Platform LSF
Version 9 Release 1.1

Release Notes

GI13-3413-01

Platform LSF
Version 9 Release 1.1

Release Notes

GI13-3413-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 45.

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.

© Copyright IBM Corporation 1992, 2013.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Release Notes for Platform LSF

Version: 9.1.1

Release date: March 2013

Last modified: March 8, 2013

Support: www.ibm.com/support

Upgrade and Compatibility Notes

For additional information about Platform LSF 9.1.1, visit the IBM Platform Computing web page:

www-03.ibm.com/systems/technicalcomputing/platformcomputing/index.html

Master host selection

To achieve the highest degree of performance and scalability, we strongly recommend that you use a powerful master host.

There is no minimum CPU requirement. For the platforms LSF is supported on, any host with sufficient physical memory can run LSF as master host. Swap space is normally configured as twice the physical memory. LSF daemons use about 20 MB of memory when no jobs are running. Active jobs consume most of the memory LSF requires.

Cluster size	Active jobs	Minimum Recommended Memory	Recommended server CPU (Intel, AMD, or equivalent)
Small (<100 hosts)	1,000	1 GB	any server CPU
	10,000	2 GB	recent server CPU
Medium (100-1000 hosts)	10,000	4 GB	multi-core CPU (2 cores)
	50,000	8 GB	multi-core CPU (4 cores)
Large (>1000 hosts)	50,000	16 GB	multi-core CPU (4 cores)
	500,000	32 GB	multi-core CPU (8 cores)

Server host compatibility

Important: To use new features introduced in Platform LSF 9.1.1, you *must* upgrade all hosts in your cluster to Platform LSF 9.1.1.

Platform LSF Version 9.1.1 is fully compatible with Platform LSF Versions 8.3 and 9.1.

LSF 6.2, 7.*x*, 8.0, 8.0.1, 8.3 and 9.1 servers are compatible with Platform LSF 9.1.1 master hosts. All LSF 6.2, 7.*x*, 8.0, 8.0.1, 8.3 and 9.1 features are supported by Platform LSF 9.1.1 master hosts.

© Copyright IBM Corp. 1992, 2013

LSF Family product compatibility

Platform RTM

Customers can use Platform RTM 8.3 to collect data from a Platform LSF 9.1.1 cluster. When adding the cluster, select 'Poller for LSF 8'.

Platform License Scheduler

Platform License Scheduler 8.3 is compatible with Platform LSF 9.1.1.

Platform Analytics

Platform Analytics 8.3 is compatible with Platform LSF 9.1.1 after the following manual configuration:

To have PA 8.3 collect data from Platform LSF 9.1.1 Cluster:

1. Set the following parameters in `lsb.params`:
   - **`ENABLE_EVENT_STREAM=`**Y
   - **`ALLOW_EVENT_TYPE=`**"JOB_NEW JOB_FINISH2 JOB_STARTLIMIT JOB_STATUS2 JOB_PENDING_REASONS"
   - **`RUNTIME_LOG_INTERVAL=`**10
2. copy `elim.coreutil` to LSF:

   ```
   cp ANALYTICS_TOP/elim/os_type/elim.coreutil $LSF_SERVERDIR
   ```
3. In `lsf.shared`, create the following:

   ```
   Begin Resource
   RESOURCENAME   TYPE        INTERVAL INCREASING DESCRIPTION
   CORE_UTIL      String           300         () (Core Utilization)
   End Resource
   ```
4. In `lsf.cluster.cluster_name`, create the following:

   ```
   Begin ResourceMap
   RESOURCENAME  LOCATION
   CORE_UTIL     [default]
   End ResourceMap
   ```
5. Restart all LSF daemons.
6. Configure user group and host group.
7. Run **`lsid`** and check the output.
8. Install Platform Analytics 8.3 with **`COLLECTED_DATA_TYPE=`**LSF.
9. Check `perf.conf` to see LSF_VERSION.
10. Start the Platform Loader Controller.
11. Create the symbolic link:

    ```
    ln -s /opt/user1/pa83_node_lsf911/lsf/8.0 9.1
    ```
12. Restart the Platform Loader Controller.
13. Check the log files and table data to make sure there are no errors.
14. Change all the LSF related data loader intervals to 120 seconds, and run for one day. Check the Platform Loader Controller and data loader log files to make sure there are no errors.

Platform Application Center

Platform Application Center (PAC) 8.3 is compatible with Platform LSF 9.1.1 after the following manual configuration.

If using PAC 8.3 with LSF 9.1.1, *$PAC_TOP*/perf/lsf/8.3 must be renamed to *$PAC_TOP*/perf/lsf/9.1

For example:

```
mv /opt/pac/perf/lsf/8.3 /opt/pac/perf/lsf/9.1
```

Default LSF behavior

With no new features enabled in a newly upgraded LSF 9.1.1 cluster, the following pre-9.1 functionality has changed:

- By default, **bjobs -l** reports individual host rusage for a parallel job. Set **LSF_HPC_EXTENSIONS=NO_HOST_RUSAGE** to enable pre-LSF 9.1 behavior.
- **bswitch** does not modify the effective resource requirement of a running job based on the resource requirements of the destination queue res req definition. You can define **BSWITCH_MODIFY_RUSAGE** to enable pre-LSF 9.1 behavior.
- LSF now uses non-privileged ports by default for daemon communication. You can set **LSF_NON_PRIVILEGED_PORTS=N** in lsf.conf to enable privileged port communication. Also, **LSF_MC_NON_PRIVILEGED_PORTS** and **LSF_NON_PRIVILEGED_PORTS** are now fully decoupled, which is different from previous versions.

If you are upgrading your master host, and leaving some server hosts still running older versions, then you should do the following:

- If LSF_NON_PRIVILEGED_PORTS is already set to Y or N, continue with upgrade.
- If LSF_NON_PRIVILEGED_PORTS is not set, but **LSB_MAX_JOB_DISPATCH_PER_SESSION** is set to a value greater than 300 do the following:
  1. Shut down the cluster
  2. Set LSF_NON_PRIVILEGED_PORTS=Y
  3. Upgrade the master host
  4. Restart the cluster
- If neither LSF_NON_PRIVILEGED_PORTS nor LSB_MAX_JOB_DISPATCH_PER_SESSION is set, do the following:
  1. Set LSF_NON_PRIVILEGED_PORTS=N.
  2. Upgrade the master host.
  3. Start LSF on the master host.

See *Upgrading IBM Platform LSF on UNIX and Linux* for detailed upgrade steps.

Upgrade from an earlier version of Platform LSF on UNIX and Linux

Follow the steps in *Upgrading IBM Platform LSF on UNIX and Linux* (lsf_upgrade_unix.pdf) to run **lsfinstall** to *upgrade* LSF:

- Upgrade a pre-LSF Version 7 UNIX or Linux cluster to Platform LSF 9.1.1
- Upgrade an LSF Version 7 Update 2 or higher to Platform LSF 9.1.1

Important: *DO NOT* use the UNIX and Linux upgrade steps to migrate an existing LSF Version 7 or LSF 7 Update 1 cluster to LSF 9.1.1. Follow the manual steps in the document *Migrating to Platform LSF Version 9.1.1 on UNIX and Linux* to migrate an existing LSF Version 7 or LSF 7 Update 1 cluster to LSF 9.1.1 on UNIX and Linux.

Migrate LSF Version 7 and Version 7 Update 1 cluster to LSF 9.1.1 on UNIX and Linux

Follow the steps in *Migrating to Platform LSF Version 9.1.1 on UNIX and Linux* (`lsf_migrate_unix.pdf`) to migrate an *existing* LSF 7 or LSF 7 Update 1 cluster:

- Migrate an existing LSF Version 7 cluster to LSF 9.1.1 on UNIX and Linux
- Migrate an existing LSF Version 7 Update 1 cluster to LSF 9.1.1 on UNIX and Linux

Note:

DO NOT use these steps to migrate an existing LSF 7 Update 2 or higher cluster to LSF 9.1.1. Follow the steps in *Upgrading Platform LSF on UNIX and Linux* to upgrade LSF.

Migrate LSF Version 7 or higher cluster to LSF 9.1.1 cluster on Windows

To migrate an *existing* LSF 7 Windows cluster to Platform LSF 9.1.1 on Windows, follow the steps in *Migrating Platform LSF Version 7 to Platform LSF 9.1.1 on Windows* (`lsf_migrate_windows.pdf`).

Note:

DO NOT use these steps to migrate a pre-version 7 cluster to LSF 9.1.1. Pre-version 7 clusters must first be migrated to LSF Version 7.

Bug fixes and solutions in this release

At release, LSF 9.1.1 includes all bug fixes and solutions delivered after December 31, 2012.

System requirements

Operating system support:

- AIX 6 and 7 on POWER
- HP UX B.11.31 on PA-RISC
- HP UX B.11.31 on IA64
- Solaris 10 and 11 on Sparc
- Solaris 10 and 11 on x86-64
- Linux on x86-64 Kernel 2.6 and 3.*x*
- Linux on POWER Kernel 2.6 and 3.*x*
- Windows 2003/2008/2012/XP/7/8 32-bit and 64-bit
- Mac OS 10.*x*
- Cray Linux 4.0 or later and BASIL 1.2 protocol

For detailed LSF system support information, see:

http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/lsf/index.html

API compatibility

To take full advantage of new Platform LSF 9.1.1 features, you should recompile your existing Platform LSF applications with Platform LSF 9.1.1.

Applications need to be rebuilt if they use APIs that have changed in Platform LSF 9.1.1.

New and changed Platform LSF APIs

For new APIs created for LSF 9.1.1, refer to the *IBM Platform LSF 9.1.1 API Reference*.

Platform LSF Editions

LSF Advanced Edition

New in LSF 9.1, Platform LSF Advanced Edition is architected to support extreme scalability and throughput. LSF Advanced Edition may provide greater than three times more scalability than earlier versions of LSF, enabling you to consolidate your compute resources to achieve maximum flexibility and utilization. LSF Advanced Edition has been tested on configurations up to 18,000 nodes and 160,000 cores running high-throughput workloads of 160,000 concurrent short jobs with 2,000,000 pending jobs. These are not hard scalability or performance limits.

LSF Advanced Edition includes advanced features like LSF/XL, multithreaded batch query and multithreaded resource requirement matching. Platform Session Scheduler is also included with LSF Advanced Edition.

LSF/XL feature

The LSF/XL feature of LSF Advanced Edition is a new architecture to address long-term scalability and performance demands for extreme workloads. It is based on Platform MultiCluster technology, but LSF/XL is not a MultiCluster mode. It is designed for a single data center.

LSF Standard Edition

LSF Standard Edition is designed for grid environments and optimized for complex workloads and user grouping structures. LSF Standard contains full functionality of LSF including functionality for Platform MultiCluster, floating clients and Platform LSF Make. Platform Session Scheduler is available as an add-on component. There are no performance or scalability restrictions.

LSF Standard Edition is subject to the following usage constraints:

- LSF Standard Edition has been tested on clusters up to 6,000 nodes and 60,000 cores running high-throughput workloads of 60,000 concurrent short jobs with 250,000 pending jobs. These are not hard scalability or performance limits. Higher node or core counts can be achieved with a lower volume of jobs such as parallel HPC workloads where cluster sizes of 75,000 to 100,000 cores are supported. Higher core counts are achievable with additional tuning and configuration.

- For high-throughput workloads, the overall system performance depends on the processing power, I/O capacity, and memory of the scheduling node. For very large clusters, you should seek configuration assistance from IBM.

LSF Express Edition (Linux only)

LSF Express Edition is a solution for Linux customers with very simple scheduling requirements and simple fairshare setup. Smaller clusters typically have a mix of sequential and parallel work as opposed to huge volumes of jobs. For this reason, several performance enhancements and complex scheduling policies designed for large-scale clusters are not applicable to LSF Express Edition clusters. Platform Session Scheduler is available as an add-on component.

