Platform LSF
Version 9 Release 1.3

Using Platform LSF Advanced Edition

SC27-5321-03

Platform LSF
Version 9 Release 1.3

Using Platform LSF Advanced Edition

IBM

SC27-5321-03

Note

Before using this information and the product it supports, read the information in "Notices" on page 57.

First edition

This edition applies to version 9, release 1 of IBM Platform LSF (product number 5725G82) and to all subsequent releases and modifications until otherwise indicated in new editions.

Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

If you find an error in any Platform Computing documentation, or you have a suggestion for improving it, please let us know.

In the IBM Knowledge Center, add your comments and feedback to any topic.

You can also send your suggestions, comments and questions to the following email address:

pccdoc@ca.ibm.com

Be sure include the publication title and order number, and, if applicable, the specific location of the information about which you have comments (for example, a page number or a browser URL). When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

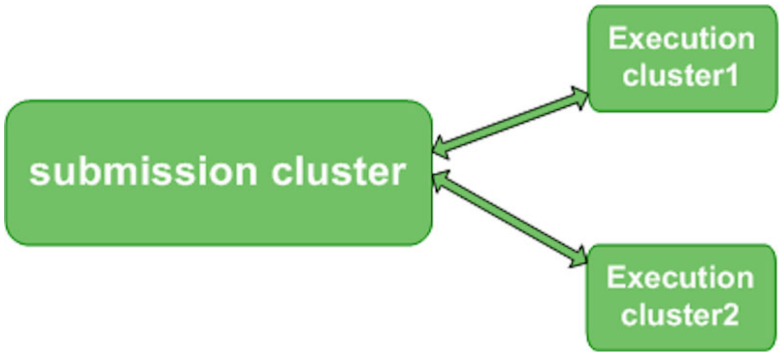
Chapter 1. Overview

Overview

IBM Platform LSF Advanced Edition (LSF AE) extends the functionality of IBM Platform LSF Standard Edition (LSF) and IBM Platform License Scheduler Standard Edition (License Scheduler), applying multi-cluster technology and improved job scheduling for enhanced scalability. Maintaining the front-end behavior of a single LSF cluster, LSF AE splits into multiple execution clusters behind the scenes, allowing efficient scheduling and completion of large numbers of jobs.

Use LSF AE to combine multiple LSF clusters and streamline administration, or to increase the capacity of already large clusters.

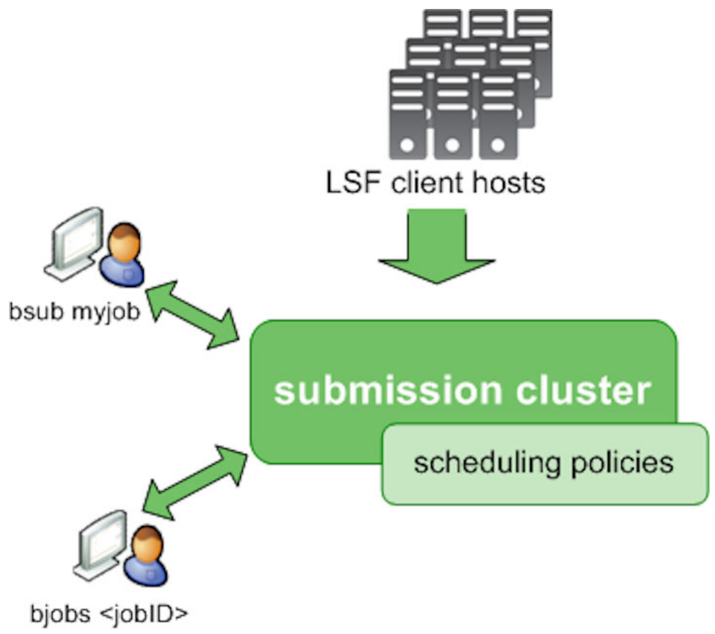
LSF AE also works with License Scheduler. Use LSF AE with License Scheduler to allocate, share, and preempt license tokens among LSF AE clusters, and to include License Scheduler resources when working with job forwarding policies.



Submission cluster

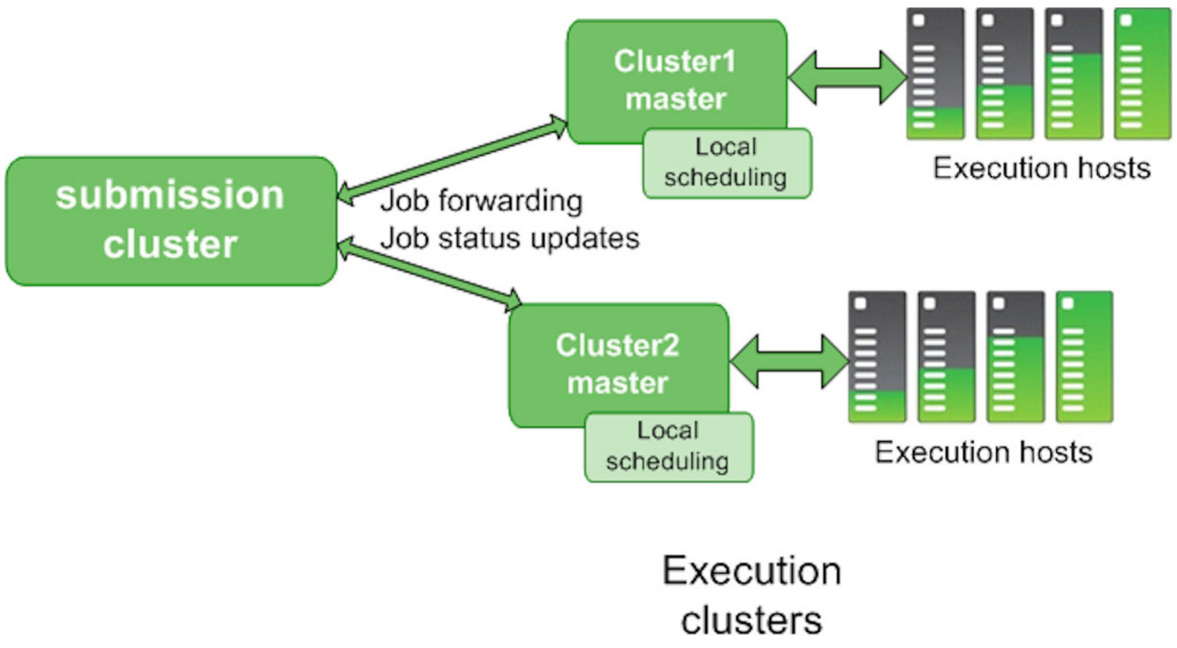
The submission cluster is the job submission gateway, the scheduling hub, and the user interface of LSF AE. Through the submission cluster jobs are forwarded for rapid scheduling in each execution cluster, LSF AE policies are implemented, and job queries are submitted.

The submission cluster silently communicates with multiple execution clusters, sending jobs and receiving updates behind the scenes.



Execution clusters

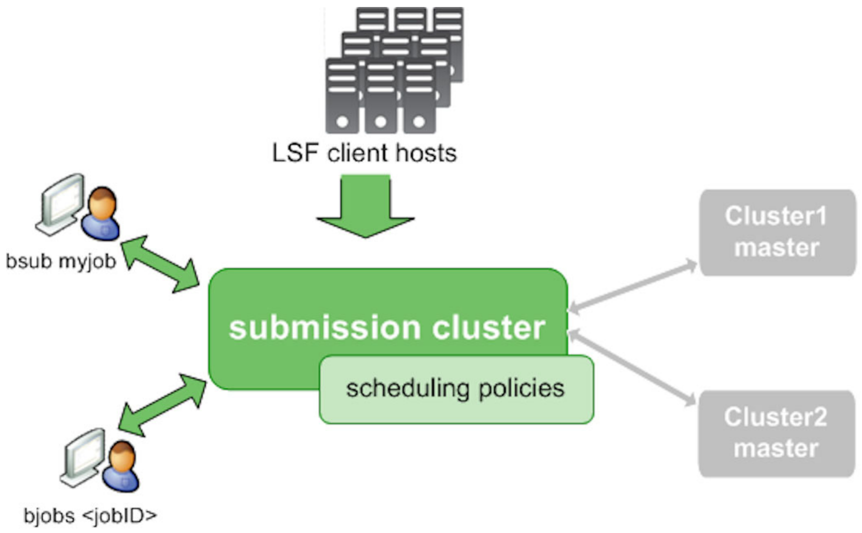
Execution clusters are transparent to users, configured by administrators as natural extensions of the submission cluster. Execution cluster configuration mirrors that of the submission cluster, while using the capacity of each execution cluster master host for local job scheduling.



Job forwarding

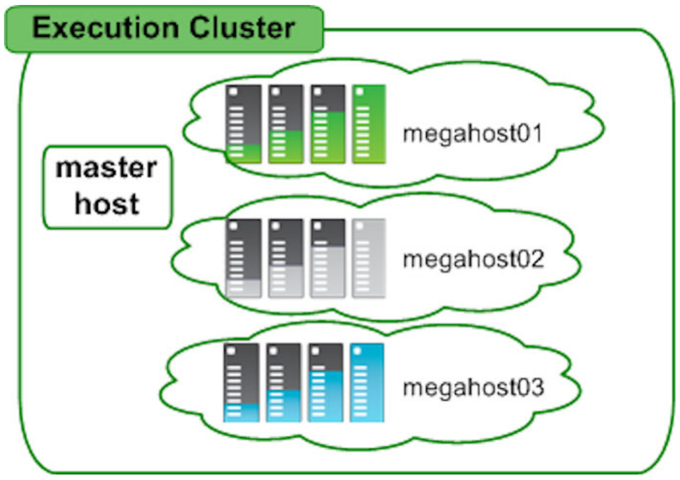
The submission cluster acts as a gateway for all job submission. Multiple execution clusters are configured internally by administrators for efficient job handling, but the appearence of a single cluster is maintained for regular users.

Job status, pending reasons, and resource usage are returned to users through the submission cluster.



Megahosts

LSF AE job forwarding uses the concept of megahosts to forward jobs effectively. A megahost is a grouping of hosts with the same key attributes, all within one execution cluster. Megahosts provide detailed information about resources in the execution clusters without requiring the LSF AE forwarding scheduler to spend time considering each individual host.



Grouping of execution cluster hosts into megahosts is done internally by LSF AE. Megahosts are created automatically by LSF AE and require no special configuration.

When planning your LSF AE installation, keep in mind that hosts within each execution cluster are grouped into megahosts during job forwarding. To optimize the formation of megahosts within LSF AE, group similar hosts in the same execution cluster.

Job scheduling under LSF AE

Scheduling of LSF AE jobs is a process with three scheduling phases:

- Local scheduling (optional)
- The submission cluster selects a suitable execution cluster, and forwards the job.
- The execution cluster selects a suitable host and dispatches the job.

  Local scheduling automatically favors local hosts; a submission cluster scheduler always attempts to find a suitable local host before considering execution clusters.

Job submission

1. The submission cluster receives the job submission request from a user.
2. If the job is accepted, it pends with a job ID assigned by the submission cluster.

Phase I, submission cluster local scheduling (optional)

Local scheduling in the submission cluster is available for use during migration from the single LSF cluster model to the multiple execution cluster LSF AE configuration.

Once migration is complete and only the master host and master host candidates remain in the submission cluster, local scheduling can be bypassed by setting `HOSTS=none` (`lsb.queues`) for all submission cluster local queues.

1. During the next scheduling cycle, the LSF AE scheduler attempts to place the job on a local host. If a suitable host is found, the job is dispatched locally.
2. If the job cannot be placed locally, the submission cluster attempts to forward the job to an execution cluster.

Phase II, submission cluster job forwarding

1. The submission cluster evaluates each execution cluster based on configured LSF AE policies.

   By default, load balancing is used to select an execution cluster, and the cluster with the most available slots is selected. If all job slots in all clusters are full, the cluster with the lowest ratio of pending jobs to total slots is selected.

   Fairshare, job limits, and cluster preferences can also be configured in the submission cluster for optimal job forwarding.
2. If available information indicates that the selected execution cluster is suitable, the submission cluster forwards the job to that cluster.

   For jobs with resource requirements, only `swp`, `mem`, `type`, `ncpus`, `model`, boolean resources, CPU factors, and cluster-wide shared resources are considered by the submission cluster during job forwarding.
3. If the submission cluster cannot forward the job to any execution cluster, the job remains pending and is evaluated again during the next scheduling cycle.

Phase III, execution cluster scheduling (forwarded jobs)

1. The execution cluster queue receives the forwarded job.
2. If the job is accepted, it pends in the execution cluster queue using the same job ID assigned by the submission cluster.
3. During the next scheduling cycle, the execution cluster queue evaluates all resource requirements and attempts to place the job on a host in the execution cluster.

If a suitable host is found, the job is dispatched. If a suitable host is not found, the job remains pending in the execution cluster, and is evaluated again in the next scheduling cycle.

4. If the job is dispatched to the execution host but cannot start after the time interval **MAX_RSCHED_TIME** (set in the queue configuration in `lsb.queues`), it returns to the submission cluster to be rescheduled.

