrator: Can administer any objects and workload in the cluster
- Consumer Administrator: Can administer any objects and workload in consumers to which they have access
- Consumer User: Can run workload in consumers to which they have access

User accounts are created and managed in EGO. EGO authorizes users from its user database.

Platform LSF and UNIX user groups

LSF allows you to use any existing UNIX user groups directly by specifying a UNIX user group anywhere an LSF user group can be specified.

External authentication

LSF provides a security plug in for sites that prefer to use external or third-party security mechanisms, such as Kerberos, LDAP, ActiveDirectory, and so on.

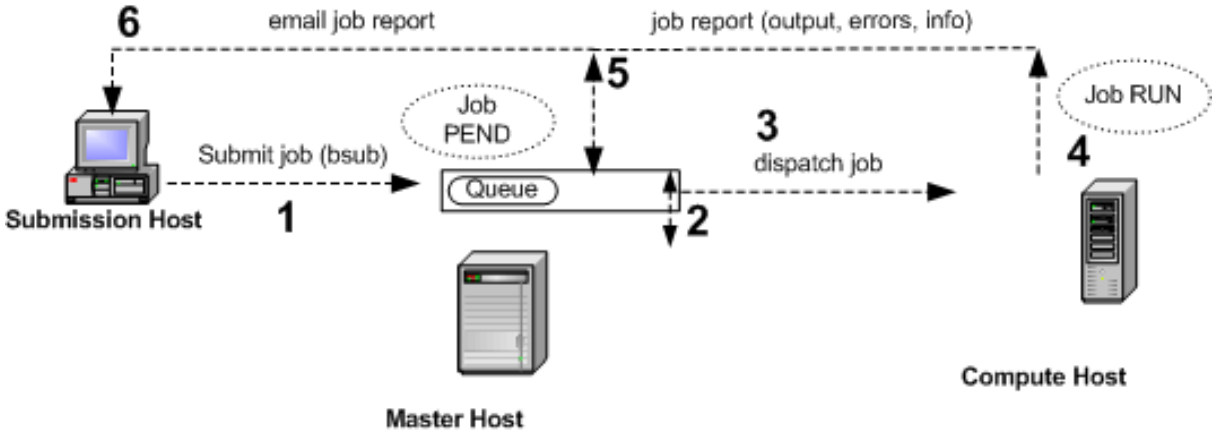
You can create a customized `eauth` executable to provide external authentication of users, hosts, and daemons. Credentials are passed from an external security system. The `eauth` executable can also be customized to obtain credentials from an operating system or from an authentication protocol such as Kerberos.

CHAPTER

3

Inside Workload Management

Job life cycle



1. Submit a job

You submit a job from an LSF client or server with the `bsub` command.

If you do not specify a queue when submitting the job, the job is submitted to the default queue.

Jobs are held in a queue waiting to be scheduled and have the PEND state. The job is held in a job file in the `LSF_SHAREDIR/`*cluster_name*`/logdir/info/` directory, or in one of its subdirectories if MAX_INFO_DIRS is defined in the configuration file `lsb.params`.

- Job ID: LSF assigns each job a unique job ID when you submit the job.
- Job name: You can also assign a name to the job with the `-J` option of `bsub`. Unlike the job ID, the job name is not necessarily unique.

2. Schedule the job

1. The master batch daemon (`mbatchd`) looks at jobs in the queue and sends the jobs for scheduling to the master batch scheduler (`mbschd`) at a preset time interval (defined by the parameter JOB_SCHEDULING_INTERVAL in the configuration file `lsb.params`).
2. `mbschd` evaluates jobs and makes scheduling decisions based on:
   - Job priority
   - Scheduling policies
   - Available resources
3. `mbschd` selects the best hosts where the job can run and sends its decisions back to `mbatchd`.

   Resource information is collected at preset time intervals by the master load information manager (LIM) from LIMs on server hosts. The master LIM communicates this information to `mbatchd`, which in turn communicates it to `mbschd` to support scheduling decisions.

3. Dispatch the job

As soon as `mbatchd` receives scheduling decisions, it immediately dispatches the jobs to hosts.

4. Run the job

The slave batch daemon (`sbatchd`):

1. Receives the request from `mbatchd`.
2. Creates a child `sbatchd` for the job.
3. Creates the execution environment.
4. Starts the job using a remote execution server (`res`).