Out of box configuration for LSF Express Edition is optimized for smaller clusters with basic scheduling requirements. An LSF Express cluster can have a maximum of 200 server hosts and 200 static client hosts.

LSF Express Edition is subject to the following restrictions:

- LSF Express Edition is supported only on x86_64 Linux.
- LSF Express Edition, Platform LSF Standard Edition, or Platform LSF Advanced Edition cannot coexist in the same cluster.
- Master candidate hosts defined in the **LSF_MASTER_LIST** in `lsf.conf` must exist within the first 200 server hosts in the configuration file. Additional hosts over the limit will not be loaded.

The following LSF Standard Edition features are supported when you upgrade from LSF Express Edition to LSF Standard Edition:

- Job groups
- Live reconfiguration
- Delegation of administrator rights
- Resizable jobs
- Chunk jobs
- Floating clients
- LSF Make
- Resource preemption
- Cross-queue user-based fairshare
- Goal-oriented SLA-driven scheduling
- Guaranteed resource pools
- Slot-based scheduling
- Fairshare scheduling (queue and user fairshare are enabled with a fairshare tree depth of 2 levels in Express and more than two levels in Standard)
- Parallel job scheduling (PAM/PJL is supported. **blaunch** is supported with IBM Platform MPI by default)
- Parallel **mbatchd** restart (**badmin mbdrestart -p**)
- Rapid detection of host failure
- Fast dispatching
- Alternative resource requirement
- **bjobs** shows all levels of resource requirement
- Interaction of select[] and rusage[]
- Process tracking/short jobs

- Platform MultiCluster features
- Multithreaded batch query
- LSF integration with IBM Parallel Environment Runtime Edition

Platform product support with LSF Express Edition

The following IBM Platform products are supported in LSF Express Edition:

- IBM Platform Report, Track, Monitor (RTM)
- IBM Platform Application Center
- IBM Platform License Scheduler

The following IBM Platform products are *not* supported in LSF Express Edition:

- IBM Platform Analytics
- IBM Platform Process Manager

Default configuration

The following table lists the configuration enforced in LSF Express Edition:

Parameter	Setting	Description
RESIZABLE_JOBS in `lsb.applications`	N	If enabled, all jobs belonging to the application will be auto resizable.
EXIT_RATE in `lsb.hosts`	Not defined	Specifies a threshold for exited jobs.
BJOBS_RES_REQ_DISPLAY in `lsb.params`	None	Controls how many levels of resource requirements bjobs –l will display.
CONDENSE_PENDING_REASONS in `lsb.params`	N	Condenses all host-based pending reasons into one generic pending reason.
DEFAULT_JOBGROUP in `lsb.params`	Disabled	The name of the default job group.
EADMIN_TRIGGER_DURATION in `lsb.params`	1 minute	Defines how often LSF_SERVERDIR/eadmin is invoked once a job exception is detected. Used in conjunction with job exception handling parameters JOB_IDLE, JOB_OVERRUN, and JOB_UNDERRUN in lsb.queues.
ENABLE_DEFAULT_EGO_SLA in `lsb.params`	Not defined	The name of the default service class or EGO consumer name for EGO-enabled SLA scheduling.
EVALUATE_JOB_DEPENDENCY in `lsb.params`	Unlimited	Sets the maximum number of job dependencies mbatchd evaluates in one scheduling cycle.
GLOBAL_EXIT_RATE in `lsb.params`	2147483647	Specifies a cluster-wide threshold for exited jobs
JOB_POSITION_CONTROL_BY_ADMIN in `lsb.params`	Disabled	Allows LSF administrators to control whether users can use **`btop`** and **`bbot`** to move jobs to the top and bottom of queues.

Parameter	Setting	Description
LSB_SYNC_HOST_STAT_FROM_LIM in `lsb.params`	N	Improves the speed with which **mbatchd** obtains host status, and therefore the speed with which LSF reschedules rerunnable jobs. This parameter is most useful for a large clusters, so it is disabled for LSF Express Edition.
MAX_CONCURRENT_QUERY in `lsb.params`	100	Controls the maximum number of concurrent query commands.
MAX_INFO_DIRS in `lsb.params`	Disabled	The number of subdirectories under the `LSB_SHAREDIR/`*`cluster_name`*`/logdir/info` directory.
MAX_JOBID in `lsb.params`	999999	The job ID limit. The job ID limit is the highest job ID that LSF will ever assign, and also the maximum number of jobs in the system.
MAX_JOB_NUM in `lsb.params`	1000	The maximum number of finished jobs whose events are to be stored in `lsb.events`.
MIN_SWITCH_PERIOD in `lsb.params`	Disabled	The minimum period in seconds between event log switches.
MBD_QUERY_CPUS in `lsb.params`	Disabled	Specifies the master host CPUs on which **mbatchd** child query processes can run (hard CPU affinity).
NO_PREEMPT_INTERVAL in `lsb.params`	0	Prevents preemption of jobs for the specified number of minutes of uninterrupted run time, where minutes is wall-clock time, not normalized time.
NO_PREEMPT_RUN_TIME in `lsb.params`	-1 (not defined)	Prevents preemption of jobs that have been running for the specified number of minutes or the specified percentage of the estimated run time or run limit.
PREEMPTABLE_RESOURCES in `lsb.params`	Not defined	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.
PREEMPT_FOR in `lsb.params`	0	If preemptive scheduling is enabled, this parameter is used to disregard suspended jobs when determining if a job slot limit is exceeded, to preempt jobs with the shortest running time, and to optimize preemption of parallel jobs.
SCHED_METRIC_ENABLE in `lsb.params`	N	Enables scheduler performance metric collection.
SCHED_METRIC_SAMPLE_PERIOD in `lsb.params`	Disabled	Performance metric sampling period.

Parameter	Setting	Description
SCHEDULER_THREADS in `lsb.params`	0	Sets the number of threads the scheduler uses to evaluate resource requirements.
DISPATCH_BY_QUEUE in `lsb.queues`	N	Increases queue responsiveness. The scheduling decision for the specified queue will be published without waiting for the whole scheduling session to finish. The scheduling decision for the jobs in the specified queue is final and these jobs cannot be preempted within the same scheduling cycle.
LSB_JOBID_DISP_LENGTH in `lsf.conf`	Not defined	By default, LSF commands bjobs and bhist display job IDs with a maximum length of 7 characters. Job IDs greater than 9999999 are truncated on the left. When LSB_JOBID_DISP_LENGTH=10, the width of the JOBID column in bjobs and bhist increases to 10 characters.
LSB_FORK_JOB_REQUEST in `lsf.conf`	N	Improves **mbatchd** response time after **mbatchd** is restarted (including parallel restart) and has finished replaying events.
LSB_MAX_JOB_DISPATCH_PER_SESSION in `lsf.conf`	300	Defines the maximum number of jobs that **mbatchd** can dispatch during one job scheduling session.
LSF_PROCESS_TRACKING in `lsf.conf`	N	Tracks processes based on job control functions such as termination, suspension, resume and other signaling, on Linux systems which support cgroups' freezer subsystem.
LSB_QUERY_ENH in `lsf.conf`	N	Extends multithreaded query support to batch query requests (in addition to **bjobs** 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`).
LSB_QUERY_PORT in `lsf.conf`	Disabled	Increases **mbatchd** performance when using the **bjobs** command on busy clusters with many jobs and frequent query request.
LSF_LINUX_CGROUP_ACCT in `lsf.conf`	N	Tracks processes based on CPU and memory accounting for Linux systems that support cgroup's memory and cpuacct subsystems.

IBM Platform entitlement files

Entitlement files are used for determining which edition of the product is enabled. The following entitlement files are packaged for LSF:

- LSF Standard Edition: `platform_lsf_std_entitlement.dat`
- LSF Express Edition: `platform_lsf_exp_entitlement.dat`.
- LSF Advanced Edition: `platform_lsf_adv_entitlement.dat`.

The entitlement file for the edition you use is installed as `LSF_TOP/conf/lsf.entitlement`.

If you have installed LSF Express, you can upgrade later to LSF Standard Edition or LSF Advanced Edition to take advantage of the additional functionality. Simply reinstall the cluster with the LSF Standard entitlement file (`platform_lsf_std_entitlement.dat`) or the LSF Advanced entitlement file (`platform_lsf_adv_entitlement.dat`).

You can also manually upgrade from LSF Express Edition to Standard Edition or Advanced Edition. Get the LSF Standard or Advanced Edition entitlement file, copy it to `LSF_TOP/conf/lsf.entitlement` and restart you cluster. The new entitlement enables the additional functionality of LSF Standard Edition, but you may need to manually change some of the default LSF Express configuration parameters to use the LSF Standard or Advanced features.

To take advantage of LSF SLA features in LSF Standard Edition, copy `LSF_TOP/LSF_VERSION/install/conf_tmpl/lsf_standard/lsb.serviceclasses` into `LSF_TOP/conf/lsbatch/LSF_CLUSTERNAME/configdir/`.

Once LSF is installed and running, run the **lsid** command to see which edition of LSF is enabled.

What's New in Platform LSF Version 9.1.1

New and changed behavior

Change the global LSF default sorting order

You can change the global LSF system default sorting order of resource requirements so the scheduler can find the right candidate host. This makes it easier to maintain a single global default order instead of having to set a default order in the `lsb.queues` file for every queue defined in the system. You can also specify a default order to replace the default sorting value of `r15s:pg`, which could impact performance in large scale clusters.

Define application-specific environment variables

You can use application profiles to pass application-specific tuning and runtime parameters to the application by defining application-specific environment variables. Once an environment variable is set, it applies for each job that uses the same application profile. This provides a simple way of extending application profiles to include additional information.

Controlling CPU and memory affinity for NUMA hosts

Platform LSF can schedule jobs that are affinity aware. This allows jobs to take advantage of different levels of processing units (NUMA nodes, sockets, cores, and threads).

An *affinity resource requirement* string specifies CPU or memory binding requirements for the tasks of jobs requiring topology-aware scheduling. An `affinity[]` resource requirement section controls a CPU and memory resource allocations and specifies the distribution *processor units* within a host according to the hardware topology information that LSF collects. The syntax supports basic affinity requirements for sequential jobs, as well as very complex task affinity requirements for parallel jobs.

Enable CPU and memory affinity scheduling with the `AFFINITY` keyword in `lsb.hosts`.

`affinity` sections are accepted by **bsub -R**, and by **bmod -R** for non-running jobs, and can be specified in the **RES_REQ** parameter in `lsb.applications` and `lsb.queues`. Job-level affinity resource requirements take precedence over application-level requirements, which in turn override queue-level requirements.

This feature is only supported for Linux and Power Linux. It is not supported for cpuset or Cray machines.

Running MPI workload through IBM Parallel Environment Runtime Edition

Platform LSF integrates with the IBM Parallel Environment Runtime Edition (IBM PE Runtime Edition) program product - Version 1.3 or later to run PE jobs through the IBM Parallel Operating Environment (POE). The integration enables network-aware scheduling, allowing an LSF job to specify network resource requirements, collect network information, and schedule the job according to the requested network resources. The new integration is based on **blaunch** and the MPI job overhead is much reduced compared to the old PAM/TS framework; four times less by the most conservative measures.

Network requirements can be specified at job submission with the **bsub -network** option, and configured at the queue (`lsb.queues`) and application level (`lsb.applications`) with the NETWORK_REQ parameter.

Supported platforms are AIX and Power Linux (x86_64).

Related information

For more information about IBM Parallel Environment Runtime Edition, see the *IBM Parallel Environment: Operation and Use* guide (SC23-6667).

To access the most recent Parallel Environment documentation in PDF and HTML format, refer to the IBM Clusters Information Center:

`http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp`

Both the current Parallel Environment documentation and earlier versions of the library are also available in PDF format on the IBM Publications Center:

www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss

Previous PE Integration framework for PAM Taskstarter

For LSF 9.1.1 the esub names were changed from `esub.poe` to `esub.oldpoe` and from `esub.tvpoe` to `esub.oldtvpoe`.