Differences between LSF and LSF AE

LSF AE supports many LSF features and commands. Some less common features and commands are not supported:

- backfill time window display (**bslots**)
- HPC integrations (**brlainfo**)
- Windows is not a supported platform (**lspasswd**, **tspeek**, **tssub**, **wgpasswd**, **wguser**)

Some features and commands are not supported by job forwarding in LSF AE, but do apply within each local execution cluster:

- SLAs (**bsla**) applies to local job scheduling only
- host partition fairshare (**bhpart**) applies to local resources and scheduling only
- checkpointing and restarting jobs (**bchkpnt**, **brestart**) applies to local and forwarded jobs
- job slot reservation information (**bjobs**, **bqueues**, **bhosts**, **busers**) for local jobs only
- host affinity information (**bhosts -aff**) for local hosts only

Chapter 2. LSF AE Setup

Install and configure LSF AE

LSF AE files are automatically installed by LSF's regular Setup program (**lsfinstall**). Install LSF and make sure each cluster works properly as a standalone cluster before you proceed to configure LSF AE. When grouping hosts into execution clusters, remember that LSF AE groups together hosts with similar resources within each cluster to form megahosts. For optimal job forwarding, group hosts with similar resources within one execution cluster.

Ensure all clusters are installed under the same shared file system, since execution clusters read submission cluster job files directly. If using a separate directory for job information (**LSB_JOBINFO_DIR** in `lsf.conf`), ensure this directory is accessible from all clusters.

LSF AE queues

By default, clusters do not share resources, even if LSF AE has been installed. To enable job forwarding, configure queues with the same name in both the submission and execution clusters.

Send-jobs queue

A send-jobs queue can forward jobs to a specified remote queue. By default, LSF attempts to run jobs in the local cluster first. LSF only attempts to place a job remotely if it cannot place the job locally. The submission cluster uses send-job queues.

Receive-jobs queue

A receive-jobs queue accepts jobs from queues in a specified remote cluster. The execution clusters use receive-job queues. All receive-jobs queues in execution clusters must have the same name as the send-jobs queue in the submission cluster.

Set common ports

Participating clusters must use the same port numbers for the daemons LIM, RES, MBD, and SBD.

By default, all clusters have identical port settings, as shown:

```
LSF_LIM_PORT=7869
LSF_RES_PORT=6878
LSB_MBD_PORT=6881
LSB_SBD_PORT=6882
```

Troubleshooting

To check your port numbers, check the `LSF_TOP/conf/lsf.conf` file in each cluster. (**LSF_TOP** is the LSF installation directory. On UNIX, this is defined in the `install.config` file). Make sure you have identical settings in each cluster for these port settings parameters.

Define participating clusters and valid master hosts

To enable LSF AE, define all participating clusters in the Cluster section of the `LSF_TOP/conf/lsf.shared` file.

1. For ClusterName, specify the name of each participating cluster. On UNIX, each cluster name is defined by **`LSF_CLUSTER_NAME`** in the `install.config` file.
2. For Servers, specify the master host and optionally candidate master hosts for the cluster. A cluster will not participate in LSF AE resource sharing unless its current master host is listed here.

Example

In the `lsf.shared` file:

```
Begin Cluster
ClusterName  Servers
Sub_Cluster (hostA hostB)
Cluster1    (hostD)
End Cluster
```

In this example, `hostA` should be the master host of `Sub_Cluster` with `hostB` as the backup, and hostD should be the master host of `Cluster1`. If the master host fails in `Sub_Cluster`, LSF AE will still work because the backup master is also listed here. However, if the master host fails in `Cluster1`, LSF AE will not recognize any other host as the master, so `Cluster1` will no longer participate in LSF AE resource sharing.

Common resource definitions

For resource sharing to work between clusters, the clusters should have common definitions of host types, host models, and resources. The submission cluster uses the local configuration files to make job forwarding decisions, so it is important these definitions remain consistent across all clusters.

Much of this information is in `lsf.shared`, so the best way to configure LSF AE is to make sure `lsf.shared` is identical for each cluster. If you do not have a shared file system, replicate `lsf.shared` across all clusters.

Set common resource definitions:

1. Ensure that the `lsf.shared` file is identical in all clusters. The order of resource definitions as well as the definitions themselves must not change.
2. Ensure that the `lsb.resources` file ReservationUsage section is identical for all clusters.
3. Ensure that the `lsf.cluster.`*`cluster_name`* file ResourceMap section is consistent for all clusters.

   Since the resources defined in the ResourceMap section vary from cluster to cluster, this section should not be identical across clusters. However, resources with the same name should have the same behavior (such as shared or non-shared) in all clusters.

   Note:

   The local ResourceMap definition is used within each cluster. Inconsistent definitions can result in jobs being forwarded to the wrong execution cluster.

Submission cluster configuration

1. In the `lsf.conf` configuration file:
   a. Set `LSF_XL=`*`sub_clustername`*, where *sub_clustername* is the actual name of your submission cluster.
   b. Set `LSB_FS_COMM_ENH=Y` to enable the fairshare performance enhancement.

2. Add the LSF AE plugin to the `lsb.modules` file, and comment out the Platform MultiCluster plugin:

   ```
   Begin PluginModule
   #(comment this out) schmod_mc ()  ()
   schmod_xl ()  ()
   End PluginModule
   ```

   Note:

   Make sure the schmod_xl plugin is listed before the schmod_preemption plugin in the PluginModule configuration.
3. Configure the submission cluster `lsb.queues` file.
   a. For each LSF AE send-jobs queue, define SNDJOBS_TO in the `lsb.queues` queue definition. Specify a space-separated list of execution cluster names in the format *cluster1_name cluster2_name*.

      If SNDJOBS_TO is not configured in the queue, it cannot forward jobs. The jobs remains pending in the submission cluster and are evaluated again during the next scheduling cycle.
   b. Make sure the `lsb.queues` HOSTS parameter for each queue specifies only local hosts (or the special keyword none).

   ```
   Begin Queue
   QUEUE_NAME=queue1
   SNDJOBS_TO=cluster2 cluster3
   HOSTS=none
   PRIORITY=30
   NICE=20
   End Queue
   ```

   The queue in this example is a send-jobs queue in the submission cluster, sending jobs to multiple execution clusters. Jobs submitted to `queue1` in the submission cluster are forwarded to `queue1` in `cluster2`, or to `queue1` in `cluster3`.

Execution cluster configuration

The execution clusters used by LSF AE require specific configuration settings.

1. In the `lsf.conf` configuration file:
   a. Set `LSF_XL=`*`sub_clustername`*, where *sub_clustername* is the actual name of your submission cluster.
   b. Set `LSB_FS_COMM_ENH=Y` to enable the fairshare performance enhancement.
2. In the `lsb.modules` configuration file, comment out the Platform MultiCluster plugin:

   ```
   # schmod_mc  ()  ()
   ```
3. Configure receive-jobs queues, defining RCVJOBS_FROM in each `lsb.queues` queue definition, matching queue names to those in the execution cluster. Specify the submission cluster name jobs will be forwarded from.

   ```
   Begin Queue
   QUEUE_NAME=queue1
   RCVJOBS_FROM=sub_cluster
   PRIORITY=30
   NICE=20
   End Queue
   ```

   This queue is a receive-jobs queue in the execution cluster, receiving jobs from the configured send-jobs queue of the same name in the submission cluster.

Test communication between clusters

1. Restart each cluster using the **lsadmin** and **badmin** commands:

```
% lsadmin limrestart all
% badmin mbdrestart
```

2. To verify that the LSF/XL feature is enabled, run commands on the submission cluster:

```
% lsclusters
CLUSTER_NAME    STATUS    MASTER_HOST    ADMIN    HOSTS    SERVERS
sub_cluster      ok        hostA          lsfadmin 1        1
cluster2        ok        hostB          lsfadmin 1        1
cluster3        ok        hostC          lsfadmin 2        2


% lshosts
HOST_NAME type    model     cpuf  ncpus maxmem maxswp server RESOURCES
hostA     X86_64  PC6000    116.1 2     1000M  1983M  Yes      (mg)
hostB     X86_64  Opteron2  47.0  1     1000M  1961M  Yes      (mg)
hostC     X86_64  Intel_EM  60.0  2     7978M  1992M  Yes      (mg)
hostD     X86_64  Intel_EM  60.0  2     3828M  2055M  Yes      (mg)


% lsload
HOST_NAME status r15s r1m r15m ut  pg  ls it tmp   swp   mem
hostA     ok     0.2   0.0 0.0 1%  0.0 6  4  22G   1948M 725M
hostC     ok     0.4   0.1 0.0 7%  0.0 3  0  3270M 1990M 5448M
hostB     ok     0.6   0.0 0.0 0%  0.0 2  0  18M   1554M 518M
hostD     ok     1.0   1.0 1.1 13% 0.0 8  20 21G   353M  491M


% bclusters
 [Job Forwarding Information ]
 LOCAL_QUEUE     JOB_FLOW   REMOTE     CLUSTER    STATUS
 queue1          send       queue1     cluster2   ok
 queue1          send       queue1     cluster3   ok

 [Resource Lease Information ]
 No resources have been exported or borrowed


% bhosts
 HOST_NAME STATUS JL/U   MAX   NJOBS  RUN  SSUSP  USUSP  RSV
 hostA     ok     -     2      0      0    0      0      0
 hostB     ok     -     1      0      0    0      0      0
 hostC     ok     -     2      0      0    0      0      0
 hostD     ok     -     2      0      0    0      0      0


% bsub -q queue1 sleep 1234
Job <1> is submitted to queue <queue1>.


% bjobs -fwd
JOBID USER    STAT QUEUE  EXEC_HOST JOB_NAME   CLUSTER  FORWARD_TIME
1     lsfuser RUN  queue1 hostC     sleep 1234 cluster3 Nov 29 14:08
```

Chapter 3. Configuring Features

Load balancing and resource aware job forwarding

By default, the submission cluster applies load balancing when selecting an execution cluster. Possible execution clusters are ordered by the fraction of available slots (out of the total in the cluster), from greatest to least. If all available job slots in all clusters are full, clusters are ordered by the fraction of pending job slots (spaces for jobs to pend) from greatest to least.

Megahosts

LSF AE job forwarding uses internally configured megahosts with the same key attributes (type, model, maxmem, maxswp, ncpus, boolean resources, and cpu factors), all within one execution cluster. Shared resources in an execution cluster are attached to all megahosts within that cluster for job-forwarding purposes.

The megahost names along with the key attributes selected for each megahost are listed in the file `megahostid` in the directory `$LSB_SHAREDIR/`*`cluster_name`*`/logdir`. Megahost names appear in **`JOB_FORWARD`** events, and are used internally by LSF.

Resource view from the submission cluster

To facillitate rapid dispatch in each execution cluster, a few jobs should always be pending. Without pending jobs ready to dispatch, the execution clusters are left waiting for jobs to be forwarded from the submission cluster. In addition, not all resources are considered by the submission cluster when jobs are forwarded, so some forwarded jobs may not be able to run. As a result forwarding more jobs than can actually run at one time uses execution cluster resources effectively.

The number of jobs forwarded is controlled by the XL_RES_RATIO in the submission cluster (`lsb.params`). XL_RES_RATIO sets the virtual capacity of each execution cluster as seen from the submission cluster, given by the number of slots waiting to hold forwarded jobs:

forward slots = (available slots in cluster)*(**`XL_RES_RATIO`** for cluster)

Setting **`XL_RES_RATIO`** greater than one results in more jobs being forwarded than can run, up to the enlarged virtual capacity of the execution cluster. These excess jobs pend in the execution cluster until resources become available, ensuring resources in the execution cluster are not left idle when there are jobs to run. Forwarding too many jobs results in an unbalanced execution cluster workload, thus **`XL_RES_RATIO`** also sets a forwarding threshold.

The exact value of forward slots set by **`XL_RES_RATIO`** acts as a guideline instead of a strict threshold. Although the submission cluster follows the forward slots value when forwarding jobs, the actual number of forwarded jobs in each execution cluster reflects local scheduling policies, resource availability, and re-forwarded jobs. The submission cluster compensates by forwarding more or less jobs in the next forwarding cycle, as required.

Shared resource view

Forwarding of jobs with shared resource requirements follows the same principle of rapid dispatch, with the goal of forwarding more jobs than can run. For static user-defined resources, the number of configured resources is adjusted to reflect the virtual capacity set by **`XL_RES_RATIO`**:

total (static) resources = (total configured)*(**`XL_RES_RATIO`** for cluster)

In the case of dynamic shared user-defined resources, the submission cluster depends on execution cluster ELIMs to collect resource availability data. The ELIM information is combined with resource use to provide the total number of shared resources in the execution cluster.

The resource total seen by the submission cluster is adjusted to reflect the virtual capacity set by **`XL_RES_RATIO`**:

total (dynamic) resources = (available in cluster + used by running jobs in cluster)*(**`XL_RES_RATIO`** for cluster)

When the user-defined resources include licenses, duration may be used in the rusage string to release resource reservations, and applications may or may not check out licenses for the duration they are in use. The ELIM-reported license availability can include some license tokens actually in use but not checked out. Since the goal is to forward more jobs than can run, inflated resource availability reports do not cause a problem.

Resource requirement evaluation

The LSF AE submission cluster forwards jobs using resource-aware scheduling. Scheduling efficiency dictates that only the most important and frequently used resources be considered, namely: swp, mem, type, ncpus, model, boolean resources, cpu factors, and user-defined resources. Job resource requirements for these commonly used resources are matched to megahost resources within the execution clusters before jobs are forwarded.

The execution cluster evaluates all resource requirements before scheduling jobs.