   LSF copies the execution environment from the submission host to the execution host and includes the following:

   - Environment variables needed by the job
   - Working directory where the job begins running
   - Other system-dependent environment settings, for example:
     - On UNIX and Linux, resource limits and `umask`
     - On Windows, desktop and Windows root directory

       The job runs under the user account that submitted the job and has the status RUN.

5. Return output

When a job is completed, it is assigned the DONE status if the job was completed without any problems. The job is assigned the EXIT status if errors prevented the job from completing.

`sbatchd` communicates job information including errors and output to `mbatchd`.

6. Send email to client

`mbatchd` returns the job output, job error, and job information to the submission host through email. Use the `-o` and `-e` options of `bsub` to send job output and errors to a file.

- Job report: A job report is sent by email to the LSF client and includes:
  - Job information:
    - CPU use
    - Memory use
    - Name of the account that submitted the job
  - Job output
  - Errors

Job submission

On the command line, `bsub` is used to submit jobs and you can specify many options with `bsub` to modify the default behavior. Jobs must be submitted to a queue.

You can also use the Platform Application Center to submit jobs.

Queues

Queues represent a set of pending jobs, lined up in a defined order and waiting for their opportunity to use resources. Queues implement different job scheduling and control policies.

Jobs enter the queue via the `bsub` command. Queues have the following attributes associated with them:

- Priority
- Name
- Queue limits (restrictions on hosts, number of jobs, users, groups, or processors)
- Standard UNIX limits: memory, swap, process, CPU
- Scheduling policies
- Administrators
- Run conditions
- Load-sharing threshold conditions
- UNIX `nice(1)` value, (sets the UNIX scheduler priority)

Queue priority

Defines the order in which queues are searched to determine which job will be processed. Queues are assigned a priority by the LSF administrator, where a higher number has a higher priority. Queues are serviced by LSF in order of priority from the highest to the lowest. If multiple queues have the same priority, LSF schedules all the jobs from these queues in first-come, first-served order.

Automatic queue selection

When you submit a job, LSF considers the requirements of the job and automatically chooses a suitable queue from a list of candidate default queues.

LSF selects a suitable queue according to:

- User access restriction: Queues that do not allow this user to submit jobs are not considered.
- Host restriction: If the job explicitly specifies a list of hosts on which the job can be run, then the selected queue must be configured to send jobs to hosts in the list.
- Queue status: Closed queues are not considered.
- Exclusive execution restriction: If the job requires exclusive execution, then queues that are not configured to accept exclusive jobs are not considered.
- Job's requested resources: These must be within the resource allocation limits of the selected queue.

If multiple queues satisfy the above requirements, then the first queue listed in the candidate queues that satisfies the requirements is selected.

Job scheduling and dispatch

Submitted jobs wait in queues until they are scheduled and dispatched to a host for execution. When a job is submitted to LSF, many factors control when and where the job starts to run:

- Active time window of the queue or hosts
- Resource requirements of the job
- Availability of eligible hosts
- Various job slot limits
- Job dependency conditions
- Fairshare constraints (configured user share policies)
- Load conditions

Scheduling policies

To solve diverse problems, LSF allows multiple scheduling policies in the same cluster. LSF has several queue scheduling policies such as exclusive, preemptive, fairshare, and hierarchical fairshare.

- First-come, first-served (FCFS) scheduling: By default, jobs in a queue are dispatched in FCFS order. This means that jobs are dispatched according to their order in the queue.
- Service level agreement (SLA) scheduling: An SLA in LSF is a “just-in-time” scheduling policy that schedules the services agreed to between LSF administrators and LSF users. The SLA scheduling policy defines how many jobs should be run from each SLA to meet the configured goals.
- Fairshare scheduling: If you specify a fairshare scheduling policy for the queue or if host partitions have been configured, LSF dispatches jobs between users based on assigned user shares, resource usage, or other factors.
- Preemption: You can specify desired behavior so that when two or more jobs compete for the same resources, one job preempts the other. Preemption can apply to not only job slots, but also to advance reservation (reserving hosts for particular jobs) and licenses (using Platform License Scheduler).
- Backfill: Allows small jobs to run on job slots reserved for other jobs, provided the backfilling job completes before the reservation time expires and resource usage is due.