New commands

`permapi.so` was added on Linux platforms.

New configuration files

No new configuration files are added for LSF 9.1.1.

Changed commands, options, and output

The following command options and output are new or changed for LSF 9.1.1:

bacct

- The **`-aff`** option displays information about jobs with CPU and memory affinity resource requirement for each task in the job. A table headed `AFFINITY` shows detailed memory and CPU binding information for each task in the job, one line for each allocated processor unit.

  Use only with the `-l` option.

  For example the following job starts 6 tasks with the following affinity resource requirements:

```
bsub -n 6 -R"span[hosts=1] rusage[mem=100]affinity[core(1,same=socket,
exclusive=(socket,injob)):cpubind=socket:membind=localonly:distribute=pack]" myjob
Job <6> is submitted to default queue <normal>.
bacct -l -aff 6

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

Job <6>, User <user1>, Project <default>, Status <DONE>, Queue <normal>, Comma
                     nd <myjob>
Thu Feb 14 14:13:46: Submitted from host <hostA>, CWD <$HOME>;
Thu Feb 14 14:15:07: Dispatched to 6 Hosts/Processors <hostA> <hostA> <hostA>
                     <hostA> <hostA> <hostA>, Effective RES_REQ <sel
                     ect[type == local] order[r15s:pg] rusage[mem=100.00] span[
                     hosts=1] affinity[core(1,same=socket,exclusive=(socket,inj
                     ob))*1:cpubind=socket:membind=localonly:distribute=pack] >
                     ;
Thu Feb 14 14:16:47: Completed <done>.

AFFINITY:
                    CPU BINDING                          MEMORY BINDING
                    -----------                          --------------
HOST                TYPE   LEVEL  EXCL   IDS            POL   NUMA SIZE
hostA               core   socket socket /0/0/0         local 0    16.7MB
hostA               core   socket socket /0/1/0         local 0    16.7MB
hostA               core   socket socket /0/2/0         local 0    16.7MB
hostA               core   socket socket /0/3/0         local 0    16.7MB
hostA               core   socket socket /0/4/0         local 0    16.7MB
hostA               core   socket socket /0/5/0         local 0    16.7MB
```

```
Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.01       81            181     done         0.0001     2M    137M
------------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done jobs:       1      Total number of exited jobs:     0
 Total CPU time consumed:       0.0      Average CPU time consumed:     0.0
 Maximum CPU time of a job:     0.0      Minimum CPU time of a job:     0.0
 Total wait time in queues:    81.0
 Average wait time in queue:   81.0
 Maximum wait time in queue:   81.0      Minimum wait time in queue:   81.0
 Average turnaround time:       181 (seconds/job)
 Maximum turnaround time:       181      Minimum turnaround time:       181
 Average hog factor of a job:  0.00 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.00      Minimum hog factor of a job:  0.00
```

- The **-l** option displays network resource allocations (`PE Network ID`) for IBM Parallel Edition (PE) jobs submitted with the `bsub -network` option, or to a queue (defined in `lsb.queues`) or an application profile (defined in `lsb.applications`) with the NETWORK_REQ parameter defined.

  For example:

```
bacct -l 210
Job <210>, User <user1>;, Project <default>, Status <DONE>. Queue <normal>,
                     Command <my_pe_job>
Tue Jul 17 06:10:28: Submitted from host <hostA>, CWD </home/pe_jobs>;
Tue Jul 17 06:10:31: Dispatched to <hostA>, Effective RES_REQ <select[type
                     == local] order[r15s:pg] rusage[mem=1.00] >, PE Network
                     ID <1111111>  <2222222> used <1> window(s)
                     per network per task;
Tue Jul 17 06:11:31: Completed <done>.
```

bapp

The **-l** option displays **ENV_VARS**, the name/value pairs defined by the application specific environment variables.

bhist

- The **-aff** option displays historical job information about jobs with CPU and memory affinity resource requirement for each task in the job. If the job is pending, the requested affinity resources are displayed. For running jobs, the effective and combined affinity resource allocation decision made by LSF is also displayed, along with a table headed `AFFINITY` that shows detailed memory and CPU binding information for each task, one line for each allocated processor unit. For finished jobs (EXIT or DONE state), the affinity requirements for the job, and the effective and combined affinity resource requirement details are displayed.

  Use only with the `-l` option.

  For example the following job starts 6 tasks with the following affinity resource requirements:

```
bsub -n 6 -R"span[hosts=1] rusage[mem=100]affinity[core(1,same=socket,
exclusive=(socket,injob)):cpubind=socket:membind=localonly:distribute=pack]" myjob
Job <6> is submitted to default queue <normal>.
```

```
bhist -l -aff 6

Job <6>, User <user1>, Project <default>, Command <myjob>
Thu Feb 14 14:13:46: Submitted from host <hostA>, to Queue <normal>, CWD <$HO
                     ME>, 6 Processors Requested, Requested Resources <span[hos
                     ts=1] rusage[mem=100]affinity[core(1,same=socket,exclusive
                     =(socket,injob)):cpubind=socket:membind=localonly:distribu
```

```
                     te=pack]>;
Thu Feb 14 14:15:07: Dispatched to 6 Hosts/Processors <hostA> <hostA> <hostA>
                     <hostA> <hostA> <hostA>, Effective RES_REQ <sel
                     ect[type == local] order[r15s:pg] rusage[mem=100.00] span[
                     hosts=1] affinity[core(1,same=socket,exclusive=(socket,inj
                     ob))*1:cpubind=socket:membind=localonly:distribute=pack] >
                     ;

AFFINITY:
                     CPU BINDING                          MEMORY BINDING
                     -----------------------              --------------------
HOST                 TYPE   LEVEL  EXCL   IDS            POL   NUMA SIZE
hostA                core   socket socket /0/0/0         local 0    16.7MB
hostA                core   socket socket /0/1/0         local 0    16.7MB
hostA                core   socket socket /0/2/0         local 0    16.7MB
hostA                core   socket socket /0/3/0         local 0    16.7MB
hostA                core   socket socket /0/4/0         local 0    16.7MB
hostA                core   socket socket /0/5/0         local 0    16.7MB

Thu Feb 14 14:15:07: Starting (Pid 3630709);
Thu Feb 14 14:15:07: Running with execution home </home/jsmith>, Execution CWD
                     </home/jsmith>, Execution Pid <3630709>;
Thu Feb 14 14:16:47: Done successfully. The CPU time used is 0.0 seconds;
Thu Feb 14 14:16:47: Post job process done successfully;

MEMORY USAGE:
MAX MEM: 2 Mbytes;  AVG MEM: 2 Mbytes

Summary of time in seconds spent in various states by  Thu Feb 14 14:16:47
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  81       0        100      0        0        0        181
```

- The **-l** option displays network resource requirements (`Requested Network` and `PE Network ID`) for IBM Parallel Edition (PE) jobs submitted with the `bsub -network` option.

  For example:

```
bhist -l 749
Job <749>, User <user1>;, Project <default>, Command <my_pe_job>

Mon Jun  4 04:36:12: Submitted from host <hostB>, to Queue <
                     priority>, CWD <$HOME>, 2 Processors Requested, Requested
                     Network <protocol=mpi:mode=US: type=sn_all: instance=1:usage=
                     dedicated>;
Mon Jun  4 04:36:15: Dispatched to 2 Hosts/Processors <hostB>,
                     Effective RES_REQ <select[ty
                     pe == local] rusage[nt1=1.00] >, PE Network
                     ID <1111111>  <2222222> used <1> window(s)
                     per network per task;
Mon Jun  4 04:36:17: Starting (Pid 21006);
```

- **bhist -l** also displays the network requirements and windows usage for run jobs.

bhosts

- The **-aff** option displays host topology information for CPU and memory affinity scheduling. Only the topology nodes containing CPUs in the CPULIST defined in `lsb.hosts` are displayed.

  For example:

```
bhosts -l -aff hostA
HOST  hostA
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
ok              60.00     -      8      0      0      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem  slots
```

```
Total           0.0   0.0   0.0   30%   0.0   193   25     0 8605M  5.8G 13.2G      8
Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M      -


LOAD THRESHOLD USED FOR SCHEDULING:
          r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
loadSched   -     -     -     -       -     -    -     -     -      -      -
loadStop    -     -     -     -       -     -    -     -     -      -      -
```

```
CONFIGURED AFFINITY CPU LIST: all

AFFINITY: Enabled
Host[15.7G]
    NUMA[0: 100M / 15.7G]
        Socket0
            core0(0)
        Socket1
            core0(1)
        Socket2
            core0(2)
        Socket3
            core0(3)
        Socket4
            core0(4)
        Socket5
            core0(5)
        Socket6
            core0(6)
        Socket7
            core0(7)
```

- The **bhosts -l** command displays network resource information network information (PE NETWORK INFORMATION) for IBM Parallel Edition (PE) jobs submitted with the bsub -network option. The physical network ID and status of each network, how many windows are used, and the total number of windows are displayed.

  For example:

  ```
  bhosts -l

  ...
  PE NETWORK INFORMATION
  NetworkID                       Status                  rsv_windows/total_windows
  1111111                         ok                            4/64
  2222222                         closed_Dedicated              4/64
  ...
  ```

bjobs

- The **-aff** option displays information about jobs with CPU and memory affinity resource requirements for each task in the job. If the job is pending, the requested affinity resources are displayed. For running jobs, the effective and combined affinity resource allocation decision made by LSF is also displayed, along with a table headed AFFINITY that shows detailed memory and CPU binding information for each task, one line for each allocated processor unit. For finished jobs (EXIT or DONE state), the affinity requirements for the job, and the effective and combined affinity resource requirement details are displayed.

  Use **bhist -l -aff** to show the actual affinity resource allocation for finished jobs.

  Use only with the -l or -UF option.

  For example the following job starts 6 tasks with the following affinity resource requirements:

```
bsub -n 6 -R"span[hosts=1] rusage[mem=100]affinity[core(1,same=socket,
exclusive=(socket,injob)):cpubind=socket:membind=localonly:distribute=pack]" myjob
Job <6> is submitted to default queue <normal>.
bjobs -l -aff 6

Job <6>, User <user1>, Project <default>, Status <RUN>, Queue <normal>, Comman
                     d <myjob1>
Thu Feb 14 14:13:46: Submitted from host <hostA>, CWD <$HOME>, 6 Processors R
                     equested, Requested Resources <span[hosts=1] rusage[mem=10
                     0]affinity[core(1,same=socket,exclusive=(socket,injob)):cp
                     ubind=socket:membind=localonly:distribute=pack]>;
Thu Feb 14 14:15:07: Started on 6 Hosts/Processors <hostA> <hostA> <hostA
                     > <hostA> <hostA> <hostA>, Execution Home </home/user1
                     >, Execution CWD </home/user1>;

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

 RESOURCE REQUIREMENT DETAILS:
 Combined: select[type == local] order[r15s:pg] rusage[mem=100.00] span[hosts=1
                     ] affinity[core(1,same=socket,exclusive=(socket,injob))*1:
                     cpubind=socket:membind=localonly:distribute=pack]
 Effective: select[type == local] order[r15s:pg] rusage[mem=100.00] span[hosts=
                     1] affinity[core(1,same=socket,exclusive=(socket,injob))*1
                     :cpubind=socket:membind=localonly:distribute=pack]

 AFFINITY:
                     CPU BINDING                          MEMORY BINDING
                     -----------------------              -------------------
 HOST                TYPE   LEVEL  EXCL   IDS            POL   NUMA SIZE
 hostA               core   socket socket /0/0/0         local 0    16.7MB
 hostA               core   socket socket /0/1/0         local 0    16.7MB
 hostA               core   socket socket /0/2/0         local 0    16.7MB
 hostA               core   socket socket /0/3/0         local 0    16.7MB
 hostA               core   socket socket /0/4/0         local 0    16.7MB
 hostA               core   socket socket /0/5/0         local 0    16.7MB
```

- Displays network resource information (`Requested Network`) for IBM Parallel Edition (PE) jobs submitted with the `bsub -network` option.