Submission cluster

Resource requirements other than those listed are not considered when jobs are forwarded by the submission cluster, but are considered when jobs are scheduled in the execution clusters.

Since dynamic mem and swp usage is not known, maxmem and maxswp values are used instead.

`select string`

When these resources are specified in the select string, the submission cluster confirms the resource is present (based on megahost resources) before forwarding the job:

- maxmem
- maxswp
- mem>, mem>=
- swp>, swp>=

- boolean resources
- ncpus
- cpufactor
- type
- model

`rusage string`

When these resources are specified in the rusage string, the submission cluster confirms the resource is present (based on megahost resources) before forwarding the job:

- mem>, mem>=
- swp>, swp>=
- user-defined resources

`shared user-defined resources`

Shared user-defined resource requirements in the job rusage string are evaluated further before job forwarding. Jobs are forwarded based not only on the presence of the required resource, but also on the current availability. Shared resources are reserved by the submission cluster during job forwarding so resource availability is updated.

Since the submission cluster forwards jobs based on megahosts, any **PER_HOST** resources are treated as **PER_JOB** resources for forwarding purposes.

Execution clusters

Jobs forwarded to execution clusters have all resource requirements evaluated before being scheduled, including those already considered by the submission cluster when forwarding the job. In the event job resource requirements cannot be met by hosts in the execution cluster, jobs pend. After the maximum pend time configured by **MAX_RSCHED_TIME**, jobs are returned to the submission cluster.

Cluster selection

`cluster preference`

Cluster preference, similar to host preference, allows the LSF administrator to preferentially forward jobs from specific queues to certain clusters. This preference can be defined based on capacity, load, or use policies. For example, a queue with large, high-priority jobs can preferentially forward to a more powerful cluster.

`available slots`

Available slots are idle slots able to run jobs. Clusters with available slots are considered before clusters with pending slots.

`pending slots`

Pending slots are the number of spaces for forwarded jobs that will pend, based on the virtual capacity of the execution cluster set by **XL_RES_RATIO** (`lsb.params`).

`available resources`

Available resources are free and able to run jobs. Clusters with available user-defined shared resources are considered before clusters with pending resources.

`pending resources`

Pending resources are currently in use, but are available in future for forwarded jobs that pend while waiting for resources. The number of user-defined shared pending resources is based on the expanded view of resources in the execution cluster set by **XL_RES_RATIO** (`lsb.params`).

Execution cluster selection order

By default the job forwarding algorithm considers space in the cluster, cluster preference, and resource requirements:

1. Space in the execution clusters is considered:
   - Clusters with available slots come first.
   - Clusters with pending slots come next.
2. If cluster preference is defined:
   - Clusters with available slots are ordered by cluster preference.
   - Clusters with pending slots are ordered by cluster preference.
3. If cluster preference is not defined:
   - Clusters with available slots are ordered by the number of available slots.
   - Clusters with pending slots are ordered by the fraction of pending slots out of the total slots in the cluster.
4. If there are requirements for user-defined shared resources:
   - If there are multiple clusters with available slots, clusters with available resources are considered before clusters with pending resources.
5. If there are resource requirements (for any of the common resources evaluated by the submission cluster), clusters without required resources are not considered.

Set cluster preference

Cluster preference is defined at the queue level in **SNDJOBS_TO** (`lsb.queues`) of the submission cluster for each corresponding execution cluster queue receiving forwarded jobs.

1. For the LSF AE job-forwarding queue, add the cluster preference to the send-jobs queue definition **SNDJOBS_TO** in `lsb.queues`.

   SNDJOBS_TO=[*queue*@]*cluster_name*[+*preference*]...

   For example:

   ```
   SNDJOBS_TO=cluster_e2+1 cluster_e3+2
   ```
2. Run **`badmin reconfig`** for the changes to take effect.

   Note:

   Cluster preference specified at job submission using `bsub -m` *`host_name@cluster_name`* takes precedence over cluster preference set in **SNDJOBS_TO**.

For example:

```
Begin Queue
QUEUE_NAME=Q1
PRIORITY=40
NICE=10
SNDJOBS_TO=cluster_e2+1 Q1@cluster_e3+2
END QUEUE
```

In this example, jobs forwarded from the submission cluster through queue Q1 consider execution clusters in the following order:

- If both cluster_e2 and cluster_e3 have available slots, cluster_e3 comes first because of higher preference.
- If only one cluster has available slots, it is considered first.
- If both cluster_e2 and cluster_e3 have pending slots, cluster_e3 comes first because of higher preference.
- If only one cluster has pending slots, it is considered first

Absolute cluster preference

By default, jobs are forwarded to execution clusters based on a combination of available slots, pending slots, resource requirements, and cluster preference. LSF AE can be configured to only consider resource requirements and cluster preference when forwarding jobs, simplifying the execution cluster selection process.

Set absolute cluster preference:
Cluster preference is defined at the queue level in **`SNDJOBS_TO`** (`lsb.queues`) of the submission cluster for each corresponding execution cluster queue receiving forwarded jobs.

1. For the LSF AE job-forwarding queue, add the cluster preference to the send-jobs queue definition **`SNDJOBS_TO`** in `lsb.queues`.
2. For the LSF AE job-forwarding queue, set **`ABS_CLUSTER_PREFERENCE=Y`** in `lsb.queues`.
3. Run **`badmin reconfig`** for the changes to take effect.

   Note:

   Cluster preference specified at job submission using `bsub -m` *`host_name@cluster_name`* takes precedence over cluster preference set in **`SNDJOBS_TO`**.

For example:

```
Begin Queue
QUEUE_NAME=queue1
PRIORITY=40
NICE=10
SNDJOBS_TO=cluster_e1+1 cluster_e2+2 cluster_e3
ABS_CLUSTER_PREFERENCE=Y
END QUEUE
```

In this example the submission cluster always considers execution clusters based on preference regardless of load. Jobs from queue1 will be sent to cluster_e2 if possible, with cluster_e1 as the second choice and cluster_e3 as the third choice.

Cluster reselection

Jobs are forwarded to the execution cluster based on swp, mem, type, ncpu, model, boolean resources, cpu factors, and user-defined shared resource requirements. In some cases, however, jobs may reach the execution cluster and find other required resources are not available. The cluster reselection policy sets the length of time jobs pend, and allows returned jobs to be forwarded to other execution clusters which may have different resources available.

The time forwarded jobs pend in execution clusters before returning to the submission cluster for rescheduling can be configured for each queue. The wait time before the same job is returned to the same execution cluster can also be set. Both values are defined in **MAX_RSCHED_TIME** in `lsb.queues` and are multiplied by **MBD_SLEEP_TIME** (`lsb.params`):

```
MAX_RSCHED_TIME = timeout [wait_time]
```

timeout

timeout***MBD_SLEEP_TIME** determines how long a job stays pending in the execution cluster before returning to the submission cluster.

The timeout value can be customized based on average job runtime and **XL_RES_RATIO** for the cluster. For example:

timeout = MAX(job_runtime, (**XL_RES_RATIO** - 1) * job_runtime) * 2 / **MBD_SLEEP_TIME**

wait time

wait_time***MBD_SLEEP_TIME** determines how long the submission cluster waits for other execution clusters to become available before returning to the same execution cluster. The wait time only applies when there are execution clusters the job has not yet tried.

The wait time applies when there are execution clusters the job has not yet tried, or when the job cannot be forwarded to those clusters because the matching failed (due to no available slots, forward limit reached, or unsatisfied resource requirements).

Set timeout and wait time

By default, timeout and wait time are set to 360 and 10 respectively, but can be changed to suit each individual job-forwarding queue.

1. For the LSF AE job-forwarding queue define **MAX_RSCHED_TIME** in `lsb.queues`.

   MAX_RSCHED_TIME=*timeout* [*wait_time*]
2. Run **badmin reconfig** for the changes to take effect.

For example:

```
Begin Queue
QUEUE_NAME=Q1
PRIORITY=40
NICE=10
SNDJOBS_TO=cluster_e2+1 cluster_e3
MAX_RSCHED_TIME=50 10
END QUEUE
```

and in `lsb.params`, `MBD_SLEEP_TIME=20`

Thus the cluster reselection timeout and wait time are given by:

timeout = 50x20 seconds = 1000 seconds

wait time = 10x20 seconds = 200 seconds

Jobs submitted to queue Q1 will pend in an execution cluster for up to 1000 seconds before returning to the submission cluster for rescheduling. Jobs can be forwarded to the same cluster once again after 200 seconds.

Fairshare aware job forwarding

In a single LSF cluster, pending jobs are not included in the dynamic priority calculation since all pending jobs have equal footing. In LSF AE this is not the case. Pending jobs in the submission cluster are in a different scheduling phase from pending jobs already forwarded to execution clusters.

Since forwarded pending jobs in execution clusters will run once slots become available, forwarded pending jobs becomes another factor affecting dynamic priority. **`FWD_PEND_JOB_FACTOR`** is used to calculate the dynamic user priority for fairshare jobs to be forwarded from the submission cluster. With **`FWD_PEND_JOB_FACTOR`** defined, user priority is updated based on the number of forwarded pending jobs as well as the number of running jobs in all submission clusters.

Dynamic user priority

LSF calculates a *dynamic user priority* for individual users or for a group, depending on how the shares are assigned. The priority is dynamic because it changes as soon as any variable in formula changes. By default, a user's dynamic priority gradually decreases after a job starts, and the dynamic priority immediately increases when the job finishes.

How LSF AE calculates dynamic priority

By default, LSF calculates the dynamic priority for each user based on:

- The number of shares assigned to the user
- The resources used by jobs belonging to the user:
  - Number of job slots reserved and in use
  - Number of job slots of forwarded jobs
  - Run time of running jobs
  - Cumulative actual CPU time (not normalized), adjusted so that recently used CPU time is weighted more heavily than CPU time used in the distant past

If you enable additional functionality, the formula can also involve additional resources used by jobs belonging to the user:

- Historical run time of finished jobs
- Committed run time, specified at job submission with the **`-W`** option of **`bsub`**, or in the queue with the RUNLIMIT parameter in `lsb.queues`

How Platform LSF measures fairshare resource usage

LSF measures resource usage differently, depending on the type of fairshare:

- For user-based queue-level fairshare, LSF measures the resource consumption of all the user's jobs in the queue. This means a user's dynamic priority can be different in every queue.

Default dynamic priority formula:
By default, LSF calculates dynamic priority according to the following formula:

dynamic priority = *number_shares* / (*cpu_time* * **`CPU_TIME_FACTOR`** + *run_time* * **`RUN_TIME_FACTOR`** + (1 + *job_slots*) * **`RUN_JOB_FACTOR`** + *remote_pend_slots****`FWD_PEND_JOB_FACTOR`**)

Note:

The maximum value of dynamic user priority is 100 times the number of user shares (if the denominator in the calculation is less than 0.01, LSF rounds up to 0.01).

For *cpu_time*, *run_time*, and *job_slots*, LSF uses the total resource consumption of all the jobs in the queue that belong to the user or group.

number_shares

The number of shares assigned to the user.

cpu_time

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

run_time

The total run time of running jobs (measured in hours).

job_slots

The number of job slots in use.

remote_pend_slots

The number of job slots forwarded jobs are waiting to occupy in execution clusters.

Configure the default dynamic priority

If you modify the parameters used in the dynamic priority formula, it affects every fairshare policy in the cluster if you modify the parameters in `lsb.params`, or in the queue if you modify the parameters in `lsb.queues`.

For cross-queue fairshare scheduling, the parameters only take effect on the master queue. Any parameters for the slave queue are ignored.

`CPU_TIME_FACTOR`

The CPU time weighting factor.

Default: 0.7

`RUN_TIME_FACTOR`

The run time weighting factor.

Default: 0.7

`RUN_JOB_FACTOR`

The job slots weighting factor.

Default: 3

`HIST_HOURS`

Interval for collecting resource consumption history

Default: 5

FWD_PEND_JOB_FACTOR

The pending job slots weighting factor.

Default: Set to **RUN_JOB_FACTOR** if not defined.

Generic job forwarding limit

In a single LSF cluster, a limit is a local resource threshold applying to one consumer. In LSF AE this changes, since jobs are forwarded to slots across multiple execution clusters using resources in different places. LSF AE extends LSF resource allocation limits with the new job slot limit FWD_SLOTS, and the new resource consumer CLUSTERS. Job forwarding limits are defined in Limit sections in the `lsb.resources` file.

FWD_SLOTS limits only apply to the resource consumers:

- PROJECTS or PER_PROJECT
- QUEUES or PER_QUEUE
- USERS or PER_USER
- CLUSTERS or PER_CLUSTER

FWD_SLOTS cannot combine with other resources (such as SLOTS or MEM). Combining FWD_SLOTS with other resource consumers results in a warning message, and the limit is ignored.

If CLUSTERS is set to all, '-', or not specified, the limit applies to all execution clusters configured in LSF AE.

Both horizontal and vertical configuration is supported.

Example 1:

```
Begin Limit
USERS QUEUES PROJECTS CLUSTERS    FWD_SLOTS
-     -      proj1    cluster_e1  10
-     -      proj2    cluster_e2  20
-     -      -        all         25
```

In this example proj1 can forward up to 10 jobs to cluster_e1, proj2 can forward up to 20 jobs to cluster_e2, and the total number of forwarded jobs cannot exceed 25 for all clusters.