Scheduling and dispatch

Jobs are scheduled at regular intervals (5 seconds by default). Once jobs are scheduled, they can be immediately dispatched to hosts.

To prevent overloading any host, by default LSF waits a short time between dispatching jobs to the same host.

Dispatch order

Jobs are not necessarily dispatched in order of submission.

Each queue has a priority number set by an LSF Administrator when the queue is defined. LSF tries to start jobs from the highest priority queue first.

By default, LSF considers jobs for dispatch in the following order:

- For each queue, from highest to lowest priority. If multiple queues have the same priority, LSF schedules all the jobs from these queues in first-come, first-served order.
- For each job in the queue, according to FCFS order.

- If any host is eligible to run this job, start the job on the best eligible host, and mark that host ineligible to start any other job until JOB_ACCEPT_INTERVAL has passed.

Host selection

Each time LSF attempts to dispatch a job, it checks to see which hosts are eligible to run the job. A number of conditions determine whether a host is eligible:

- Host dispatch windows
- Resource requirements of the job
- Resource requirements of the queue
- Host list of the queue
- Host load levels
- Job slot limits of the host
- User quota and user limits

A host is only eligible to run a job if all the conditions are met. If a job is queued and there is an eligible host for that job, the job is placed on that host. If more than one host is eligible, the job is started on the best host based on both the job and the queue resource requirements.

Host load levels

A host is available if the values of the load indices (such as `r1m`, `pg`, `mem`) of the host are within the configured scheduling thresholds. There are two sets of scheduling thresholds: host and queue. If any load index on the host exceeds the corresponding host threshold or queue threshold, the host is not eligible to run any job.

Eligible hosts

When LSF tries to place a job, it obtains current load information for all hosts.

The load levels on each host are compared to the scheduling thresholds configured for that host in the `Host` section of `lsb.hosts`, as well as the per-queue scheduling thresholds configured in `lsb.queues`.

If any load index exceeds either its per-queue or its per-host scheduling threshold, no new job is started on that host.

Job execution environment

When LSF runs your jobs, it tries to make it as transparent to the user as possible. LSF copies the environment from the submission host to the execution host. The execution environment includes the following:

- Environment variables needed by the job
- Working directory where the job begins running
- Other system-dependent environment settings; for example, resource usage limits

Shared user directories

To provide transparent remote execution, LSF commands determine the user's current working directory and use that directory on the remote host.

Executables and the PATH environment variable

Search paths for executables (the PATH environment variable) are passed to the remote execution host unchanged.

Note:

In mixed clusters, LSF works best when the user binary directories have the same path names on different host types. This makes the PATH variable valid on all hosts.

For easy administration, LSF configuration files are stored in a shared directory.

CHAPTER

4

Platform LSF with Platform EGO Enabled

EGO component overview

EGO can be enabled with LSF to provide a system infrastructure to control and manage cluster resources.

Just as an operating system running on a single machine aggregates and virtualizes physical resources and allocates them to applications, EGO performs similar functions, but across a distributed environment.

EGO manages both logical and physical resources and supports all forms of applications. EGO manages the supply of resources, making them available to applications.

Hosts can be divided into two groups: management hosts and compute hosts. Management hosts provide specialized services to the cluster, while compute hosts run user workload.

Management hosts

Management hosts provide both cluster and workload management services within the cluster, and are not expected to run workload for users. The master host, all master candidate hosts, and session manager hosts must be management hosts. Other management hosts include the host running the data loaders and data purger for the reporting feature.

Management hosts all run on the same operating system: all Windows or all UNIX.

Master host	The master host is the first host installed in the cluster. The resource manager (`vemkd`) for the cluster resides on this host. The master host controls the rest of the hosts in the cluster and is the interface to the clients of the cluster.
Master candidates	There is only one master host at a time. If the master host should fail, another host automatically takes over the master host role. Hosts that can act as the master are called master candidates.
Session manager host	One or more management hosts run session managers. There is one session manager per available slot on a management host. There is one session manager per application.

Compute hosts

Compute hosts are those hosts in the cluster that provide computing resources to consumers. A cluster may contain any number of compute hosts, but must have at least one compute host.

CPU slots	A CPU slot is the unit used to measure compute resources. A single CPU slot can run one service instance on a compute host, or one session manager on a management host.