  For example:

  ```
  bjobs -l
  Job <2106>, User <user1>;, Project <default>;, Status <RUN>;, Queue <normal>, Co
                        mmand <my_pe_job>
  Fri Jun  1 20:44:42: Submitted from host <hostA>, CWD <$HOME>, Requested Network
                         <protocol=mpi: mode=US: type=sn_all: instance=1: usage=dedicated>
  ```

- The `-l` option displays the sort order for resource requirements in the `RESOURCE REQUIREMENT DETAILS` section.

bsub and bmod

- The `-R` option now supports a CPU and memory affinity resource requirements section (affinity[]). The affinity section defines an *affinity resource requirement* string that specifies CPU and memory binding requirements for a resource allocation that is topology aware. An `affinity[]` resource requirement section controls the allocation and distribution of *processor units* within a host according to the hardware topology information that LSF collects.
- The `-network=`*network_res_req*: For LSF IBM Parallel Environment (PE) integration. Specifies the network resource requirements to enable network-aware scheduling for PE jobs. If any network resource requirement is specified in the job, queue, or application profile, the job is treated as a PE job. PE jobs can only run on hosts where IBM PE **pnsd** daemon is running.

The network resource requirement string *network_res_req* has the same syntax as the NETWORK_REQ parameter defined in `lsb.applications` or `lsb.queues`. *network_res_req* has the following syntax:

`[type=sn_all | sn_single]`
`[:protocol=`*protocol_name*`[(`*protocol_number*`)][,`*protocol_name*`[(`*protocol_number*`)]]`
`[:mode=US | IP] [:usage=shared | dedicated] [:instance=`*positive_integer*`]`

LSF_PE_NETWORK_NUM must be defined to a non-zero value in `lsf.conf` for the LSF to recognize the `-network` option. If LSF_PE_NETWORK_NUM is not defined or is set to 0, the job submission is rejected with a warning message.

The `-network` option overrides the value of NETWORK_REQ defined in `lsb.applications` or `lsb.queues`.

lshosts

The **-T** option displays host topology information for each host or cluster, including Maximum memory available on the host followed by the host name and maximum NUMA node memory.

If no NUMA nodes are present, then the NUMA layer in the output is not shown. Other relevant items such as host, socket, core and thread are still shown.

If the host is not available, only the host name is displayed. A dash (-) is shown where available host memory would normally be displayed.

lshosts -T differs from the **bhosts -aff** output:

- Socket and core IDs are not displayed for each NUMA node.
- The requested memory of a NUMA node is not displayed
- **lshosts -T** displays all enabled CPUs on a host, not just those defined in the CPU list in `lsb.hosts`

lsload

If LSF_PE_NETWORK_NUM is set to a value greater than zero in `lsf.conf`, LSF collects network information for scheduling IBM Parallel Environment (PE) jobs. **lsload -l** shows network information for PE jobs. For example, the following **lsload** command displays network information for `hostA` and `hostB`, both of which have 2 networks available. Each network has 256 windows, and **pnsd** is responsive on both hosts. In this case, LSF_PE_NETWORK_NUM=2 should be set in `lsf.conf`:

```
lsload -l
HOST_NAME   status  r15s   r1m  r15m   ut    pg    io  ls    it   tmp   swp   mem   pnsd
pe_network
hostA               ok   1.0   0.1   0.2  10%   0.0     4  12     1   33G 4041M 2208M  Y
ID= 1111111,win=256;ID= 2222222,win=256
hostB               ok   1.0   0.1   0.2  10%   0.0     4  12     1   33G 4041M 2208M  Y
ID= 1111111,win=256;ID= 2222222,win=256
```

New and changed configuration parameters and environment variables

The following configuration parameters and environment variables are new or changed for LSF 9.1.1:

lsb.applications

- **BIND_JOB**

Note: This parameter is obsolete in LSF 9.1.1. You should enable LSF CPU and memory affinity scheduling in with the AFFINITY parameter in `lsb.hosts`. If both BIND_JOB and affinity scheduling are enabled, affinity scheduling takes effect, and BIND_JOB is disabled.

- If `DJOB_ENV_SCRIPT=openmpi_rankfile.sh` is set in `lsb.applications`, LSF creates a host rank file and sets the environment variable LSB_RANK_HOSTFILE.
- **`DJOB_TASK_BIND`**: For CPU and memory affinity scheduling jobs launched with the **`blaunch`** distributed application framework. To enable LSF to bind each task to the proper CPUs or NUMA nodes you must use **`blaunch`** to start tasks. You must set DJOB_TASK_BIND=Y in `lsb.applications` or LSB_DJOB_TASK_BIND=Y in the submission environment before submitting the job. When set, only the CPU and memory bindings allocated to the task itself will be set in each tasks environment. If DJOB_TASK_BIND=N or LSB_DJOB_TASK_BIND=N, or they are not set, each task will have the same CPU or NUMA node binding on one host.
- **`ENV_VARS`** defines application-specific environment variables that will be used by jobs for the application. Use this parameter to define name/value pairs as environment variables. These environment variables are also used in the pre/post-execution environment.
- **`NETWORK_REQ`**: For LSF IBM Parallel Environment (PE) integration. Specifies the network resource requirements for a PE job. If any network resource requirement is specified in the job, queue, or application profile, the job is treated as a PE job. PE jobs can only run on hosts where IBM PE **`pnsd`** daemon is running. The network resource requirement string *network_res_req* has the same syntax as the **`bsub -network`** option.
- **`RES_REQ`**: supports an `affinity[]` section to specify CPU and memory affinity resource requirements.

lsb.hosts

- **`AFFINITY`**: Specifies whether the host can be used to run affinity jobs, and if so which CPUs are eligible to do so. The syntax accepts Y, N, a list of CPUs, or a CPU range.

lsb.params

- **`DEFAULT_RESREQ_ORDER=order_string`**: Specifies the global LSF default sorting order for resource requirements so the scheduler can find the right candidate host.
- **`JOB_DISTRIBUTE_ON_HOST=pack | balance | any`**: For NUMA CPU and memory affinity scheduling. Specifies how LSF distributes tasks for different jobs. For CPU and memory affinity jobs, if `JOB_INCLUDE_POSTPROC=Y`, LSF does not release affinity resources until post-execution processing has finished, since slots are still occupied by the job during post-execution processing.
- **`MAX_CONCURRENT_QUERY`** controls the maximum number of concurrent query commands. The parameter applies to all query commands and defines the maximum batch queries (including job queries) that **`mbatchd`** can handle. **`MAX_CONCURRENT_QUERY`** replaces the **`MAX_CONCURRENT_JOB_QUERY`** parameter, which is obsolete in LSF 9.1.1.
- **`MAX_PROTOCOL_INSTANCES=`***integer*: For LSF IBM Parallel Environment (PE) integration. Specify the number of parallel communication paths (windows) available to the protocol on each network. If number of windows specified for the job (with the `instance` option of `bsub -network` or the NETWORK_REQ parameter in `lsb.queues` or `lsb.applications`) is greater than the specified maximum value, LSF rejects the job.

Specify MAX_PROTOCOL_INSTANCES in a queue (lsb.queues) or cluster-wide in lsb.params. The value specified in a queue overrides the value specified in lsb.params.

- **STRIPING_WITH_MINIMUM_NETWORK=y | n**: For LSF IBM Parallel Environment (PE) integration. Specifies whether or not nodes which have more than half of their networks in READY state are considered for PE jobs with type=sn_all specified in the network resource requirements (in the bsub -network option or the NETWORK_REQ parameter in lsb.queues or lsb.applications). This makes certain that at least one network is UP and in READY state between any two nodes assigned for the job.

  When set to y, the nodes which have more than half the minimum number of networks in the READY state are considered for **sn_all** jobs. If set to n, only nodes which have all networks in the READY state are considered for **sn_all** jobs.

lsb.queues

- **EXCLUSIVE**: EXCLUSIVE=Y or EXCLUSIVE=CU[*cu_type*] must be configured to enable affinity jobs to use CPUs exclusively, when the alljobs scope is specified in the exclusive option of an affinity[] resource requirement string.
- **MAX_PROTOCOL_INSTANCES=***integer*: For LSF IBM Parallel Environment (PE) integration. Specify the number of parallel communication paths (windows) available to the protocol on each network. If number of windows specified for the job (with the instance option of bsub -network or the NETWORK_REQ parameter in lsb.queues or lsb.applications) is greater than the specified maximum value, LSF rejects the job.

  Specify MAX_PROTOCOL_INSTANCES in a queue (lsb.queues) or cluster-wide in lsb.params. The value specified in a queue overrides the value specified in lsb.params.
- **NETWORK_REQ**: For LSF IBM Parallel Environment (PE) integration. Specifies the network resource requirements for a PE job. If any network resource requirement is specified in the job, queue, or application profile, the job is treated as a PE job. PE jobs can only run on hosts where IBM PE **pnsd** daemon is running. The network resource requirement string *network_res_req* has the same syntax as the **bsub -network** option.
- **RES_REQ**: supports an affinity[] section to specify CPU and memory affinity resource requirements.

lsf.conf

- **LSF_BIND_JOB**: *This parameter is obsolete in LSF 9.1.1.* You should enable LSF CPU and memory affinity scheduling in with the AFFINITY parameter in lsb.hosts. If both LSF_BIND_JOB and affinity scheduling are enabled, affinity scheduling takes effect, and LSF_BIND_JOB is disabled.
- **LSF_PE_NETWORK_NUM=***num_networks*: For LSF IBM Parallel Environment (PE) integration. When LSF collects network information for PE jobs, LSF_PE_NETWORK_UPDATE_INTERVAL specifies the interval for updating network information. LSF_PE_NETWORK_NUM must be defined with a non-zero value in lsf.conf for LSF to collect network information to run PE jobs.
- **LSF_PE_NETWORK_UPDATE_INTERVAL=***seconds*: For LSF IBM Parallel Environment (PE) integration. When LSF collects network information for PE jobs, LSF_PE_NETWORK_UPDATE_INTERVAL specifies the interval for updating network information.

Environment variables

- LSB_AFFINITY_HOSTFILE: defines the Path to the CPU and memory affinity binding decision file for CPU and memory affinity jobs.
- LSB_BIND_CPU_LIST: contains allocated CPUs on each host. LSF will combine all of the allocated CPUs for all the tasks that ended up on the host (for the given job), and set this environment variable to the entire list before launching tasks.
- LSB_BIND_MEM_LIST: contains allocated memory nodes on each host. If the job is submitted with a memory affinity requirement, LSF will combine all of the allocated NUMA node IDs for all the tasks that ended up on the host (for the given job), and set this environment variable to the entire list before launching tasks. The following example corresponds to the tasks specified in the example for LSB_AFFINITY_HOSTFILE:
- LSB_BIND_MEM_POLICY: for jobs submitted with a memory affinity resource requirements, LSF sets the memory binding policy in LSB_BIND_MEM_POLICY environment variable, as specified in the `membind` affinity resource requirement parameter: either `localprefer` or `localonly`.
- LSB_DJOB_PE_NETWORK: specifies network resource information for IBM Parallel Environment (PE) jobs submitted with the `bsub -network` option, or to a queue (defined in `lsb.queues`) or an application profile (defined in `lsb.applications`) with the NETWORK_REQ parameter defined.
- LSB_DJOB_TASK_BIND: specifies task binding for CPU and memory affinity scheduling jobs launched with the **`blaunch`** distributed application framework. If LSB_DJOB_TASK_BIND=Y in the submission environment before submitting the job, you must use **`blaunch`** to start tasks to enable LSF to bind each task to the proper CPUs or NUMA nodes Only the CPU and memory bindings allocated to the task itself will be set in each tasks environment. If LSB_DJOB_TASK_BIND=N, or it is not set, each task will have the same CPU or NUMA node binding on one host.
- LSB_RANK_HOSTFILE: specifies the path to the OpenMPI host rank file specified by the `DJOB_ENV_SCRIPT` option in `lsb.applications`.
- OMP_NUM_THREADS: LSF will set the environment variable OMP_NUM_THREADS to the total number of CPUs allocated to each task if the job requests more than one per task (that is, the job is a multithreaded job).
- RM_CPUTASK*n*: is the IBM Parallel Environment equivalent of LSB_BIND_CPU_LIST. Each RM_CPUTASK represents the CPU allocation for one task, as opposed to having a single variable which amalgamates them all.
- RM_MEM_AFFINITY: informs the IBM Parallel Environment memory binding policy. If a local-only policy (`membind=localonly` affinity resource requirement parameter) has been set, RM_MEM_AFFINITY will be set to Yes. If local preference policy (`membind=localprefer` affinity resource requirement parameter) has been set, RM_MEM_AFFINITY will be set to No.