Example 2:

```
Begin Limit
USERS QUEUES PER_CLUSTER              FWD_SLOTS
-     queue1  (cluster_e1 cluster_e2)  100
```

In this example queue1 can forward up to 100 single jobs to cluster_e1 and 100 jobs to cluster_e2.

View job forwarding limits

When jobs are running in the cluster, **blimits** shows current limit information.

Run **blimits**.
All limits are displayed by default, including job forwarding limits.
The options `-fwd` and `-fwd -C` filter by forwarding limits, and forwarding limits for a specific cluster.

For example:

```
%blimits
INTERNAL RESOURCE LIMITS:
NAME      USERS QUEUES HOSTS PROJECTS SLOTS MEM TMP SWP JOBS
NONAME000 -     normal -      -        1/10  -  -   -   1/10

FORWARD LIMITS:

NAME     USERS QUEUES PROJECTS CLUSTERS   FWD_SLOTS
e1_limit -     -      proj1    cluster_e1 1/10
e2_limit -     -      -        cluster_e2 1/100

REMOTE CLUSTER <cluster_e1>:

INTERNAL RESOURCE LIMITS:

NAME      USERS QUEUES HOSTS PROJECTS SLOTS MEM TMP SWP OBS
NONAME000 -     normal  -     -       1/10  -   -   -   1/10

REMOTE CLUSTER <cluster_e2>:

INTERNAL RESOURCE LIMITS:

NAME     USERS QUEUES HOSTS PROJECTS SLOTS MEM TMP SWP JOBS
NONAME000 -    normal -      -       1/10  -  -   -   1/10
```

```
%blimits -fwd

FORWARD LIMITS:

NAME     USERS QUEUES PROJECTS CLUSTERS   FWD_SLOTS
e1_limit -     -      proj1    cluster_e1 1/10
e2_limit -     -      -        cluster_e2 1/100
```

```
%blimits -fwd -C cluster_e1

FORWARD LIMITS:

NAME     USERS QUEUES PROJECTS CLUSTERS   FWD_SLOTS
e1_limit -     -      proj1    cluster_e1 1/10
```

Convert LSF limits to LSF AE limits

To continue using configured LSF limits in your LSF AE installation, some conversion is necessary. Conversion includes shifting some limits from the execution clusters to the submission cluster.

- Changes SLOTS limits applying to all hosts into FWD_SLOTS limits, and add the CLUSTERS column. For example:

  Before limit conversion, `lsb.resources` for cluster_e1:

  ```
  Begin Limit
  PROJECTS SLOTS USERS HOSTS QUEUES
  all      2000  all   all   (normal short)
  eng2     100   all   all   (normal long)
  End Limit
  ```

  After limit conversion, `lsb.resources` for cluster_sub:

  ```
  Begin Limit
  PROJECTS FWD_SLOTS USERS QUEUES         CLUSTERS
  all      2000      all   (normal short) cluster_e1
  eng2     100       all   (normal long)  cluster_e1
  End Limit
  ```

  Before limit conversion, `lsb.resources`:

```
Begin Limit
PER_USER    QUEUES SLOTS
(all~user1) bulk   300
user1       bulk   600
End Limit
```

After limit conversion, `lsb.resources`:

```
Begin Limit
PER_USER    QUEUES SLOTS CLUSTERS
(all~user1) bulk   300   cluster_e1
user1       bulk   600   cluster_e1
End Limit
```

- Change UJOB_LIMIT into a FWD_SLOTS limit for PER_USER. For example:

  Before limit conversion, `lsb.queues` for cluster_e1:

```
Begin Queue
QUEUE_NAME =interactive
...
QJOB_LIMIT =50
UJOB_LIMIT =5
...
End Queue
```

After limit conversion, the cluster_e1 `lsb.queues` file remains unchanged and limits are added to the `lsb.resources` for cluster_sub:

```
Begin Limit
QUEUES      FWD_SLOTS CLUSTERS
interactive 50        cluster_e1
End Limit
Begin Limit
PER_USER QUEUES      FWD_SLOTS CLUSTERS
all      interactive 5         cluster_e1
End Limit
```

LSF AE update intervals

By default, execution clusters update job usage, resource load, and pending reasons with the submission cluster periodically. You can modify how often LSF AE resource usage is updated. Depending on load, updating the information very frequently can affect the performance of LSF.

Job resource usage updates

To change the timing of rusage updates from execution clusters, set **`MC_RUSAGE_UPDATE_INTERVAL`** in `lsb.params` in the execution cluster. Specify how often to update the information to the submission cluster, in seconds.

mbatchd updates

To set the interval between mbatchd remote host load updates from each execution cluster, set **`XL_RES_UPDATE_INTERVAL`** in `lsb.params` in each execution cluster. The default value is the mbatchd sleep time set by **`MBD_SLEEP_TIME`** (`lsb.params`).

The remote resource information and remote host information returned every **`XL_RES_UPDATE_INTERVAL`** is displayed by **`bhosts`**, and used for job scheduling.

lim updates

The interval between updates made by the submission cluster lim is configured in the submission cluster. Set **`CACHE_INTERVAL`** in the RemoteClusters section of `lsf.cluster.`*`cluster_name`*. The default value for **`CACHE_INTERVAL`** is 60 seconds.

Host load and shared resource updates take place every **CACHE_INTERVAL**. Host updates take place 2x**CACHE_INTERVAL** (120 seconds by default).

The remote resource information collected by the lim is displayed by **lsload** and **lshosts**.

Job pending reason updates

By default, the pending reasons for jobs are updated every 5 minutes by the execution cluster, but the maximum amount of data transferred between clusters is 512 KB. If LSF cannot update the pending reasons for all jobs at once, it will update the additional jobs during the next cycles.

You can disable the feature or modify how often the pending reasons are updated and how much data can be transferred at one time. Depending on load, updating the information very frequently or sending an unlimited amount of information can affect the performance of LSF.

Configure the pending reason update interval

Change the timing of pending reason updating between clusters.

1. Set MC_PENDING_REASON_UPDATE_INTERVAL in `lsb.params` in the execution cluster.
2. Specify how often to update the information in the submission cluster, in seconds.

   To disable pending reason updating between clusters, specify zero:

   ```
   MC_PENDING_REASON_UPDATE_INTERVAL=0
   ```

Configure the pending reason update package size

Change the package size of each pending reason update.

1. Set MC_PENDING_REASON_PKG_SIZE in `lsb.params` in the execution cluster.
2. Specify the maximum package size, in KB.

   To disable the limit and allow any amount of data in one package, specify zero:

   ```
   MC_PENDING_REASON_PKG_SIZE=0
   ```

   This parameter has no effect if pending reason updates are disabled (MC_PENDING_REASON_UPDATE_INTERVAL=0).

Performance and scalability

Several enhancements made to LSF scheduling and job queries have improved the performance and scalability of LSF AE.

Multithread batch queries

Earlier versions of LSF supported multithread for **bjobs** queries only, but not for other query commands. LSF AE now supports multithread batch queries for several other common batch query commands. Only the following batch query commands do not support multithread batch queries:

- **bread**
- **bstatus**
- **tspeek**

The **LSB_QUERY_ENH** parameter (in `lsf.conf`) extends multithreaded query support to other batch query commands in addition to **bjobs**. In addition, the **mbatchd** query starts automatically instead of being triggered by a query request. This ensures a consistent query response time within the system.

To extend multithread queries to other batch query commands, set `LSB_QUERY_ENH=Y` in `lsf.conf` and run **badmin mbdrestart** for the change to take effect.

Parallel mbatchd restart

In a large cluster, restarting LSF can take several minutes. LSF AE introduces a parallel restart option, minimizing system downtime when loading new configuration.

Command option **badmin mbdrestart -p** spawns a parallel **mbatchd** to load new configuration files and replay events while the old **mbatchd** continues to run. As a result, LSF is only unavailable for a short time while the old and new **mbatchd** merge, transferring recently logged events to the new **mbatchd** and shutting down the old **mbatchd**.

When a parallel **mbatchd** restart is in progress, the most recent **mbatchd** start time, the PID of the currently active **mbatchd**, the time of the latest **mbatchd** reconfiguration, the start time of the new **mbatchd**, and the PID of the new **mbatchd** are displayed in **badmin showstatus** output:

```
% badmin showstatus
...
    Latest mbatchd start:       Thu Nov 22 21:17:01 2012
    Active mbatchd PID:         26283

    Latest mbatchd reconfig:    Thu Nov 22 21:18:06 2012

    mbatchd restart information
        New mbatchd started:    Thu Nov 22 21:18:21 2012
        New mbatchd PID:        27474
```

Parallel **mbatchd** restart cannot be used with duplicate event logging (**LSB_LOCALDIR** in `lsf.conf`).

Multithread notification for bsub -K

If your cluster runs a large amount of blocking mode jobs (**bsub -K**), response to batch queries can become very slow. In LSF AE, the number of callback threads to use for batch queries is configurable.

By default, `LSB_NUM_NIOS_CALLBACK_THREADS=4` (`lsf.conf`) specifies the number of callback threads to use for batch queries. If you run a large number of **bsub -K** jobs, you can increase this number.

Job dependency evaluation

In order to speed up each scheduling cycle, the number of job dependencies evaluated within each scheduling cycle is now configurable. This limits the time spent reevaluating job dependencies when there is a large number of pending jobs in the system.

In each scheduling cycle, LSF AE evaluates the number of job dependencies given by **EVALUATE_JOB_DEPENDENCY** in `lsb.params` (50k by default). The job dependency evaluation resumes from the stopping point in the next scheduling cycle. Once every 10 minutes the complete job list is evaluated for dependencies regardless of the **EVALUATE_JOB_DEPENDENCY** setting.

Multithread resource requirement evaluation

In order to speed up job resource requirement evaluation in the scheduling cycle, the number of scheduler threads evaluating resource requirements is now configurable.

By default, `SCHEDULER_THREADS=0` (`lsb.params`) runs a single thread. If you are running on multi-core machines, you can increase this number to match the number of cores assigned to the scheduler by **`LSF_DAEMONS_CPUS`** (`lsf.conf`).

Job dispatch by queue

To speed up job scheduling decisions, the scheduler can broadcast scheduling decisions for each queue instead of waiting for the entire scheduling cycle to complete.

Decisions for each queue are final for the scheduling cycle, and may not optimize allocation for certain cases such as preemption. For example, jobs cannot be preempted until the next scheduling cycle.

Set job dispatch by queue

1. For the LSF AE job-forwarding queue, define `DISPATCH_BY_QUEUE=Y` in `lsb.queues`.
2. Run **`badmin reconfig`** for the changes to take effect.

Minimize job dispatch time in a high priority queue

A cluster may have certain jobs that require a minimal job dispatch time, such as interactive jobs. LSF AE allows the configuration of a high priority queue for enhanced dispatch to minimize the job dispatch time.

Jobs submitted to a queue configured for enhanced dispatch are among the first to be scheduled because LSF normally schedules jobs in high priority queues first. In a queue configured for enhanced dispatch, after processing the queue and collecting the scheduling decisions for that queue, the scheduler publishes the decisions immediately instead of waiting until all the queues are processed. Jobs in this queue are dispatched immediately without waiting until the end of the scheduling session. This minimizes the dispatch time of a job submitted to this queue.

Configure a queue for enhanced dispatch to minimize job dispatch time

1. For a LSF AE high priority queue, define `DISPATCH_BY_QUEUE=Y` in `lsb.queues`.
2. Run **`badmin reconfig`** for the changes to take effect.

Job information directory

The LSF job information directory is used to store job definition files. By default is is defined as `$LSB_SHAREDIR/`*`cluster_name`*`/logdir/info`.

The large number of jobs supported by LSF AE results in a high number of file operations occurring in the **`LSB_SHAREDIR`** directory. To alleviate some of this load on the shared file system, the job information directory can be configured separately from other files in the LSF working directory using the optional parameter **`LSB_JOBINFO_DIR`** (`lsf.conf`).

The directory specified by **`LSB_JOBINFO_DIR`** in the submission cluster must be owned by the primary LSF administrator, and be accessible with read, write, and execute access from all potential master hosts in all clusters.

Set job information directory

1. In the submission cluster, define `LSB_JOBINFO_DIR=`*`directory`* in `lsf.conf`.
2. Run **`badmin mbdrestart`** for the changes to take effect.

Change job information directory for a running cluster

1. Run `badmin hshutdown` *`master_host`* and `badmin mbdrestart` to shut down the **`mbatchd`** on the master host and prevent new job submission.

   Note:

   The badmin mbdrestart -p parallel option cannot be used in this case.
2. Create the new directory.
3. Copy all existing job files and sub directories under `$LSB_SHAREDIR/`*`cluster_name`*`/logdir/info` into the new directory.
4. In the submission cluster, define `LSB_JOBINFO_DIR=`*`directory`* in `lsf.conf`.
5. Run `badmin hstartup` *`master_host`* for the changes to take effect.

Optimized fairshare job selection

As LSF AE is expected to support more users and larger fairshare trees than individual LSF clusters, fairshare performance has been improved based on the user group of jobs submitted. No special configuration is needed.

- Empty share accounts (not containing jobs) are marked so they are only evaluated once in the scheduling cycle when jobs are submitted with **`bsub -G`**.
- Jobs for each queue are sorted by user during the scheduling process, allowing for faster job selection once a user is selected.

Chapter 4. Using LSF AE

Job submission

Job submission through LSF AE maintains a single cluster view, with enhancements to support remote hosts and clusters. You can select the remote cluster or host best suited for your jobs at job submission, using the **`bsub -m`** command option.