Daemons

- VEMKD: The VEM kernel daemon that runs on the master host. It starts other daemons and responds to allocation requests
- EGOSC: The EGO service controller requests appropriate resources from the VEMKD and controls service instances.
- PEM: Process execution manager works for the VEMKD, starting, controlling, and monitoring activities, as well as collecting and sending run time resource usage.

Resources

Resources are physical and logical entities that are used by applications in order to run. While resource is a generic term, and can include low-level things such as shared memory segments or semaphores, in LSF, EGO manages CPU slots.

A resource of a particular type has attributes. For example, a compute host has the attributes of memory, CPU utilization, operating system type, and so on.

Resource groups

Resources may be grouped together into logical groups to simplify identification, resource allocation, or for administration and monitoring purposes. These resource groups are used to provide a consumer with a like group of hosts to run workload—any host in a resource group should be able to run the same workload.

Figure 1:
Out-of-Box Resource Groups

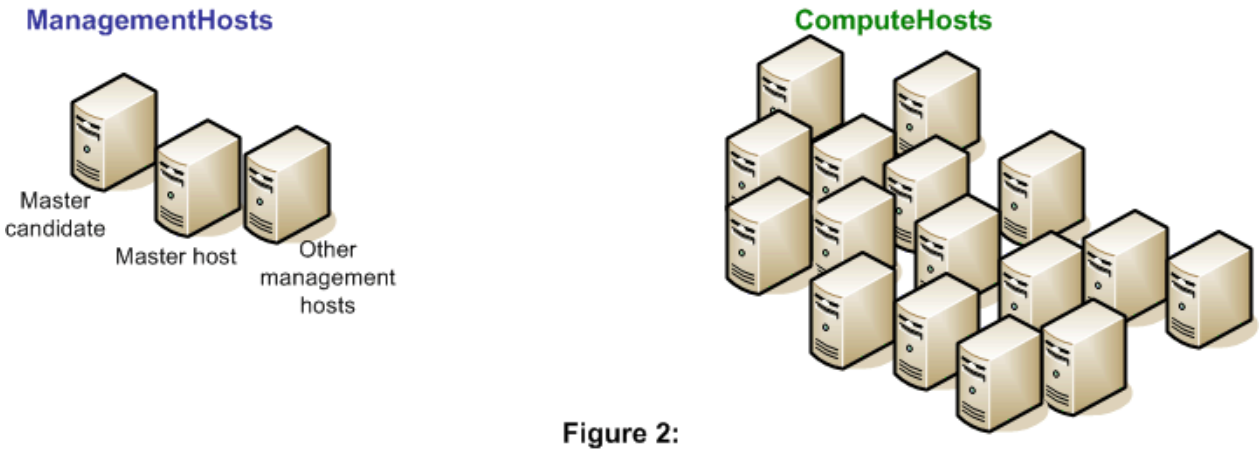
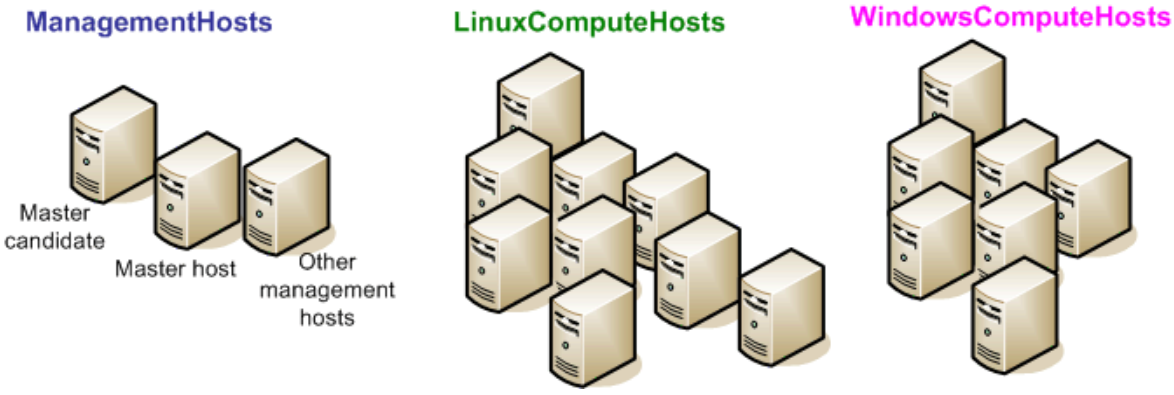


Figure 2:
Grouping Like Resources



As shown in Figure 1, there are two resource groups out of the box:

- ManagementHosts
- ComputeHosts

If all of your hosts are identical, these resource groups may suffice. If your application requires a specific type of hosts (for example, with a minimum processor speed), and not all hosts meet this criteria, you likely need to create resource groups to group like hosts together.