New and changed accounting and job event fields

lsb.acct and lsb.events

New records and fields have been added for LSF 9.1.1 See the *IBM Platform LSF Configuration Reference* for more details.

What's New in Platform LSF Version 9.1

New and changed behavior

Platform Dynamic Cluster

IBM Platform Dynamic Cluster is an add-on to Platform LSF that provides the following benefits:

- Dynamically create new virtual machines to satisfy job demands
- Allow jobs to be saved and migrated to another host to release the resources of a priority host
- Restrict job memory usage on a host by running it in a virtual machine, avoiding the possibility of one job hoarding all of a host's memory and interfering with other running jobs

See *Using IBM Platform Dynamic Cluster* for more information.

Fast job dispatch

This is a general performance enhancement related to **mbschd** and **mbatchd**. It combines six different performance items:

- Multithreaded resource evaluation is applied to improve resource matching performance for large clusters with huge numbers of hosts (LSF Advanced Edition only).
- Optimized job dependency evaluation: Set the maximum number of job dependencies that **mbatchd** evaluates in one scheduling cycle.
- Separate job information directory: To improve job file I/O operations and cluster performance, you can specify a job information directory which is separate from the LSB_SHARE directory.
- Enhanced job dispatching by queue: Increase queue responsiveness by publishing the scheduling decision for the specified queue without waiting for the whole scheduling session to finish (LSF Standard and Advanced Editions only).
- Improved fairshare for job selection performance.

Batch query enhancement

LSF Standard Edition now includes the multithreaded batch query feature. Multithreading is extended (via the **LSB_QUERY_ENH** parameter in lsf.conf) to include all batch query commands except **bread**, **bstatus** and **tspeek**. 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.

bswitch for arrays

You can improve **mbatchd** performance when switching large job arrays to another queue by enabling the **JOB_SWITCH2_EVENT** in lsb.params. This lets **mbatchd** generate the JOB_SWITCH2 event log. JOB_SWITCH2 logs the switching of the array to another queue as fewer events instead of logging the switching of each individual array element.

Scheduler performance metrics

LSF automatically optimizes the default values for **LSB_MAX_JOB_DISPATCH_PER_SESSION** and **MAX_SBD_CONNS** based on the size of the cluster. The additional performance metrics are displayed to help customers analyze the performance of the scheduler in the cluster.

Logging mbatchd performance metrics

This feature lets you log performance metrics for **mbatchd**. This feature is useful for troubleshooting large clusters where a cluster has performance problems. In such cases, **mbatchd** performance may be slow in handling high volume request such as job submission, job status requests, and job rusage requests.

Process tracking with cgroup

Process tracking through `cgroups` can capture job processes that are not in the existing job's process tree and have process group IDs that are different from the existing ones, or job processes that run very quickly, before LSF has a chance to find them in the regular or on-demand process table scan issued by PIM.

This feature depends on the Control Groups (`cgroups`) functions provided by the LINUX kernel. The cgroups functions are supported on x86_64 and PowerPC LINUX with kernel version 2.6.24 or later.

Memory utilization

In large EDA clusters, memory is more valuable than cores. LSF 9.1 provides improved utilization and reporting of memory usage. Memory utilization involves collecting and displaying the current memory usage, peak memory usage and average memory usage. LSF also reports the delta between the requested memory (if specified) and the actual peak in the job report mail that is sent when the job completes. You can adjust rusage accordingly next time for the same job submission if consumed memory is larger or smaller than current rusage.

Kerberos 5 integration enhancement

Kerberos 5 integration enhancement for LSF includes the following features:

- The dedicated binary **krbrenewd** renews Ticket Granting Ticket (TGT) for pending jobs and running jobs. It is enhanced to handle a large number of jobs without creating too much overhead for **mbatchd** and KDC.
- Separate user TGT forwarding from daemon authentication with a parameter, **LSB_KRB_TGT_FWD**, to control TGT forwarding.
- Kerberos solution package is preinstalled in the LSF install directory, relieving users from compiling from source code. krb5 function calls are dynamically linked.
- Preliminary TGT forwarding support for parallel jobs, including shared directory support for parallel jobs. If all hosts at a customer site have a shared directory, you can configure this directory in lsf.conf via parameter **LSB_KRB_TGT_DIR**, and the TGT for each individual job will be stored here.

Portable hardware locality

Portable Hardware Locality (hwloc) is an open source software package distributed under BSD license. It provides a portable abstraction (across OS, versions,

architectures, etc.) of the hierarchical topology of modern architectures, including NUMA memory nodes, socket, shared caches, cores and simultaneous multithreading (SMT). It also gathers various system attributes such as cache and memory information as well as the locality of I/O device such as network interfaces. It primarily aims at helping applications with gathering information about computing hardware. Hwloc can support all platforms LSF supports

Hwloc is integrated into LSF to detect hardware information. It detects each host hardware topology when the LIM starts and the host topology information is changed. The master LIM detects the topology of the master host. The slave LIM detects the topology of the local host. It updates the topology information to the master host when it joins the cluster or sends topology information to the master LIM for host configuration. Host topology information is updated once the hardware topology changes. Hardware topology changes if any NUMA memory node, caches, socket, core, PU, etc., changes. Sometimes topology information changes even though the core number did not change.

The commands **lim -T** and **lshosts -T** display host topology information.

New built-in slots and maxslots keywords

This feature is useful when you want to pack sequential jobs onto hosts with the least slots first, ensuring that more hosts will be available to run parallel jobs.

Two new keywords, slots and maxslots, are introduced. The slots keyword refers to the number of unused slots on a host, and the maxslots keyword refers to the maximum number of slots that can be used on a host. The slots keyword can be used in the select[] and order[] sections. The maxslots keyword can be used in the select[], order[], and same[] sections. For example, to submit a job to hosts with the with the fewest unused slots:

```
bsub -R "order[-slots]" myJob
```

This tells LSF to sort candidate hosts from fewest to greatest number of unused slots when considering the job for scheduling.

Guaranteed resources

Use guaranteed resources to have LSF reserve a minimum amount of resources for a group of jobs, such as jobs for a queue, a project, a user group, an application, or some combination of these. In previous releases, LSF supported guarantee policies on slots or whole hosts (from a given set of hosts), and for licenses managed by LSF License Scheduler.

LSF 9.1 allows for guarantee policies on packages of resources, where a package consists of a number of slots and some amount of memory together on a single host. For example, suppose that jobs for a queue all require 1 slot and 32 GB to run. You want to make sure that this queue can run at least 10 jobs concurrently whenever there is demand. To achieve this, configure a guarantee for the queue of 10 packages, with each package composed of 1 slot and 32 GB.

As in previous releases, the default behavior is for LSF to keep sufficient cluster resources idle in order to honor guarantees. Optionally, you can configure LSF to allow these resources to be borrowed by other jobs when not required immediately for guarantees.

Use `bresources -g -l -m guaranteed_resource_pool_name` to display the configuration and status of the guaranteed resource pool configured in the GuaranteedResourcePool section of `lsb.resources` in a long multiline format. The `-m` option displays the names of all hosts included in each guaranteed resource pool configuration from the GuaranteedResourcePool section of `lsb.resources`.

Memory preemption

Platform LSF now allows preemption for memory in addition to slots and licenses. In `lsb.params`, configure PREEMPTABLE_RESOURCES = mem to enable this feature. High priority jobs can preempt for both memory and slots. You can also set other user-defined resources as preemptable with this parameter, as before.

Dynamically update user group information

Historically, if externally-defined user groups were defined the **mbatchd** would use the egroup executable at startup and reconfig to determine user group membership. Now it is possible to configure the **mbatchd** to evaluate externally-defined user group membership on a periodic basis by configuring the parameter **EGROUP_UPDATE_INTERVAL**=n in the `lsb.params` file, where n is the number of hours between user group evaluation.

Alternative resource requirements

There are times when a job could run with different sets of resources. It is now possible to specify multiple *alternative resource requirements* when submitting a job. When LSF attempts to schedule a job it considers each alternative in turn. If none of the alternatives can be satisfied, the job will pend. An alternative resource requirement consists of two or more simple or compound resource requirements.

Effective resource requirements

The concept of *effective resource requirement* has been introduced for LSF 9.1. Only running jobs, or jobs that have finished have an effective resource requirement. An effective resource requirement is the resource requirement used to determine a job's allocation (normally determined by the scheduler). A job's effective resource requirement can be viewed in the long output of the **bjobs** and **bhist** commands. A job's effective resource requirement is invariant across configuration changes and can be changed using the **bmod** command.

Combined resource requirements

Combined resource requirements are the result of **mbatchd** merging job, application, and queue level resource requirements for each job. This feature lets you control how many levels of resource requirements bjobs will display. You can set the **BJOBS_RES_REQ_DISPLAY** parameter to one of the following values:

- None: **bjobs -l** does not display any level of resource requirement.
- Brief: **bjobs -l** only displays the combined and effective resource requirements.
- Full: **bjobs -l** displays the job, app, queue, combined and effective resource requirement.

rusage threshold

The keyword threshold in the rusage section of a resource requirement string lets you specify a threshold at which the consumed resource must be before an

allocation should be made. If the threshold is not satisfied for every host in the cluster, the job becomes pending. To specify a threshold in the command line, use `bsub -R` to attach a threshold to a resource in the rusage section. For decreasing resources, threshold is interpreted as a minimum value. For increasing resources, threshold is interpreted as a maximum value.

Running jobs with Windows Terminal Services

To dispatch a Terminal Server job to Windows client hosts, set the following parameters in `lsf.conf` after installation, then restart the LIM (by running `lsadmin limrestart all`) and restart the TSJobHelper Windows service on the execution hosts and helper hosts to apply the changes to these parameters:

- LSB_TSJOBS_HELPER_HOSTS
- LSB_TSJOBS_HELPER_PORT
- LSB_TSJOBS_HELPER_TIMEOUT

To run a Terminal Server job on a Windows 2008 host, you must install the Terminal Server Role service (also called the Remote Desktop Role in some distributions). Run the Server Manager Add Roles wizard to add the service. You should also use the RemoteApp Manager to check if `lstsmgr.exe` and `tscon.exe` are listed in allowed programs list after LSF installation.

Flexible job output directory

The flexible job output directory feature lets you create and manage the job output directory dynamically based on configuration parameters. This feature is useful if you are running applications that have specific requirements for job output directory, such as copying data to the directory after the job finishes. This feature ensures this data will not be overwritten. A job output directory can be specified with **bsub -outdir** or through the DEFAULT_JOB_OUTDIR configuration parameter in `lsb.params`.

Flexible current working directory (CWD)

The CWD feature lets you create and manage the job CWD dynamically based on configuration parameters, and if the path includes dynamic patterns. This feature is useful if you are running applications that have specific requirements for job CWD, such as copying data to the directory before the job starts running. The CWD feature ensures this data will not be overwritten.