`submit to a remote cluster`

Using **`bsub -m`**, specify a cluster.

The job is forwarded to the specified cluster, and scheduled to run on a host within the cluster.

`submit to a remote host`

Using **`bsub -m`**, specify a host.

The job is forwarded to the cluster containing the host you specify, and runs on the host when it becomes available. Cluster names do not need to be included if the host names are unique.

Select remote cluster

Specify a cluster at job submission.

Submit your job using `bsub -m` *`cluster_name`*.
The job is forwarded to the selected cluster for scheduling.

For example:

```
bsub -m "cluster3"
```

The job is sent to cluster3 to run on any host in the cluster.

Select remote host

Specify a host at job submission.

Submit a job using `bsub -m` *`host_name`*.

For example:

```
bsub -m "hostA"
```

The submission cluster forwards the job to the cluster containing hostA.

Set job-level host or cluster preference

The enhanced **`bsub -m`** option for LSF AE supports setting remote host and cluster preference for individual jobs. Host preference can be set for remote hosts in any cluster.

Submit a job using `bsub -m` *`cluster_name`* or `bsub -m` *`host_name@cluster_name`*.

Note:

Cluster preference specified at job submission using `bsub -m` *`host_name@cluster_name`* or `bsub -m` *`cluster_name`* takes precedence over cluster preference set in **`SNDJOBS_TO`**.

For example:

```
bsub -m "cluster3 cluster2+1"
```

In this example the submission cluster first considers cluster2, with cluster3 as the second choice.

```
bsub -m "hostA hostB+1"
```

The submission cluster first considers the cluster containing hostB, with the cluster containing hostA as the second choice.

```
bsub -m "cluster3 cluster4+1"
```

The submission cluster first considers cluster4 , with cluster3 as the second choice.

```
bsub -m "hostC@cluster1+1 hostD@cluster1+2 cluster1"
```

All hosts in this example are in cluster1. The cluster selection is combined with the host preference; the submission cluster forwards the job to cluster1 with the filtered and merged host preference `"others hostC+1 hostD+2"`.

For further information on **`bsub -m`** see "LSF AE commands" on page 31.

Submit jobs directly from active running jobs

LSF AE supports commands issued by jobs running on execution clusters. **`bsub`** commands issued by running jobs are sent to the submission cluster, and then forwarded to an appropriate execution cluster to run.

This is necessary since LSF AE execution clusters do not accept local job submissions.

Query commands

By default, job query commands display remote information for the entire LSF AE installation, including information about remote hosts, remote host groups, and remote limits.

Supported LSF query commands (and new options):

- **`lsclusters`**
- **`lshosts`** (**`-cname`**)
- **`lsload`** (**`-cname`**)
- **`bclusters`**
- **`bhosts`** (**`-cname`**)
- **`bmgroup`** (**`-cname`**)
- **`bjobs`** (**`-cname`**, **`-fwd`**, **`-sum`**,)
- **`blimits`** (**`-fwd -C`**)

For full descriptions of all LSF AE commands along with sample output see "LSF AE commands" on page 31.

Jobs running queries

LSF AE supports some queries submitted by running jobs. Most **mbatchd** queries (b* commands) made by running jobs are forwarded to the submission cluster, maintaining the single cluster view.

The following commands are not sent to the submission cluster, and instead run locally in the execution cluster:

- **bhist**
- **bacct**
- **brlainfo**

Job control

The single cluster view in LSF AE extends to job control commands, which are either run through the submission cluster, or are sent back to the submission cluster by LSF AE when run locally.

Run these LSF job control commands (and changed options) from any cluster:

- **bstop**: suspend a job
- **bresume**: resume a suspended job
- **bkill**: kill a job
- **badmin diagnose**/**hopen**/**hclose**: forward requests through the submission cluster to the appropriate remote host orcluster

Run these LSF job control commands only from the submission cluster:

- **brun** (**-m**): force a pending or finished job to run or be forwarded to a specified cluster
- **bbot**: send a job back to the submission cluster for rescheduling, at the bottom of the queue
- **brequeue**: requeue a job

Run these LSF job control commands only from the submission cluster. Forwarded jobs are not returned to the submission cluster, and any changes to forwarded jobs are restricted to the same execution cluster:

- **bmod**: modify job submission options
- **bswitch**: switch queues
- **btop**: change job order

For example:

Run `bmod -m` *`host_name`* on a job pending in the submission cluster (before forwarding), and specify any host.

Run `bmod -m` *`host_name`* on a job pending in an execution cluster (after forwarding), and specify a host in the same execution cluster.

Run **btop** on a job pending in an execution cluster to send to the top of the queue in the same execution cluster.

For full descriptions of all LSF AE commands along with sample output see “LSF AE commands” on page 31.

Chapter 5. Reference

LSF AE commands

The single cluster view maintained by LSF AE gives end users a single-cluster experience using existing Platform LSF commands. Additional command options allow administrators to track job progress through to the execution clusters.

For the output examples shown, consider a submission cluster with two execution clusters:

- Submission cluster: `sub_cluster`

  Master host: `hostA`

  Slave hosts: no.
- First execution cluster: `cluster2`

  Master host: `hostB`

  Slave hosts: no
- Second execution cluster: `cluster3`

  Master host: `hostC`

  Slave hosts: `hostD`

lsclusters

Displays information and current status for the submission cluster and all configured execution clusters.

This command runs locally (in the execution cluster) when submitted by a running job.

```
% lsclusters
CLUSTER_NAME STATUS MASTER_HOST ADMIN    HOSTS SERVERS
sub_cluster   ok     hostA       lsfadmin 1     1
cluster2     ok     hostB       lsfadmin 1     1
cluster3     ok     hostC       lsfadmin 2     2
```

lshosts

Shows host information for all hosts within all clusters connected by LSF AE.

This command runs locally (in the execution cluster) when submitted by a running job.

```
% lshosts
HOST_NAME type   model    cpuf  ncpus maxmem maxswp server RESOURCES
hostA     X86_64 PC6000   116.1 2     1000M  1983M  Yes    (mg)
hostB     X86_64 Opteron2 47.0  1     1000M  1961M  Yes    (mg)
hostC     X86_64 Intel_EM 60.0  2     7978M  1992M  Yes    (mg)
hostD     X86_64 Intel_EM 60.0  2     3828M  2055M  Yes    (mg)
```

-cname

New option **-cname** includes the cluster name for execution cluster hosts and host groups in output for **lshosts**.

```
% lshosts -cname
HOST_NAME      type   model    cpuf  ncpus maxmem maxswp server RESOURCES
hostA          X86_64 PC6000   116.1 2     1000M  1983M  Yes    (mg)
```

```
hostB@cluster2 X86_64 Opteron2 47.0  1     1000M  1961M  Yes    (mg)
hostC@cluster3 X86_64 Intel_EM 60.0  2     7978M  1992M  Yes    (mg)
hostD@cluster3 X86_64 Intel_EM 60.0  2     3828M  2055M  Yes    (mg)
```

lsload

Shows host load information for all hosts within all clusters connected by LSF AE.

This command runs locally (in the execution cluster) when submitted by a running job.

```
% lsload
HOST_NAME status r15s r1m r15m ut  pg  ls it tmp   swp   mem
hostA     ok     0.2  0.0 0.0  1%  0.0 6  4  22G   1948M 725M
hostB     ok     0.6  0.0 0.0  0%  0.0 2  0  18M   1554M 518M
hostC     ok     0.4  0.1 0.0  7%  0.0 3  0  3270M 1990M 5448M
hostD     ok     1.0  1.0 1.1  13% 0.0 8  20 21G   353M  491M
```

-cname

New option **-cname** includes the cluster name for execution cluster hosts and host groups in output for **lsload**.

```
% lsload -cname
HOST_NAME      status r15s r1m r15m ut  pg  ls it tmp   swp   mem
hostA          ok     0.2  0.0 0.0  1%  0.0 6  4  22G   1948M 725M
hostB@cluster2 ok     0.6  0.0 0.0  0%  0.0 2  0  18M   1554M 518M
hostC@cluster3 ok     0.4  0.1 0.0  7%  0.0 3  0  3270M 1990M 5448M
hostD@cluster3 ok     1.0  1.0 1.1  13% 0.0 8  20 21G   353M  491M
```

bacct

Displays accounting statistics for finished jobs within all clusters connected by LSF AE.

This command runs locally (in the execution cluster) when submitted by a running job.

badmin diagnose/hopen/hclose

Forwards requests made through the submission cluster to the appropriate remote host or cluster. Not supported within the local execution cluster.

badmin mbdrestart -p

Starts mbatchd in parallel, leaving the existing mbatchd free to respond to queries and run commands while the new mbatchd restarts, reading configuration files and replaying events. Once all events have been read mbatchd daemons merge, replaying new events and leaving only the new mbatchd running.

During a parallel mbatchd restart, new **badmin mbdrestart** and **badmin reconfig** commands are not accepted. Parallel mbatchd restart using **badmin mbdrestart -p** does not work with duplicate event logging (**LSB_LOCALDIR** in `lsf.conf`).

The existing command **badmin mbdrestart** remains unchanged.

badmin showstatus

In the submission cluster, displays a summary of the current LSF runtime information about the submission cluster and all execution clusters. In the execution cluster, displays a summary of the current LSF runtime information about the local execution cluster only.

The current LSF runtime information displayed includes information about hosts, jobs, users, user groups, and **mbatchd** startup and reconfiguration.

bbot

Sends the job back to the submission cluster for rescheduling, at the bottom of the queue.

Run **bbot** commands through the submission cluster, not through individual execution clusters.

bclusters

Displays execution cluster resource provider and consumer information, resource flow information, and connection status between the submission cluster and execution cluster.

Use **-app** to view available application profiles in remote clusters.

Information related to LSF AE is displayed under the heading Job Forwarding Information.

- LOCAL_QUEUE: Name of a LSF AE queue.
- JOB_FLOW: Indicates direction of job flow.
  - send

    The local queue is a submission cluster send-jobs queue (SNDJOBS_TO is defined in the local queue).
  - recv

    The local queue is an execution cluster receive-jobs queue (RCVJOBS_FROM is defined in the local queue).
- REMOTE: Shows the name of the remote queue, always the same as the local queue name.

  For receive-jobs queues, always "-".
- CLUSTER: For send-jobs queues, shows the name of the execution cluster containing the receive-jobs queue.

  For receive-jobs queues, shows the name of the submission cluster that can send jobs to the local queue.
- STATUS: Indicates the connection status between the local queue and remote queue.
  - ok

    The submission cluster and execution cluster can exchange information and the system is properly configured.
  - disc

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

    The remote queue rejects jobs from the send-jobs queue. The local queue and remote queue are connected and the clusters communicate, but the queue-level configuration is not correct.

```
% bclusters
[Job Forwarding Information ]
LOCAL_QUEUE JOB_FLOW REMOTE CLUSTER  STATUS
queue1      send     queue1 cluster2 ok
```

```
queue1      send     queue1 cluster3 ok
[Resource Lease Information ]
No resources have been exported or borrowed
```

bconf

Submits live reconfiguration requests, updating configuration settings in active memory without restarting daemons.

***action object_type=identity* ["*key-value_pair*[;*key-value_pair*...]"] [-c "*comments*"] [-f]**

The limit *object_type* has enhanced support for the following LSF AE *key-value_pair* keywords in `lsb.resources` for generic job forwarding limits:

- CLUSTERS for the following *action* types: addmember, rmmember, update, create, delete
- PER_CLUSTER for the following *action* types: addmember, rmmember, update, create, delete
- FWD_SLOTS for the following *action* types: update, create, delete

Example:

```
bconf update limit=cluster_e1 "FWD_SLOTS=10"
```

bhist

Displays historical information for all jobs within all clusters connected by LSF AE.

This command runs locally (in the execution cluster) when submitted by a running job.

bhosts/bmgroup

Displays information for all hosts within all clusters connected by LSF AE. This includes the submission cluster and execution clusters. The same format is used for all hosts, both in the submission cluster and in the execution clusters. The output displayed is sorted by host or host group name.

Execution cluster hosts and host groups can be identified by name only (*host_name*), or by name and cluster. (*host_name@cluster_name*). If there are multiple hosts or host groups with the same name, all are displayed.

```
% bhosts
HOST_NAME STATUS JL/U MAX NJOBS RUN SSUSP USUSP RSV
hostA     ok     -    2   0     0   0     0     0
hostB     ok     -    1   0     0   0     0     0
hostC     ok     -    2   0     0   0     0     0
hostD     ok     -    2   0     0   0     0     0
```

```
% bmgroup -l
GROUP_NAME    CONDENSE   HOSTS
hgroup1       Yes        hostB
hgroup2       No         hostC hostD
```

-cname

Includes the cluster name for execution cluster hosts and host groups in output for **bhosts** and **bmgroup**. The output displayed is sorted by cluster and then by host or host group name.

```
% bhosts -cname
HOST_NAME      STATUS JL/U MAX NJOBS RUN SSUSP USUSP RSV
hostA          ok     -    2   0     0   0     0     0
```

```
hostB@cluster2 ok     -    1   0     0   0     0     0
hostC@cluster3 ok     -    2   0     0   0     0     0
hostD@cluster3 ok     -    2   0     0   0     0     0
```

```
% bmgroup -l -cname
GROUP_NAME          CONDENSE   HOSTS
hgroup1             Yes        hostB
hgroup2@Cluster3    No         hostC hostD
```

bjobs

Displays job status, adds job forward information and supports the **-fwd** filter option.

```
% bjobs
JOBID USER    STAT QUEUE  FROM_HOST EXEC_HOST JOB_NAME   SUBMIT_TIME
1     lsfuser RUN  queue1 hostA     hostC     sleep 1234 Nov 29  1 14:08
```

-l

Displays forwarding time and cluster name for forwarded pending and running jobs.

```
% bjobs -l
Job <1>, User <lsfuser>, Project <default>, Status <RUN>, Queue <queue1>,
Command <sleep 1234>
Mon Nov 29 14:08:35: Submitted from host <hostA>, CWD </home/lsfuser >,
                     Re-runnable;
Mon Nov 29 14:08:38: Job <1> forwarded to cluster <cluster3> as Job <1>;
Mon Nov 29 14:08:44: Started on <hostC>, Execution Home </home/lsfuser>,
                     Execution CWD </home/lsfuser>;
Mon Nov 29 14:08:46: Resource usage collected.
 MEM: 2 Mbytes;  SWAP: 32 Mbytes;  NTHREAD: 1
 PGID: 6395;  PIDs: 6395

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

-cname

Includes the cluster name for execution cluster hosts in output for **bjobs -l**.