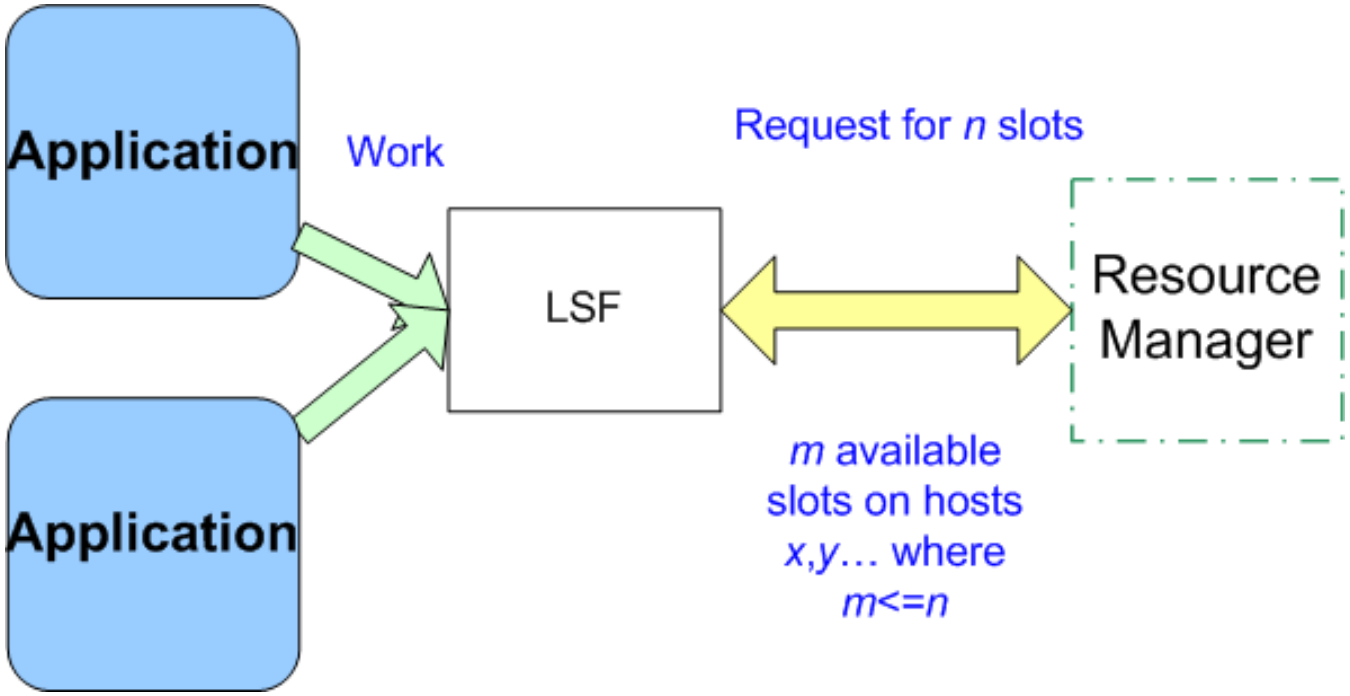
For example, a simple way to group resources may be to group your hosts by operating system type.

EGO provides a common grouping mechanism for resources. Resources may come and go from the system, so EGO supports dynamic membership in a resource group. Hosts can be placed explicitly into individual resource groups, or the resource groups can be defined to have a dynamic membership based on specific criteria. This criteria includes operating system type, CPU speed, total memory, or swap configuration, or custom attributes.

Sharing of Platform LSF resources

LSF resources are shared as defined in the resource distribution plan.

LSF requests resources from EGO's resource manager. Based on the values specified in the resource distribution plan, the resource manager returns the number of available slots (*m*) and the names of the hosts on which the slots reside.



Index