Parallel mbatchd restart (LSF Standard and Advanced Editions)

A new option, **badmin mbdrestart -p**, allows parallel **mbatchd** restart. This will fork a child **mbatchd** process to help minimize downtime. LSF starts a new or child **mbatchd** process to read the configuration files and replay the event file. The old master **mbatchd** can respond to client commands (**bsub**, **bjobs**, etc.), handle job scheduling and status updates, dispatching, and updating new events to event files. When complete, the child takes over as master **mbatchd**, and the old master **mbatchd** dies

LSF daemon communications

LSF now uses non-privileged ports by default for daemon communication. You can set **LSF_NON_PRIVILEGED_PORTS=N** in `lsf.conf` to enable privileged port

communication. Also, **LSF_MC_NON_PRIVILEGED_PORTS** and **LSF_NON_PRIVILEGED_PORTS** are now fully decoupled, which is different from previous versions.

If you are upgrading your master host, and leaving some server hosts still running older versions, then you should do the following:

- If LSF_NON_PRIVILEGED_PORTS is already set to Y or N, continue with upgrade.
- If LSF_NON_PRIVILEGED_PORTS is not set, but LSB_MAX_JOB_DISPATCH_PER_SESSION is set to a value greater than 300 do the following:
  1. Shut down the cluster
  2. Set LSF_NON_PRIVILEGED_PORTS=Y
  3. Upgrade the master host
  4. Restart the cluster
- If neither LSF_NON_PRIVILEGED_PORTS nor LSB_MAX_JOB_DISPATCH_PER_SESSION is set, do the following:
  1. Set LSF_NON_PRIVILEGED_PORTS=N.
  2. Upgrade the master host.
  3. Start LSF on the master host.

See *Upgrading IBM Platform LSF on UNIX and Linux* for detailed upgrade steps.

Diagnose query requests

LSF provides **mbatchd** system query monitoring mechanisms to help administrators diagnose problems with clusters. This is useful when query requests generate a heavy load on the system, slowing down LSF and preventing responses to requests. Some possible causes of performance degradation by query requests include the following

Show LSF system runtime information

Use **badmin showstatus** to display current LSF runtime information about the whole cluster, including information about hosts, jobs, users, and **mbatchd** startup and reconfiguration.

```
% badmin showstatus
LSF runtime mbatchd information
    Available local hosts (current/peak):
        Clients:                0/0
        Servers:                8/8
           CPUs:               14/14
           Cores:              50/50
           Slots:              50/50

    Number of servers:          8
        Ok:                     8
        Closed:                 0
        Unreachable:            0
        Unavailable:            0

    Number of jobs:             7
        Running:                0
        Suspended:              0
        Pending:                0
        Finished:               7
```

```
Number of users:            3
Number of user groups:      0
Number of active users:     1

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
```

Hung job management

If a job is submitted with a run limit, Platform LSF attempts to kill the job after it reaches the run limit. A job becomes a *hung job* if Platform LSF cannot kill the job after its run limit is expired (specifically, **mbatchd** attempts to send a signal to **sbatchd** to kill a job, but **sbatchd** is unable to kill the job).

Hung jobs continue to reserve shared resources, making the shared resources unavailable for other jobs. When hung job management is enabled with REMOVE_HUNG_JOBS_FOR in `lsb.params`, Platform LSF removes hung jobs after the grace period expires. The grace period only begins once a job run limit expires.

Rapid detection of host failure

LSF 9.1 can more rapidly detect host failure by using a non-blocking connection to probe **sbatchd**, which allows **mbatchd** to probe more **sbatchd**s in one cycle. The timeout when **sbatchd** is down has also been shortened. When LIM reports SBDDOWN or OK, but the current host status in **mbatchd** does not match the status reported by LIM, a probe will be requested to confirm the **sbatchd** status. When LIM reports UNAVAIL, a probe will be requested to confirm the **sbatchd** status.

Temporary job directories

Jobs use temporary directories for working files and temporary output. By default, LSF uses the default operating system temporary directory. To enable and use temporary directories specific to each job, specify `LSF_TMPDIR=`*`directory_name`* in `lsf.conf`.

The name format of the job temporary directory has changed. For an array job element the file name format changes to `LSF_TMPDIR/`*`jobID_array_index`*`.tmpdir`. For a single jobs, `LSF_TMPDIR/`*`jobID`*`.tmpdir` is not changed. For backward compatibility, LSF tries `LSF_TMPDIR/`*`64bit_jobid`*`.tmpdir` if the directory in new naming formate does not exist.

Platform MultiCluster

Multisite configuration

Platform MultiCluster now supports common `lsf.shared` and `lsb.applications` configuration files. `#include` lines must be inserted before any real configuration setting of the local configuration file. Multiple `#include` sections are supported in a single local configuration file. The `#include` syntax can only appear in local configuration files. `#include` syntax appearing in common configuration files will be treated as comments and ignored.

Job priority support

The Platform MultiCluster job forwarding protocol is enhanced to send initial job priority to the execution cluster. Both the submission cluster and the execution cluster must define MAX_USER_PRIORITY parameters. Forwarded job priority will be scaled based on MAX_USER_PRIORITY values on both clusters: **bjobs –l** on the execution cluster shows forwarded job priority.

Remote job modification

You can now modify a forwarded job from the submission cluster. Modification of most job parameters is supported except for following:

- Requested hosts (`-m`) and requested clusters (`-clusters`)
- Job Dependency (`-w`)
- Queue (`-q`)
- SLA (`-sla`)

Remote job modification is supported if the execution cluster is LSF version 9.1 or later.

Cluster preference

A new `-clusters` option for **bsub** and **bmod** allows you to specify candidate clusters for the job. The scheduler will take the intersection of job-level cluster candidates and remote cluster candidates defined in the SNDJOBS_TO parameter.

Numeric and string resource scheduling

A new parameter MC_RESOURCE_MATCHING_CRITERIA in `lsb.params` of the execution cluster defines numeric and string resources that can be considered in MultiCluster resource requirement evaluation. Resources in MC_RESOURCE_MATCHING_CRITERIA must be host-based resources. Numeric resources must be non-consumable.

Dynamic cluster weighting

A new MultiCluster job forwarding policy considers cluster and queue preference, available slots, pending queue length and slot threshold on the remote queue. The policy is enabled by MC_PLUGIN_SCHEDULE_ENHANCE=DYN_CLUSTER_WEIGHTING in `lsb.params` from the submission cluster. The MC_PLUGIN_UPDATE_INTERVAL parameter must be defined on the execution cluster to collect the available slots on each remote queue.

Visibility of remote job ID

bjobs, **bhist** and **bacct** are enhanced to display both submission job ID and execution job ID from both clusters. **bjobs** is enhanced to support job query based on the submission job ID and submission cluster name.

Recall a forwarded job

Use **brequeue –p** to recall a remote pending job back to submission cluster. The recalled job will be put into the front of the queue to be forwarded as high priority. **brequeue –p** can only be used to requeue remote pending jobs.

Pending job threshold on receive queue

A new **IMPT_SLOTBKLG** parameter in `lsb.queues` of the receive queue on the execution cluster defines a threshold for the number of pending job slots in the receive queue of the remote execution cluster.

New commands

The following command is new for LSF 9.1:

bdc

The new **bdc** command provides a set of subcommands to monitor Dynamic Cluster. If no subcommands are supplied, **bdc** prompts for a subcommand from standard input. Information about each subcommand is available through the `bdc help` subcommand.

New configuration files

The following configuration file is new for LSF 9.1:

dc_conf.cluster_name.xml

This is a new file for Platform Dynamic Cluster configuration parameters.

Changed commands, options, and output

The following command options and output are new or changed for LSF 9.1:

bacct

The **-l** option output now shows the following as a job termination reason if the job was removed after reaching a run time limit:

```
TERM_REMOVE_HUNG_JOB: Job removed from LSF system after reaching a job
limit
```

If the **-l** option is run from an older version of **bacct**, the job termination reason is unknown:

```
TERM_UNKNOWN: job exited, reason unknown
```

badmin

badmin mbdrestart -p allows parallel **mbatchd** restart. This will fork a child *mbatchd* process to help minimize downtime for LSF. LSF starts a new or child **mbatchd** process to read the configuration files and replay the event file. The old master **mbatchd** can respond to client commands (**bsub**, **bjobs**, etc.), handle job scheduling and status updates, dispatching, and updating new events to event files. When complete, the child takes over as master **mbatchd**, and the old master **mbatchd** dies

bhist

The **-l** option output now shows the following as a job termination reason if the job was removed after reaching a run time limit:

```
TERM_REMOVE_HUNG_JOB: Job removed from LSF system after reaching a job
limit
```

If the **-l** option is run from an older version of **bhist**, the job termination reason is unknown:

```
TERM_UNKNOWN: job exited, reason unknown
```

The **-l** option output now shows memory usage.

The **-app** option output displays accounting information about jobs submitted to the specified application profile. You must specify an existing application profile configured in `lsb.applications`.

bjobs

The **-l** option output now shows the following as a job termination reason if the job was removed after reaching a run time limit:

```
TERM_REMOVE_HUNG_JOB: Job removed from LSF system after reaching a job
limit
```

If the **-l** option is run from an older version of **bjobs**, the job termination reason incorrectly shows that the termination request is issued and the job will be killed when the host is ok:

```
Termination request issued; the job will be killed once the host is ok;
```

This is only an issue with an older version of **bjobs** not correctly processing a new termination reason as unknown.

The **-l** option output now shows memory usage.

The **-fwd** option In MultiCluster job forwarding mode filters output to display information on forwarded jobs, including the forwarded time (FORWARD_TIME) and the name of the cluster (CLUSTER) to which the job was forwarded. **-fwd** can be used with other options to further filter the results.

The **-cname** option in LSF Advanced Edition includes the cluster name for execution cluster hosts in output for **bjobs -l**.

The **-sum** option 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.

The **-UF** option displays unformatted job detail information. This makes it easy to write scripts for parsing keywords on bjobs. The results of this option have no wide control for the output. Each line starts from the beginning of the line. Information for **SCHEDULING PARAMETERS** and **PENDING REASONS** remain formatted. The rusage message lines ending without any separator have a semicolon added to separate their different parts. The first line and all lines starting with the time stamp are displayed unformatted in a single line. There is no line length and format control.

blimits

The **-fwd** option in LSF Advanced Edition 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.

bmgroup and bhosts

The **-cname** option in LSF Advanced Edition 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.

brequeue

The **-p** option in the MultiCluster job forwarding mode requeues specified jobs that are pending in a remote cluster for MultiCluster job forwarding modes.

bresources

Use `bresources –g –l –m` *guaranteed_resource_pool_name* to display the configuration and status of the guaranteed resource pool configured in the GuaranteedResourcePool section of `lsb.resources` in a long multiline format. The `-m` option displays the names of all hosts included in each guaranteed resource pool configuration from the GuaranteedResourcePool section of `lsb.resources`.

bsub and bmod

The **-cwd** option specifies the current working directory (CWD) for job execution. The system creates a CWD if the path for the CWD includes dynamic patterns for both absolute and relative paths. LSF cleans the created CWD based on the time to live value set in JOB_CWD_TTL in `lsb.params`.

The **-outdir** option creates the job output directory. The **-outdir** option supports the following dynamic patterns for the output directory:

- `%J` - job ID
- `%JG` - job group (if not specified, it will be ignored)
- `%I` - array index (default value is 0)
- `%EJ` - execution job ID
- `%EI` - execution index
- `%P` - project name
- `%U` - user name
- `%G` - User group

The **-clusters** option specifies cluster names when submitting jobs for Platform MultiCluster.

lshosts

The **-cname** option in LSF Advanced Edition 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.

The **-T** option displays host topology information for each host or cluster.