```
% bjobs -l -cname
Job <1>, User <lsfuser>, Project <default>, Status <RUN>, Queue <queue1>,
Command <sleep 1234>
Mon Nov 29 14:08:35: Submitted from host <hostA>, CWD </home/lsfuser >,
                     Re-runnable;
Mon Nov 29 14:08:38: Job <1> forwarded to cluster <cluster3> as Job <1>;
Mon Nov 29 14:08:44: Started on <hostC@cluster3>, Execution Home
                     </home/lsfuser>, Execution CWD </home/lsfuser>;
Mon Nov 29 14:08:46: Resource usage collected.
 MEM: 2 Mbytes;  SWAP: 32 Mbytes;  NTHREAD: 1
 PGID: 6395;  PIDs: 6395

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

-sum

Displays summary information about unfinished jobs. **bjobs -sum** displays the count of job slots in the following states: running (RUN), system suspended (SSUSP), user suspended (USUSP), pending (PEND), forwarded to remote clusters and pending (FWD_PEND), and UNKNOWN.

bjobs -sum displays the job slot count only for the user's own jobs.

```
% bjobs -sum
RUN        SSUSP       USUSP      UNKNOWN    PEND       FWD_PEND
123        456         789        5          5          3
```

Use **-sum** with other options (like **-m**, **-P**, **-q**, and **-u**) to filter the results. For example, **bjobs -sum -u user1** displays job slot counts just for user user1.

```
% bjobs -sum -u user1
RUN        SSUSP       USUSP      UNKNOWN    PEND       FWD_PEND
20         10          10         0           5          0
```

blimits

Displays cluster limit data across all clusters, including forward limits and aggregated execution cluster limits. All clusters are shown by default.

In order to specify a remote limit with the **-n** option, include the cluster name as well as the limit name: blimits -n *limit_name@cluster_name*.

```
% blimits
Sub-master: blimits

FORWARD LIMITS:

    NAME          USERS            QUEUES          PROJECTS         CLUSTERS        FWD_SLOTS
 NONAME000          -                -               proj1       ec1 ec2       1/5
 NONAME001          -              queue1              -         ec1 ec2       2/7
 NONAME002          -                -                 -                -               2/10

REMOTE CLUSTER <ec1>:

INTERNAL RESOURCE LIMITS:

    NAME        USERS          QUEUES         HOSTS          PROJECTS    SLOTS   MEM    TMP    SWP    JOBS
 NONAME000        -            queue1         all              -         2/10    -      -      -      -

REMOTE CLUSTER <ec2>:

No resource usage found.
```

-fwd

Displays forward slot allocation limits.

Use **-fwd** with **-c** to display forward slot limit configuration.

-fwd -C *cluster_name*...

Displays forward slot allocation limits for one or more specific clusters. **-C** cannot be used without **-fwd**.

Use **-fwd -C** with **-c** to display forward slot limit configuration for the specified cluster.

bmod

Modifies job submission options. Forwarded jobs can only be modified within one execution cluster. Changes across execution clusters are not supported.

Run **bmod** commands through the submission cluster, not through individual execution clusters.

bparams

Displays new LSF AE parameters.

bqueues

Displays status of queues in the cluster, including all running and pending jobs counters, and fairshare information.

Enhanced output for LSF AE displays information about forwarded pending jobs for each queue in the SHARE_INFO_FOR section under the heading FWD_PEND.

```
% bqueues -lr queue1

QUEUE: queue1
 -- No description provided.

PARAMETERS/STATISTICS
PRIO NICE STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV
 40   10  Open:Active       -    -    -    -     1     0     1     0     0    0
Interval for a host to accept two jobs is 0 seconds

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

SCHEDULING POLICIES:  FAIRSHARE  ABS_CLUSTER_PREFERENCE
USER_SHARES:  [default, 1]

SHARE_INFO_FOR: queue1/
 USER/GROUP   SHARES  PRIORITY  STARTED  RESERVED  CPU_TIME  RUN_TIME  FWD_PEND
lsfuser          1      0.167      1        0         0.0        0        0

USERS: all
HOSTS:  none
REQUEUE_EXIT_VALUES:  1
SEND_JOBS_TO:  queue1@cluster2, queue1@cluster3
RERUNNABLE :  yes

MAX_RSCHED_TIME: 360 10
```

brequeue

Requeues a job.

By default, when a job running on an execution cluster is requeued it returns to the submission cluster in the PEND state awaiting rescheduling.

Automatic job requeue using **REQUEUE_EXIT_VALUES** (in `lsb.queues`) returns a running job to the PEND state in the same execution cluster for local rescheduling.

```
% bhist -l 887

Job <887>, User <lsfadmin>, Project <default>, Command <sleep 10000>
Wed Dec  8 19:37:21: Submitted from host <hostD>, to Queue <test1>, CWD <$HOME>,
                     Requested Resources <type==any>;
Wed Dec  8 19:37:25: Forwarded job to cluster cluster2;
Wed Dec  8 19:37:28: Dispatched to <hostB@cluster3>;
Wed Dec  8 19:37:28: Starting (Pid 22557);
Wed Dec  8 19:37:28: Running with execution home </home/lsfadmin>, Execution CWD
                     </home/lsfadmin>, Execution Pid <22557>;
Wed Dec  8 19:37:29: Signal <REQUEUE_PEND> requested by user or administrator
                     <lsfadmin>;
Wed Dec  8 19:37:30: Exited with exit code 130. The CPU time used is 0.1 seconds;
Wed Dec  8 19:38:31: Pending: Job has been requeued;
Wed Dec  8 19:38:31: Forwarded job to cluster cluster3;
Wed Dec  8 19:38:33: Dispatched to <hostD@cluster3>;
Wed Dec  8 19:38:33: Starting (Pid 24115);
Wed Dec  8 19:38:33: Running with execution home </home/lsfadmin>, Execution CWD
```

```
                     </home/lsfadmin>, Execution Pid <24115>;

Summary of time in seconds spent in various states by  Wed Dec  8 19:39:24
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  70       0        53       0        0        0        123
```

Automatic requeue example with **REQUEUE_EXIT_VALUES** defined:

```
% bhist -l 889

Job <889>, User <lsfadmin>, Project <default>, Command <sleep 5; exit 13>
Wed Dec  8 20:03:07: Submitted from host <hostD>, to Queue <test1>, CWD <$HOME>,
                     Requested Resources <type==any>;
Wed Dec  8 20:03:13: Forwarded job to cluster ecluster2;
Wed Dec  8 20:03:19: Dispatched to <host3@ecluster2>;
Wed Dec  8 20:03:19: Starting (Pid 29850);
Wed Dec  8 20:03:19: Running with execution home </home/lsfadmin>, Execution CWD
                     </home/lsfadmin>, Execution Pid <29850>;
Wed Dec  8 20:03:24: Pending: Job is requeued on the execution cluster due to exit
                     value;
Wed Dec  8 20:03:29: Dispatched to <host3@ecluster2>;
Wed Dec  8 20:03:31: Starting (Pid 29866);
Wed Dec  8 20:03:31: Running with execution home </home/lsfadmin>, Execution CWD
                     </home/lsfadmin>, Execution Pid <29866>;
Wed Dec  8 20:03:34: Pending: Job is requeued on the execution cluster due to exit
                     value;
```

brlainfo

Displays host topology information for hosts within all clusters connected by LSF AE.

This command runs locally (in the execution cluster) when submitted by a running job.

brun

Forces a pending or finished job to run or be forwarded to a specified cluster. The exact behavior of **brun** on a pending job depends on where the job is pending, and which hosts or clusters are specified in the **brun** command.

Important:

Only administrators can use the **brun** command. You can only run brun from the submission cluster.

You must specify one or more host names or a cluster name when you force a job to run.

If multiple hosts are specified, the first available host is selected and the remainder ignored. Specified hosts cannot belong to more than one cluster.

You can only specify one cluster name. The job is forced to be forwarded to the specified cluster.

You cannot specify host names and cluster names together in the same **brun** command.

A job pending in an execution cluster forced to run in a different cluster is returned to the submission cluster, and then forwarded once again.

If a job is submitted with a cluster name and the job is forwarded to a remote cluster, you cannot use **brun -m** again to switch the job to another execution cluster. For example:

```
bsub -m cluster1 -q test1 sleep 1000
```

The job is pending on cluster1. Running **brun** again to forward the job to cluster2 is rejected:

```
brun -m cluster2 1803
Failed to run the job: Hosts requested do not belong to the cluster
```

For example:

```
brun -m "host12 host27"
```

In this example, if host12 is available the job is sent to the cluster containing host12 and tries to run. If unsuccessful, the job pends in the cluster containing host12. If host12 is not available, the job is sent to the cluster containing host27 where it runs or pends.

Force a job to run on a specific host

Local host specified

Job runs locally. For example:

```
brun -m hostA 246
Job <246> is being forced to run or forwarded.
bjobs 246
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
246     user1   RUN   normal     hostD       hostA       *eep 10000 Jan  3 12:15
bhist -l 246
Job <246>, User <user1>, Project <default>, Command <sleep 10000>
Mon Jan  3 12:15:22: Submitted from host <hostD>, to Queue <normal>, CWD
                     <$HOME/envs>, Requested Resources <type == any>;
Mon Jan  3 12:16:13: Job is forced to run or forwarded by user or administrator
                     <user1>;
Mon Jan  3 12:16:13: Dispatched to <hostA>;
Mon Jan  3 12:16:41: Starting (Pid 10467);
Mon Jan  3 12:16:59: Running with execution home </home/user1>, Execution CWD
                     </home/user1/envs>, Execution Pid <10467>;
```

Host in execution cluster specified

Job is forwarded to execution cluster containing specified host, and runs.

For example:

```
brun -m hostB 244
Job <244> is being forced to run or forwarded.
bjobs 244
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
244     user1   RUN   normal     hostD       hostB       *eep 10000 Jan  3 12:15
bhist -l 244

Job <244>, User <user1>, Project <default>, Command <sleep 10000>
Mon Jan  3 12:15:22: Submitted from host <hostD>, to Queue <normal>, CWD
                     <$HOME/envs>, Requested Resources <type == any>;
Mon Jan  3 12:19:18: Job is forced to run or forwarded by user or administrator
                     <user1>;
Mon Jan  3 12:19:18: Forwarded job to cluster cluster2;
Mon Jan  3 12:19:18: Remote job control initiated;
Mon Jan  3 12:19:18: Dispatched to <hostB>;
Mon Jan  3 12:19:18: Remote job control completed;
Mon Jan  3 12:19:19: Starting (Pid 28804);
Mon Jan  3 12:19:19: Running with execution home </home/user1>, Execution CWD
                     </home/user1/envs>, Execution Pid <28804>;
```

Host in same execution cluster specified

Job runs on the specified host in the same execution cluster. For example:

```
brun -m hostB 237
Job <237> is being forced to run or forwarded.

bjobs 237
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
237     user1   RUN   normal     hostD       hostB       *eep 10000 Jan  3 12:14

bhist -l 237

Job <237>, User <user1>, Project <default>, Command <sleep 10000>
Mon Jan  3 12:14:48: Submitted from host <hostD>, to Queue <normal>, CWD
                     <$HOME/envs>, Requested Resources <type == any>;
Mon Jan  3 12:14:53: Forwarded job to cluster cluster2;
Mon Jan  3 12:22:08: Job is forced to run or forwarded by user or administrator
                     <user1>;
Mon Jan  3 12:22:08: Remote job control initiated;
Mon Jan  3 12:22:08: Dispatched to <hostB>;
Mon Jan  3 12:22:09: Remote job control completed;
Mon Jan  3 12:22:09: Starting (Pid 0);
Mon Jan  3 12:22:09: Starting (Pid 29073);
Mon Jan  3 12:22:09: Running with execution home </home/user1>, Execution CWD
                     </home/user1/envs>, Execution Pid <29073>;
```

Host in submission cluster specified

Job runs on the specified host in the submission cluster. For example:

```
brun -m hostA 238
Job <238> is being forced to run or forwarded.

bjobs 237
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
238     user1   RUN   normal     hostB       hostA       *eep 10000 Oct  5 11:00

bhist -l 237

Job <237>, User <user1>, Project <default>, Command <sleep 10000>
Wed Oct  5 11:00:16: Submitted from host <hostB>, to Queue <normal>, CWD

                     </usr/local/xl/conf>, Requested Resources <type == any>;
Wed Oct  5 11:00:18: Forwarded job to cluster ec1;
Wed Oct  5 11:00:46: Job is forced to run or forwarded by user or administrator
                     <user1>;
Wed Oct  5 11:00:46: Pending: Job has returned from remote cluster;
Wed Oct  5 11:00:46: Dispatched to <hostA>;
Wed Oct  5 11:00:46: Starting (Pid 15686);
Wed Oct  5 11:00:47: Running with execution home </home/user1>, Execution CWD
                     </usr/local/xl/conf>, Execution Pid <15686>;

Summary of time in seconds spent in various states by  Wed Oct  5 11:01:06
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  30       0        20       0        0        0        50
```

Force a job to run in a specific cluster

Host in different execution cluster specified

Job returns to submission cluster, is forwarded to execution cluster containing specified host, and runs.

```
brun -m ec2-hostA 3111
Job <3111> is being forced to run or forwarded.

bjobs 3111
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
3111    user1 RUN   queue1   sub-master ec2-hostA       sleep 1000 Feb 23 11:21

bhist -l 3111

Job <3111>, User <user1>, Project <default>, Command <sleep 1000>
Wed Feb 23 11:21:00: Submitted from host <sub-master>, to Queue <queue1>, CWD
                     </usr/local/xl/conf>;
Wed Feb 23 11:21:03: Forwarded job to cluster cluster1;
Wed Feb 23 11:21:58: Job is forced to run or forwarded by user or administrator
                     <user1>;
Wed Feb 23 11:21:58: Pending: Job has returned from remote cluster;
Wed Feb 23 11:21:58: Forwarded job to cluster cluster2;
Wed Feb 23 11:21:58: Remote job run control initiated;
Wed Feb 23 11:21:59: Dispatched to <ec2-hostA>;
Wed Feb 23 11:21:59: Remote job run control completed;
Wed Feb 23 11:21:59: Starting (Pid 3257);
Wed Feb 23 11:21:59: Running with execution home </home/user1>, Execution CWD
                     </usr/local/xl/conf >, Execution Pid <3257>;

Summary of time in seconds spent in various states by  Wed Feb 23 11:24:59
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  59       0        180      0        0        0        239
```

Job already forwarded to execution

Job has already been forwarded to an execution cluster, and you specify a different execution cluster. The job returns to submission cluster, and is forced to be forwarded to the specified execution cluster. The job is not forced to run in the new execution cluster. After the job is forwarded, the execution cluster schedules the job according to local policies.

For example:

```
brun -m cluster2 244
Job <244> is being forced to run or forwarded.
bjobs 244
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
244     user1   RUN   normal     hostD     hostB    *eep 10000 Jan  3 12:15
bhist -l 244

Job <244>, User <user1>, Project <default>, Command <sleep 10000>
Mon Jan  3 12:15:22: Submitted from host <hostD>, to Queue <normal>, CWD
                     <$HOME/envs>, Requested Resources <type == any>;
Mon Jan  3 12:15:25: Forwarded job to cluster cluster1;
Mon Jan  3 12:19:18: Job is forced to run or forwarded by user or administrator
                     <user1>;
Mon Jan  3 12:19:18: Pending: Job has returned from remote cluster;
Mon Jan  3 12:19:18: Forwarded job to cluster cluster2;
Mon Jan  3 12:19:18: Dispatched to <hostB>;
Mon Jan  3 12:19:19: Starting (Pid 28804);
Mon Jan  3 12:19:19: Running with execution home </home/user1>, Execution CWD
                     </home/user1/envs>, Execution Pid <28804>;
```

Job pending in execution cluster

Job is forwarded to the specified execution cluster, but the job is not forced to run. After the job is forwarded, the execution cluster schedules the job according to local policies.

For example:

```
brun -m cluster2 244
Job <244> is being forced to run or forwarded.
```

```
bhist -l 244

Job <244>, User <user1>, Project <default>, Command <sleep 10000>
Mon Jan  3 12:15:22: Submitted from host <hostD>, to Queue <normal>, CWD
                     <$HOME/envs>, Requested Resources <type == any>;
Mon Jan  3 12:19:18: Job is forced to run or forwarded by user or administrator
                     <user1>;
Mon Jan  3 12:19:18: Forwarded job to cluster cluster2;
Mon Jan  3 12:19:18: Remote job control initiated;
Mon Jan  3 12:19:18: Dispatched to <hostB>;
Mon Jan  3 12:19:18: Remote job control completed;
Mon Jan  3 12:19:19: Starting (Pid 28804);
Mon Jan  3 12:19:19: Running with execution home </home/user1>, Execution CWD
                     </home/user1/envs>, Execution Pid <28804>;
```

bsub

Submits job to LSF AE and forwards to execution clusters.

-m

Enhanced option allows you to specify local hosts, remote hosts, and execution clusters.

```
bsub -m ["[host_name|hostgroup][@cluster_name][[!]|+pref_level]]|cluster_name[+[pref_level]]..."
```

- Cluster preference set at the job level (**bsub -m**) overrides cluster preference set at the queue level (**SNDJOBS_TO**).
- Local hosts are always used before remote hosts.
- Host names without a cluster specified must be unique.
- When both clusters and hosts are specified, the host list is merged and filtered after the job is forwarded to an execution cluster.
- The keyword others only applies to local hosts. To specify other remote hosts in a cluster, use the cluster name.

  For example, `bsub -m "hostA@cluster1 cluster1"` refers to hostA and others in cluster1 without using the keyword others for remote hosts.

Examples:

```
bsub -m cluster2
```

The submission cluster forwards the job to cluster2.

```
bsub -m hostB@cluster2
```

The submission cluster forwards the job to cluster2 to run on hostB.

```
bsub -m "cluster3+1 hostB@cluster2+2"
```

The job is forwarded to cluster2 to run on hostB if possible. If not, the job is forwarded to cluster3 to run on any host in that cluster.

```
bsub -m "local_host rmt_host@cluster1"
```

The job runs on local_host, if possible. If not, the job is forwarded to cluster1 to run on rmt_host.

```
bsub -m "rmt_host"
```

The job is forwarded to the cluster containing rmt_host; rmt_host must be a unique host name.

```
bsub -m "cluster2 hostE@cluster2"
```

The job is forwarded to cluster2. Since the entire cluster is specified with the same preference as hostE, no host preference applies within cluster2.

```
bsub -m "cluster3 cluster4 hostD@cluster3+1"
```

The job is forwarded to cluster3 if possible. Since both cluster and hosts within the cluster are specified, host preference is filtered (cluster3 hosts only) and merged to become `bsub -m "others hostD+1"` on cluster3.

If the job cannot be forwarded to cluster3, the submission cluster forwards the job to cluster4.

bswitch

Switches job order. Forwarded jobs can only be switched within one LSF AE execution cluster. Changes across execution clusters are not supported.

Run **bswitch** commands through the submission cluster, not through individual execution clusters.

btop

Changes job order. Forwarded job can only be moved within one LSF AE execution cluster. Changes across execution clusters are not supported.

Run **btop** commands through the submission cluster, not through individual execution clusters.

bstop/bresume/bkill

Controls job status throughout LSF AE.

Run these commands through the submission cluster. These commands are not supported on individual execution clusters.

busers

Displays all running and pending jobs counters for users in the local cluster.

Run **busers** queries through the submission cluster to see information for the complete LSF AE installation.

LSF AE requires the same user and user group definitions across all clusters.

LSF AE parameters

Some parameters are new or have added functionality in LSF AE

lsb.params parameters

EVALUATE_JOB_DEPENDENCY

Syntax

```
EVALUATE_JOB_DEPENDENCY=integer
```

Description

Set the maximum number of job dependencies `mbatchd` evaluates in one scheduling cycle. This parameter optimizes job dependency evaluation logic, which saves time on the job dependence evaluation process. Jobs dependence evaluation is an evaluation process which is used to check if each job's dependence condition is satisfied. When a job's dependence condition is satisfied, it sets a ready flag and allows itself to be scheduled by `mbschd`.

When **EVALUATE_JOB_DEPENDENCY** is set, a configured number of jobs are evaluated. Not all the dependency satisfied jobs may be set to READY status in the same session. Therefore, jobs intended to be dispatched in one scheduling session may be dispatched in different scheduling sessions.

Also, the job dependency evaluation process starts from the last evaluation end location, so it may prevent some dependency satisfied jobs that occur before the end location from being set to READY status in that particular session. This may cause one job to be dispatched before another when the other was ready first.

Starting a scheduling session triggers LSF to do job dependency evaluation. The number of jobs evaluated corresponds to the configuration and the endpoint is kept. LSF starts the job dependency evaluation from the endpoint in the next session. LSF evaluates all dependent jobs every 10 minutes regardless of the configuration for **EVALUATE_JOB_DEPENDENCY**.

Default

50,000 dependent jobs per schedule cycle.

FWD_PEND_JOB_FACTOR

Syntax

FWD_PEND_JOB_FACTOR=*number*

Description

Used in fairshare scheduling. Forwarded pending jobs weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of jobs already forwarded and pending in execution clusters.

Defined in the submission cluster only.

Default

RUN_JOB_FACTOR

MBD_REFRESH_TIME

Syntax

MBD_REFRESH_TIME=*seconds* [*min_refresh_time*]

where *min_refresh_time* defines the minimum time (in seconds) that the child **mbatchd** will stay to handle queries. The valid range is 0 - 300.

Description

Time interval, in seconds, when **mbatchd** will fork a new child **mbatchd** to service query requests to keep information sent back to clients updated. A child **mbatchd** processes query requests creating threads.

MBD_REFRESH_TIME applies only to UNIX platforms that support thread programming.

To enable **MBD_REFRESH_TIME** you must specify **LSB_QUERY_PORT** in `lsf.conf`. The child **mbatchd** listens to the port number specified by **LSB_QUERY_PORT** and creates threads to service requests until the job changes status, a new job is submitted, or **MBD_REFRESH_TIME** has expired.

- If **MBD_REFRESH_TIME** is < *min_refresh_time*, the child **mbatchd** exits at **MBD_REFRESH_TIME** even if the job changes status or a new job is submitted before **MBD_REFRESH_TIME** expires.
- If **MBD_REFRESH_TIME** > *min_refresh_time*:
  - the child **mbatchd** exits at *min_refresh_time* if a job changes status or a new job is submitted before the *min_refresh_time*
  - the child **mbatchd** exits after the *min_refresh_time* when a job changes status or a new job is submitted
- If **MBD_REFRESH_TIME** > *min_refresh_time* and no job changes status or a new job is submitted, the child **mbatchd** exits at **MBD_REFRESH_TIME**

If you extend multithreaded query support to batch query requests (by specifying `LSB_QUERY_ENH=Y` in `lsf.conf`), the child **mbatchd** will also exit if any of the following commands are run in the cluster:

- **bconf**
- **badmin reconfig**
- **badmin** commands to change a queue's status (**badmin qopen**, **badmin qclose**, **badmin qact**, and **badmin qinact**)
- **badmin** commands to change a host's status (**badmin hopen** and **badmin hclose**)
- **badmin perfmon start**

The value of this parameter must be between 0 and 300. Any values specified out of this range are ignored, and the system default value is applied.

The **bjobs** command may not display up-to-date information if two consecutive query commands are issued before a child **mbatchd** expires because child **mbatchd** job information is not updated. If you use the **bjobs** command and do not get up-to-date information, you may need to decrease the value of this parameter. Note, however, that the lower the value of this parameter, the more you negatively affect performance.

The number of concurrent requests is limited by the number of concurrent threads that a process can have. This number varies by platform:

- Sun Solaris, 2500 threads per process
- AIX, 512 threads per process
- Digital, 256 threads per process
- HP-UX, 64 threads per process

Valid values

0 - 300 seconds

Default

MBD_REFRESH_TIME=5 seconds if *min_refresh_time* is not defined or if MBD_REFRESH_TIME is set to less than 5

MBD_REFRESH_TIME=300 seconds if the defined value is more than 300

The *min_refresh_time* default value depends on the number of jobs in the cluster. When `LSB_QUERY_ENH=Y` (`lsf.conf`) the *min_refresh_time* default value is given by:

- up to 500k jobs, *min_refresh_time*=10 seconds
- more than 500k jobs, *min_refresh_time*=(10 + (#jobs-500k)/100k) seconds

See also

`LSB_QUERY_PORT`, **`LSB_QUERY_ENH`** (`lsf.conf`)

PREEMPTABLE_RESOURCES

Syntax

```
PREEMPTABLE_RESOURCES= res1 [res2] [res3] ....
```

Description

Enables preemption for resources (in addition to slots) when preemptive scheduling is enabled (has no effect if queue preemption is not enabled) and specifies the resources that will be preemptable. Specify shared resources (static or dynamic) that are numeric, decreasing, and releasable. One of the resources can be built-in resource mem, meaning that `res1` is also option if memory comes later in the list.

This parameter only applies to execution clusters and are ignored in submission clusters.

The default preemption action is the suspend the job. To force a job to release resources instead of suspending them, set `TERMINATE_WHEN=PREEMPT` in `lsb.queues`, or set **`JOB_CONTROLS`** in `lsb.queues` and specify brequeue as the SUSPEND action. Some applications will release resources when sent the SIGTSTP signal. Use **`JOB_CONTROLS`** to send this signal to suspend the job.