New and changed configuration parameters and environment variables

The following configuration parameters and environment variables are new or changed for LSF 9.1:

lsb.applications

- **JOB_CWD**: the current working directory (CWD) for the job in the application profile. The path can be absolute or relative to the submission directory.

lsb.params

- **BSWITCH_MODIFY_RUSAGE**: Enable this parameter to allow **bswitch** to update job resource usage according to the resource requirements in the new queue.
- **DEFAULT_JOB_CWD**: Specifies the cluster wide current working directory (CWD) for the job. The path can be absolute or relative to the submission directory.
- **DEFAULT_JOB_OUTDIR**: Set this parameter for LSF to create a cluster wide output directory for the job. Once set, the system starts using the new directory and always tries to create the directory if it does not exist. The directory path can be absolute or relative to the submission directory with dynamic patterns.
- **DIAGNOSE_LOGDIR**: You must enable the ENABLE_DIAGNOSE parameter for DIAGNOSE_LOGDIR to take effect. Set DIAGNOSE_LOGDIR to specify the file location for the collected information. The log file shows who issued these requests, where the requests came from, and the data size of the query. If you do not modify this parameter, the default location for the log file is LSF_LOGDIR.. The name of the log file is `query_info.querylog.`*`host_name`*. You can dynamically set the path from the command line with `badmin diagnose -c query -f` *`log_name`*, where *log_name* can be a full path. This overrides any other setting for the path. However, if you restart or reconfigure **mbatchd**, this path setting is lost and it defaults back to the setting you specified in this parameter.
- **EGO_SLOTBASED_VELOCITY_SLA**: enables slot based requirements for EGO-enabled SLA. If the value is N, LSF calculates how many slots you need by the number of jobs. If the value is Y, LSF calculates how many slots you need by the number of job slots instead of the number of jobs.
- **EGROUP_UPDATE_INTERVAL**: Specifies a time interval, in hours, for which dynamic user group information in `lsb.users` will be updated automatically.
- **ENABLE_DIAGNOSE**: Enable this parameter for **mbatchd** to write query source information to a log file. The log file shows information about the source of **mbatchd** queries, allowing you to troubleshoot problems. The log file shows who issued these requests, where the requests came from, and the data size of the query. The log file collects key information like query name, user name, host name and the data size of the query. You can write a script to format the output.
- **EVALUATE_JOB_DEPENDENCY**: sets the maximum number of job dependencies **mbatchd** evaluates in one scheduling cycle. This parameter limits the amount of time **mbatchd** spends on evaluating job dependencies in a scheduling cycle, which limits the amount of time the job dependency evaluation blocks services. Job dependency evaluation is a process that is used to check if each job's dependency condition is satisfied. When a job's dependency condition is satisfied, it sets a ready flag and allows itself to be scheduled by **mbschd**.
- **JOB_CWD_TTL**: Specifies the *time-to-live* (TTL) for the current working directory (CWD) of a job. LSF cleans created CWD directories after a job finishes based on the TTL value. LSF deletes the CWD for the job if LSF created that directory for the job.

- **PEND_EXTSCHED_REASON**: displays a detailed pending reason in **bjobs -p** and **bjobs -l** output.
- **PREEMPTABLE_RESOURCES = mem**: enables high priority jobs to preempt for both memory and slots.
- **REMOVE_HUNG_JOBS_FOR**: enables hung job management, which allows LSF to remove hung jobs. A job becomes a *hung job* if LSF cannot kill the job after its run limit is expired (specifically, **mbatchd** attempts to send a signal to **sbatchd** to kill a job, but **sbatchd** is unable to kill the job). Hung jobs occur because of one of the following reasons:
  - **sbatchd** on the execution host is down (that is, the host is in the unreach or unavail status). Jobs running on an execution host when **sbatchd** goes down go into the UNKWN state. These UNKWN jobs continue to occupy shared resources, making the shared resources unavailable for other jobs.
  - Reasons specific to the operating system on the execution host. Jobs that cannot be killed due to an issue with the operating system remain in the RUN state even after the run limit is expired.
- **SCHED_PER_JOB_SORT**: Enables the per-job sorting feature in scheduler. The feature is disabled by default.
- **SCHEDULER_THREADS**: sets the number of threads the scheduler uses to evaluate resource requirements. Multithreaded resource evaluation is useful for large scale clusters with large numbers of hosts. The idea is to do resource evaluation for hosts concurrently. For example, there are 6,000 hosts in a cluster, so the scheduler may create six threads to do the evaluation concurrently. Each thread is in charge of 1,000 hosts.

  Multithreaded resource requirement evaluation requires LSF_STRICT_RESREQ=Y to be enabled in `lsf.conf`.

lsb.queues

- **SUCCESS_EXIT_VALUES**: Specifies exit values used by LSF to determine if the job was done successfully. This parameter can now be specified at the queue level.

  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 configration in `lsb.queues` and `lsb.applications`.

  In the MultiCluster lease model, if a job is running on a host in a provider cluster with an older version of LSF, the queue-level SUCCESS_EXIT_VALUES do not take effect.
- **DISPATCH_BY_QUEUE**: Set this parameter to increase queue responsiveness. The scheduling decision for the specified queue will be published without waiting for the whole scheduling session to finish. The scheduling decision for the jobs in the specified queue is final and these jobs cannot be preempted within the same scheduling cycle.
- **IMPT_SLOTBKLG**: In MultiCluster job forwarding model, specifies the MultiCluster pending job slot limit for a receive-jobs queue. In the submission cluster, if the total of requested job slots and the number of imported pending slots in the receiving queue is greater than IMPT_SLOTBKLG, the queue stops accepting jobs from remote clusters, and the job is not forwarded to the receiving queue. Specify an integer between 0 and 2147483646 for the number of slots. Use the keyword infinit to make the queue accept an unlimited number of pending MultiCluster job slots. Set IMPT_SLOTBKLG to 0 to forbid any job being forwarded to the receiving queue.

lsb.resources

- The Limit section has two new consumers: **LIC_PROJECTS** to enforce limits on specific license projects, and **PER_LIC_PROJECT** to enforce per-project limits on license projects.

lsf.conf

- **LSB_CPUSET_DISPLAY_CPULIST**: The **bjobs**, **bhist**, and **bacct -l** commands display the CPU IDs in the dynamic cpuset allocated on each host. The CPU IDs are displayed as CPUS=cpu_ID_list after NCPUS=num_cpus for each host. The cpu_ID_list is displayed in condensed format as a range of continuous IDs.
- **LSB_JOBINFO_DIR**: use this parameter to specify a directory for job information instead of using the default directory. If this parameter is specified, LSF directly accesses this directory to get the job information files. By default, the job information directory is located in the LSF shared directory, which is in the same file system as the one used for logging events. In large scale clusters with millions of single jobs, there are several job files in the job information directory. The job information directory requires random read/write operations on multiple job files simultaneously, while the event log directory has events appended to a single events file. LSF uses separate daemons for these two types of operations. Because both directories have different file requirements, using a dedicated file system for both directories is useful in larger scale clusters to optimize the file systems for the specific requirements.
- **LSB_KRB_LIB_PATH**: specifies the library path that contains the krb5 libraries: `libkrb5.so`, `libcom_err.so`, `libk5cryto.so`, and `libkrb5support.so`.
- **LSB_KRB_CHECK_INTERVAL**: sets a time interval for how long **krbrenewd** and root **sbatchd** should wait before the next check. If this parameter is changed, restart **mbatchd** and **sbatchd**.
- **LSB_KRB_RENEW_MARGIN**: specifies how long **krbrenewd** and root **sbatchd** have to renew a Ticket Granting Ticket (TGT) before it expires. If this parameter is changed, restart **mbatchd** and **sbatchd** or reconfigure **mbatchd** and **sbatchd** to have it take effect.
- **LSB_KRB_TGT_FWD**: use this parameter to control the user Ticket Granting Ticket (TGT) forwarding feature. When set to Y, user TGT is forwarded from the submission host to the execution host. **mbatchd** or root **sbatchd** do the required renewing along the way.
- **LSB_KRB_TGT_DIR**: specifies a directory in which Ticket Granting Ticket (TGT) for a running job is stored. Each job or task will have its environment variable KRB5CCNAME pointing to the TGT file. Please note that this parameter controls the TGT location for running jobs, not for pending jobs on the mbatchd side. For pending jobs, TGTs are stored in the **mbatchd** info directory, along with their job log file.
- **LSB_MAX_JOB_DISPATCH_PER_SESSION**: Defines the maximum number of jobs that **mbatchd** can dispatch during one job scheduling session.
- **LSB_MAX_PROBE_SBD**: Specifies the maximum number of **sbatchd** instances that can be polled by **mbatchd** in the interval **MBD_SLEEP_TIME**/10. Use this parameter in large clusters to reduce the time it takes for **mbatchd** to probe all **sbatchd**s. The value of **LSB_MAX_PACK_JOBS** cannot be greater than the number of hosts in the cluster. If it is, **mbatchd** adjusts the value of **LSB_MAX_PACK_JOBS** to be same as the number of hosts. The default value for **LSB_MAX_PROBE_SBD** is 20.
- **LSB_POSTEXEC_SEND_MAIL**: enable this parameter to have LSF send an email to the user that provides the details of post execution, if any. This includes any applicable output.

- **LSB_SYNC_HOST_STAT_FROM_LIM** is now enabled by default (previously, this was disabled by default), so there is no need to configure it in the configuration file.
- **LSF_HPC_EXTENSIONS**: **HOST_RUSAGE** is enabled by default in LSF 9.1. If you are running large parallel jobs, you should also set **LSF_HPC_EXTENSIONS="CUMULATIVE_RUSAGE HOST_RUSAGE"**.
- **LSF_PIM_LINUX_ENHANCE**: Support for exact memory usage tracking for shared memory segments reporting has been extended from Linux kernel versions 2.6.25 and above to Linux kernel version 2.6.14 and above
- **LSF_LSRUN_CWD_USE_TMP=Y**: When the current working directory does not exist on an execution host, by default, jobs submitted by **lsrun** and **lsgrun** will fail because of lack of current directory. LSF_LSRUN_CWD_USE_TMP=Y allows the job try to run under `/tmp`. The LSF_LSRUN_CWD_USE_TMP is supported on Unix and Linux.
- **LSF_DYNAMIC_HOST_KEEP=Y**: Allows LIM to keep dynamic host information in memory when dynamic hosts become unavailable for longer than the period specified by LSF_DYNAMIC_HOST_TIMEOUT. This parameter also disables automatic **mbatchd** reconfiguration triggered by dynamic hosts becoming unavailable for longer than LSF_DYNAMIC_HOST_TIMEOUT. Valid values for LSF_DYNAMIC_HOST_KEEP are Y or N. The default value is N.
- **LSB_TSJOBS_HELPER_HOSTS**: Lists the Windows Terminal Services job helper hosts.
- **LSB_TSJOBS_HELPER_PORT**: Specifies a service port to use for communication with TSJobHelper. TSJobHelper uses this port to communicate with the helper hosts.
- **LSB_TSJOBS_HELPER_TIMEOUT**: The maximum time out that the local TSJobHelper service waits for a helper host to reply. After time out, the local service tries the next helper host in the LSB_TSJOBS_HELPER_HOSTS list.
- **LSF_LINUX_CGROUP_ACCT**: Tracks processes based on CPU and memory accounting for Linux systems that support cgroup's memory and cpuacct subsystems.
- **LSF_PROCESS_TRACKING**: Tracks processes based on job control functions such as termination, suspension, resume and other signaling, on Linux systems which support cgroups' freezer subsystem.

New and changed accounting and job event fields

lsb.acct and lsb.events

New records and fields have been added for LSF 9.1. See the *IBM Platform LSF Configuration Reference* for more detail.