To enable memory preemption, include `mem` in the **`PREEMPTABLE_RESOURCES`** list in `lsb.params`.

When preempting a job for memory, LSF does not free the memory occupied by the job. Rather, it suspends the job and dispatches another job to the host. It relies on the operating system to swap out the pages of the stopped job as memory of the running job grows.

Default

Not defined (if preemptive scheduling is configured, LSF preempts on job slots only)

SCHEDULER_THREADS

Syntax

SCHEDULER_THREADS=*number*

Description

On multi-core machines you can speed up scheduling by running multiple threads for the evaluation of job resource requirements, speeding up the entire scheduling cycle. The recommended number of threads is the same as the number of cores assigned to the scheduler in the parameter **`LSF_DAEMONS_CPUS`** (`lsf.conf`).

Setting **SCHEDULER_THREADS** to a non-zero value can result in increased memory use by the scheduling daemon **mbschd**.

Default

0 (a single thread is used)

XL_RES_RATIO

Syntax

```
XL_RES_RATIO=default [cluster_value/cluster_name...]
```

Description

To ensure the execution clusters always have enough pending workload to fill local idle slots, the submission cluster forwards jobs based on the number of forward slots given by:

forward slots = (available slots in execution cluster)***XL_RES_RATIO**

The exact value of `forward slots` set by **XL_RES_RATIO** acts as a guideline instead of a strict threshold. Although the submission cluster follows the `forward slots` value when forwarding jobs, the actual number of forwarded jobs in each execution cluster reflects local scheduling policies, resource availability, and re-forwarded jobs. The submission cluster compensates by forwarding more or less jobs in the next forwarding cycle, as required.

When working with License Scheduler (in License Scheduler cluster mode and fast dispatch project mode only), **XL_RES_RATIO** also takes effect for License Scheduler resources in execution clusters. The submission cluster will forward jobs that require license tokens to execution clusters, which allows these license tokens to be reused when the license tokens are free.

Defined in the submission cluster only.

Default

1.5

Examples

```
XL_RES_RATIO=2
```

The value 2 applies to all execution clusters

```
XL_RES_RATIO=1.5 2/cluster2 2.5/cluster3
```

The value 2 applies to cluster2, the value 2.5 applies to cluster3, and the default 1.5 applies to all other execution clusters.

XL_RES_UPDATE_INTERVAL

Syntax

```
XL_RES_UPDATE_INTERVAL=seconds
```

Description

Sets how often the cluster sends updated resource information to the submission cluster.

Defined for execution clusters only.

Default

`MBD_SLEEP_TIME`

lsb.queues parameters

ABS_CLUSTER_PREFERENCE

Syntax

`ABS_CLUSTER_PREFERENCE=Y|N`

Description

When enabled, forwards jobs from the submission cluster in order of cluster preference as set in **`SNDJOBS_TO`** (`lsb.queues`).

When disabled, forwards jobs from the submission cluster based on cluster preference, available slots, and pending slots.

Defined in the submission cluster only.

See also **`SNDJOBS_TO`** in `lsb.queues`.

Default

N

DISPATCH_BY_QUEUE

Syntax

`DISPATCH_BY_QUEUE=Y|N`

Description

When enabled, broadcasts scheduling decisions queue by queue instead of waiting for the entire scheduling cycle to complete.

To minimize dispatch time, use with `NEW_JOB_SCHED_DELAY=0` also defined in `lsb.queues`.

Defined in the submission cluster only.

Default

N

FWD_PEND_JOB_FACTOR

Syntax

```
FWD_PEND_JOB_FACTOR=number
```

Description

Used in fairshare scheduling. Forwarded pending jobs weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of jobs already forwarded and pending in execution clusters.

Defined in the submission cluster only.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined

MAX_RSCHED_TIME

Syntax

```
MAX_RSCHED_TIME=timeout [wait_time]
```

Description

Defined in the submission cluster only.

MAX_RSCHED_TIME sets the cluster reselection polices for LSF AE

timeout***MBD_SLEEP_TIME** determines how long a job stays pending in the execution cluster before returning to the submission cluster.

wait_time***MBD_SLEEP_TIME** determines how long the submission cluster waits for other execution clusters to become available before returning to the same execution cluster. The wait time only applies when there are execution clusters the job has not yet tried.

Default

360 10

SNDJOBS_TO

Syntax

```
SNDJOBS_TO=[queue@]cluster_name[+preference]...
```

Description

Sets the execution cluster queues jobs are forwarded to, and the preference level of each execution cluster queue. The queue names in execution clusters must match the queue names in the submission cluster.

Defined in the submission cluster only.

See also **ABS_CLUSTER_PREFERENCE** in `lsb.queues`.

Default

Not defined.

Example

```
SNDJOBS_TO=cluster1+2 cluster2 cluster3
```

SUCCESS_EXIT_VALUES

Syntax

`SUCCESS_EXIT_VALUES`=[exit_code ...]

Description

Use this parameter to specify exit values used byLSF to determine if the job was done successfully. Application-level success exit values defined with **SUCCESS_EXIT_VALUES** in `lsb.applications` override the configuration defined in `lsb.queues`. Job-level success exit values specified with the **LSB_SUCCESS_EXIT_VALUES** environment variable override the configuration in `lsb.queues` and `lsb.applications`.

Job-level success exit values are defined in the submission cluster, while application- or queue-level success exit values are defined in the execution cluster.

Use **SUCCESS_EXIT_VALUES** for submitting jobs to specific queues that successfully exit with non-zero values so that LSF does not interpret non-zero exit codes as job failure.

In MultiCluster job forwarding mode, LSF uses the **SUCCESS_EXIT_VALUES** from the remote cluster.

In a MultiCluster resource leasing environment, LSF uses the **SUCCESS_EXIT_VALUES** from the consumer cluster.

`exit_code` should be a value between 0 and 255. Use spaces to separate multiple exit codes.

Default

Not defined.

lsb.resources

| Resource definitions in the `lsb.resources` file only apply to execution clusters and
| are ignored in submission clusters.

Limit section

FWD_SLOTS:

Syntax

`FWD_SLOTS=`*integer*

`FWD_SLOTS`

`- | `*integer*

Description

Maximum number of job slots that resource consumers can forward to clusters. Specify a positive integer greater than or equal 0.

`FWD_SLOTS` limits only apply to the resource consumers:
- **`PROJECTS`** or **`PER_PROJECT`**
- **`QUEUES`** or **`PER_QUEUE`**
- **`USERS`** or **`PER_USER`**
- **`CLUSTERS`** or **`PER_CLUSTER`**

Combining **`FWD_SLOTS`** with other resource consumers results in a warning message, and the limit is ignored.

Use this parameter to prevent a host from being overloaded with too many forwarded jobs.

In horizontal format, use only one FWD_SLOTS line per Limit section.

In vertical format, use empty parentheses `()` or a dash (`-`) to indicate the default value (no limit). Fields cannot be left blank.

Default

No limit

Example

```
FWD_SLOTS=20
```

PER_CLUSTER:
Syntax

`PER_CLUSTER=all | `*cluster_name* ...

`PER_CLUSTER`

`( [-] | all | `*cluster_name* ... `)`

Description

A space-separated list of execution cluster names on which limits are enforced. Limits are enforced on each execution cluster listed.

Do not configure PER_CLUSTER and CLUSTERS limits in the same Limit section.

In horizontal format, use only one PER_CLUSTER line per Limit section.

Use the keyword **`all`** to configure limits that apply to all execution clusters configured in LSF AE.

In vertical tabular format, multiple execution cluster names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for PER_CLUSTER or CLUSTERS, no limit is enforced on any execution cluster.

Example

```
PER_CLUSTER=cluster1 cluster2
```

CLUSTERS:
Syntax

`CLUSTERS=all |` *cluster_name* ...

`CLUSTERS`

`( [-] | all |` *cluster_name* ... `)`

Description

A space-separated list of execution cluster names on which limits are enforced. Limits are enforced on all execution clusters listed.

To specify a per-cluster limit, use the PER_CLUSTER keyword. Do not configure CLUSTERS and PER_CLUSTER limits in the same Limit section.

In horizontal format, use only one CLUSTERS line per Limit section.

Use the keyword `all` to configure limits that apply to all execution clusters configured in LSF AE.

In vertical tabular format, multiple execution cluster names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate an empty field. Fields cannot be left blank.

Default

`all` (limits are enforced on all execution clusters)

Example

```
CLUSTERS=clusterA clusterB
```

lsf.cluster file

RemoteClusters section

Optional. By default, the local cluster can obtain information about all other clusters specified in `lsf.shared`. If used, make sure the list includes all clusters specified in `lsf.shared`, even if you only configure the default behavior for some of the clusters.

The RemoteClusters section is required if you want to configure cluster equivalency, cache interval, daemon authentication across clusters, or if you want to run parallel jobs across clusters.

`CLUSTERNAME` is mandatory and other parameters are optional.

Example

```
Begin RemoteClusters
CLUSTERNAME  EQUIV   CACHE_INTERVAL  RECV_FROM  AUTH
cluster1       Y         60            Y      -
cluster2       Y         60            Y      -
cluster4       Y         60            N      -
End RemoteClusters
```

CLUSTERNAME:
Description

Remote cluster name. Include all clusters.

Any clusters listed in `lsf.shared` but not listed here will be ignored by this cluster.

EQUIV:
Description

Specify ‘Y’ to make the remote cluster equivalent to the local cluster.

Only ’Y’ is supported by LSF AE.

CACHE_INTERVAL:
Description

Specify the load information cache threshold, in seconds. The host information threshold is twice the value of the load information threshold.

To reduce overhead and avoid updating information from remote clusters unnecessarily, LSF displays information in the cache, unless the information in the cache is older than the threshold value.

Default

60 (seconds)

RECV_FROM:
Description

Specifies whether the local cluster accepts parallel jobs that originate in a remote cluster

RECV_FROM does not affect regular or interactive batch jobs.

Specify 'Y' if you want to run parallel jobs across clusters. Otherwise, specify 'N'.

Default

Y

AUTH:
Description

Defines the preferred authentication method for LSF daemons communicating across clusters. Specify the same method name that is used to identify the corresponding **`eauth`** program (eauth.*method_name*). If the remote cluster does not prefer the same method, LSF uses default security between the two clusters.

Default

- (only privileged port (setuid) authentication is used between clusters)

lsf.conf parameters

LSB_FS_COMM_ENH

Syntax

`LSB_FS_COMM_ENH=Y`

Description

Set `LSB_FS_COMM_ENH=Y` in the `lsf.conf` file to enable the fairshare performance enhancement merged from LSF 8.0.

Default

Undefined (N).

LSB_JOBINFO_DIR

Syntax

`LSB_JOBINFO_DIR=`*`directory`*

Description

Directory in which job information files are kept.

The **`LSB_JOBINFO_DIR`** directory must be owned by the primary LSF administrator, and be accessible with read, write, and execute access from all potential master hosts in all clusters.

Optional. Defined for the submission cluster only.

Default

`$LSB_SHAREDIR/`*`cluster_name`*`/logdir/info`

LSB_NUM_NIOS_CALLBACK_THREADS

Syntax

`LSB_NUM_NIOS_CALLBACK_THREADS=`*`integer`*

Description

Specifies the number of callback threads to use for batch queries.If your cluster runs a large amount of blocking mode (**`bsub -K`**), response to batch queries can become very slow. If you run large number of **`bsub -K`** jobs, you can define the threads to the number of processors on the master host.

Default

4 for LSF AE

LSB_QUERY_ENH

Syntax

`LSB_QUERY_ENH=Y|N`

Description

Enables multithreaded query support for batch query requests. In addition, the **`mbatchd`** system query monitoring mechanism starts automatically instead of being triggered by a query request. This ensures a consistent query response time within the system.

Enables a new default setting for *min_refresh_time* in **`MBD_REFRESH_TIME`** (`lsb.params`).

Default

N (multithreaded query support for **`bjobs`** query requests only)

LSB_XL_MAX_FORWARD_PER_SESSION

Syntax

`LSB_XL_MAX_FORWARD_PER_SESSION=`*`integer`*

Description

Sets the maximum number of jobs forwarded within a scheduling session.

Defined in the submission cluster only.

Default

5000 for LSF AE.

LSF_STRICT_RESREQ

Syntax

`LSF_STRICT_RESREQ=Y | N`

Description

When `LSF_STRICT_RESREQ=Y`, the resource requirement selection string must conform to the stricter resource requirement syntax described in *Administering Platform LSF*. The strict resource requirement syntax only applies to the `select` section. It does not apply to the other resource requirement sections (`order`, `rusage`, `same`, `span`, or `cu`).

When `LSF_STRICT_RESREQ=Y` in `lsf.conf`, LSF rejects resource requirement strings where an `rusage` section contains a non-consumable resource.

When `LSF_STRICT_RESREQ=N`, the default resource requirement selection string evaluation is performed.

Default

N when **`LSF_XL`** is defined.

LSF_XL

Syntax

`LSF_XL=`*`clustername`*

Description

Identifies the submission cluster within the LSF AE installation. This is required for all clusters in LSF AE.

Default

Not defined.

Example

```
LSF_XL=sub_cluster
```

Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

LSF®, Platform, and Platform Computing are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software

Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Printed in USA

SC27-5321-03