LSF integration with Cray Linux

Enhancements and new features for this integration include:

- Support for heterogeneous compute nodes (number of slots and memory)
- Compute nodes become LSF server hosts and can be viewed and controlled by standard LSF commands, LSF configuration and scheduling policies.
- Users and administrators can use standard LSF commands to view and choose which compute nodes to use.
- Support for creating CRAY Linux reservations, including GPU
- CSA accounting
- Scheduling performance enhancements
- Checkpoint/restart CCM job support
- Full interactive shell job support
- Support for running jobs on login nodes without creating reservations

This integration applies to:

- LSF release 8.0 or later
- LSF Standard (LSF must not be running in Express mode)
- LSF integration with Cray Linux Environment 4.0 or later
- The BASIL 1.2 protocol

Important: For non-Cray operating systems to interoperate with a Cray Linux machine in the same cluster, the non-Cray packages require download and installation of specific patches. These patches are created on demand. Conatct IBM Support to obtain these packages.

Download and installation

1. Download the installation package and the distribution tar file for the LSF/Cray Linux integration:
   - `lsf9.1.1_lnx26-lib23-x64-cray.tar.Z`
   - `lsf9.1.1_lsfinstall.tar.Z`

     This is the standard installation package. Use this package in a heterogeneous cluster with a mix of Cray and non-Cray systems other than x86-64.
   - `lsf9.1_lsfinstall_linux_x86_64.tar.Z`

     Use this smaller installation package in a homogeneous x86-64 cluster. If you add other non x86-64 hosts you must use the standard installation package.
2. Before running the installation, confirm the Cray Linux system is working:
   - On CLE 4.0 or above, confirm the existence of `/opt/cray/rca/default/bin/rca-helper`, `/etc/xthostname`, and `/etc/opt/cray/sdb/node_classes`.

     Otherwise, confirm that the **xtuname** and **xthostname** commands exist and are in the $PATH environment variable.
   - Confirm that all compute PEs are in batch mode. If not, switch all compute PEs to batch mode and restart ALPS services on the boot node:
     - `xtprocadmin -k m batch`
     - `/etc/init.d/alps restart` (optional)
     - `apstat -rn` (optional)
3. Follow the standard LSF installation procedure to install LSF on the boot nodes:
   a. Run the **xtopview** command to switch to a shared root file system.
   b. Edit the LSF `install.config` file:
      - LSF_TOP="/software/lsf"
      - LSF_CLUSTER_NAME=*crayxt machine name*
      - LSF_MASTER_LIST=*master host candidates* # login nodes or service nodes
      - EGO_DAEMON_CONTROL="N"
      - ENABLE_DYNAMIC_HOSTS="N"
      - LSF_ADD_SERVERS=*service or login nodes*
      - ENABLE_HPC_CONFIG="Y"

      LSF_MASTER_LIST and LSF_ADD_SERVERS should only include login nodes or service nodes.

      The **startup** and **shutdown** scripts for LSF daemons can be found in `LSF_SERVERDIR/lsf_daemons`.
4. As an LSF administrator:
   a. Add the following to /opt/xt-boot/default/etc/serv_cmd:

- service_cmd_info='LSF-HPC',service_num=XXX,heartbeat=null
- start_cmd='*$LSF_SERVERDIR*/lsf_daemons start'
- stop_cmd='*$LSF_SERVERDIR*/lsf_daemons stop'
- restart_cmd='*$LSF_SERVERDIR*/lsf_daemons restart'
- fail_cmd='*$LSF_SERVERDIR*/lsf_daemons stop'

b. Create a service command:

```
xtservcmd2db -f /opt/xt-boot/default/etc/serv_cmd
```

c. Assign the LSF-HPC service to serv_cmd: xtservconfig -c login add LSF-HPC.

d. Exit **xtopview** and access a login node:

1) Make sure /ufs is shared among all login and service nodes and root and LSF administrators have write permission.

2) Set up subdirectories under /ufs the same as /opt/xt-lsfhpc/log and /opt/xt-lsfhpc/work.

3) Make sure the directory ownership and permission mode are reserved (you can use the **cp -r** command), and that root and LSF administrators have write permission to the subdirectories under /ufs/lsfhpc.

5. Use the **module** command to set the LSF environment variables:

```
module load xt-lsfhpc
```

6. From a login node, run **LSF_BINDIR/genVnodeConf**.

This command generates a list of compute nodes in BATCH mode. You can add the compute nodes to the HOST section in LSF_ENVDIR/lsf.cluster.*cluster_name*. For example:

```
HOSTNAME  model   type   server    r1m  mem  swp  RESOURCES
hosta        !     !       1       3.5   ()  ()   (craylinux vnode)
hostb        !     !       1       3.5   ()  ()   (craylinux vnode)
hostc        !     !       1       3.5   ()  ()   (craylinux vnode)
...
hostx        !     !       1       3.5   ()  ()   (craylinux vnode)
```

Configuration

1. Configure LSF_ENVDIR/hosts.

Make sure the IP addresses of compute nodes do not conflict with any IP being used.

```
cat $LSF_ENVDIR/hosts
192.0.2.1 hosta.example.com hosta
192.0.2.24     hostb        c0-0c1s0n3      sdb001  sdb002
192.0.2.22     hostc        c0-0c1s1n0      login   login1
192.0.2.20     hostd        c0-0c1s1n3
...
192.0.2.5      hostz        c0-0c1s7n3
192.0.2.2      hostj        sdb-p2 syslog ufs
```

2. Modify LSF_ENVDIR/lsf.conf:

LSB_SHAREDIR=/ufs/lsfhpc/work # A shared file system that is accessible by root and LSF admin on both Master hosts and Cray Linux login/service nodes.

LSF_LOGDIR=/ufs/lsfhpc/log # A shared file system that is accessible by root and LSF admin on both Master hosts and Cray Linux login/service nodes.

LSF_LIVE_CONFDIR=/ufs/lsfhpc/work/<cluster_name>/live_confdir # A shared file system that is accessible by root and LSF admin on both Master hosts and Cray Linux login/service nodes.

LSB_RLA_PORT=21787

LSB_SHORT_HOSTLIST=1

LSF_ENABLE_EXTSCHEDULER=Y

LSB_SUB_COMMANDNAME=Y

LSF_CRAY_PS_CLIENT=/usr/bin/apbasil

LSF_LIMSIM_PLUGIN="liblimsim_craylinux"

LSF_CRAYLINUX_FRONT_NODES="nid00060 nid00062" # A list of Cray Linux login/service nodes with LSF daemons started and running.

LSF_CRAYLINUX_FRONT_NODES_POLL_INTERVAL=120 # Interval for Master Lim polling RLA to query computer node status and configuration information. Default value is 120s. Any value less than 120s will be reset to default.

LSB_MIG2PEND=1

3. Modify `LSF_ENVDIR/lsbatch/cluster_name/configdir/lsb.modules`.

   Make sure schmod_craylinux is the last plugin module and schmod_crayxt3 is commented out. If you do not use MultiCluster or the LSF cpuset integration, comment them both out as well.

4. Modify `LSF_ENVDIR/lsbatch/cluster_name/configdir/lsb.hosts`.

   Make sure Cray Linux login/service nodes that are also LSF server hosts have a large number set in the MXJ column (the value should be larger than the total number of PEs). For example:

```
Begin Host
      HOST_NAME   MXJ     r1m  pg  ls  tmp  DISPATCH_WINDOW  # Keywords
      nid00060    9999    ()   ()  ()  ()   ()                # Example
      nid00062    9999    ()   ()  ()  ()   ()                # Example
      default      !      ()   ()  ()  ()   ()                # Example
End Host
```

5. Modify `LSF_ENVDIR/lsbatch/cluster_name/configdir/lsb.queues`.
   - JOB_CONTROLS and RERUNNABLE are required.
   - Comment out all loadSched and loadStop lines.
   - DEF_EXTSCHED and MANDATORY_EXTSCHED are optional.
   - PRE_EXEC and POST_EXEC are required to run CCM jobs.

6. Modify `LSF_ENVDIR/lsf.shared`.

   Make sure the following boolean resources are defined in the RESOURCE section:

```
vnode       Boolean   ()   ()   (sim node)
gpu         Boolean   ()   ()   (gpu)
frontnode   Boolean   ()   ()   (login/service node)
craylinux   Boolean   ()   ()   (Cray XT/XE MPI)
```

7. By default, Comprehensive System Accounting (CSA) is enabled. If CSA is not installed in your environment, you must disable CSA by setting LSF_ENABLE_CSA=N in `lsf.conf`.

8. Use the service command to start and stop the LSF services as needed:

```
service LSF-HPC start

service LSF-HPC stop
```

Submit and Run Parallel Jobs

Before you submit jobs to the cluster, note that CLE4.0 does not support multiple jobs running on one compute node. All ALPS reservations created by LSF will have the "mode=EXCLUSIVE" attribute. You can define a limit to make sure LSF does not dispatch jobs to compute nodes where a job has been running.

Modify `LSF_ENVDIR/lsbatch/cluster_name/configdir/lsb.resources`:

```
Begin Limit
  NAME     = COMPUTE_NODES_LIMIT
  USERS    = all
  PER_HOST = list_of_compute_nodes #This limit applies to compute nodes only.
  JOBS     = 1
End Limit
```

There are other ways in LSF to enforce this limitation for ALPS:

- To submit a job that requires Cray Linux reservations (for example, an aprun job or a CCM job), compound resource requirements must be used:

  ```
  bsub -extsched "CRAYLINUX[]" -R "1*{select[craylinux && \!vnode]} +
  n*{select[vnode && craylinux] span[ptile=q*p]}" aprun -n y -d p -N q
  a.out
  ```

  n must be greater than or equal to MAX(y*p, p*q) (the default of y p q is 1).
- To submit a job that requires Cray Linux reservations with GPU (for example, an **aprun** job or a CCM job):

  ```
  bsub -extsched "CRAYLINUX[GPU]" -R "1*{select[craylinux && \!vnode]} +
  n*{select[vnode && craylinux && gpu ] span[ptile=q*p] rusage[jobcnt=1]}"
  aprun -n y -d p -N q a.out
  ```

  n must be greater than or equal to MAX(y*p, p*q) (the default of y p q is 1).
- To submit a job that runs on Cray service/login nodes without creating Cray Linux reservations:

  ```
  bsub -R "select[craylinux && frontnodes]" cmd_name
  ```
- Jobs with incorrect resource requirements will be detected and put in pending state. For example, jobs asking for vnode but without "CRAYLINUX[]" will show a pending reason: `Job cannot run on hosts with "vnode"`.
- To create an advance reservation, complete the following steps:
  1. Create an advance reservation on compute nodes (hosts with craylinux && vnode).
  2. Add slots on the frontend nodes (host with craylinux && \!vnode).
  3. Submit jobs and specify the advance reservation for the job as usual.

Command output

The **bjobs**, **bhist**, and **bacct** commands display reservation_id under additionalInfo.

Assumptions and Limitations

After the integration has been installed and configured, advance reservation, preemption, and reservation scheduling policies are supported with the following limitations:

- Not all scheduling policies behave the same way or automatically support the same things as standard LSF. ALPS in CLE4.0 only supports node exclusive reservations (no two jobs can run on the same node). Resource reservations (slot and resource) in LSF are impacted as jobs that reserved slots may not be able to run due to this ALPS limitation.
- Only one Cray Linux machine per cluster is allowed.

Known Issues

- CPU and memory affinity scheduling has the following limitations.

  - When reservation is enabled, affinity reservation allocations appear as part of the allocated resources in **bhosts -aff**

    Jobs that are submitted with a `membind=localprefer` binding policy may overcommit the memory of the NUMA node they are allocated to

    bhosts -aff output may occasionally show the total allocated memory on the NUMA nodes of a host as exceeding the maximum memory of the host, this is because the *reservations* that show in **bhosts -aff** overcommit the NUMA node. However, LSF will never allow the allocation of *running* jobs on a host to exceed the maximum memory of a host.
  - When reservation is enabled, and an affinity job requests enough resources to consume an entire node in the host topology. (for example, enough cores to consume an entire socket), LSF will not reserve the socket for the job if there are any jobs running on its cores. In a situation when there are always smaller jobs running consuming cores, then larger jobs that require entire sockets will not be able to reserve resources. The workaround is to require that all jobs have estimated runtimes, and to use time-based reservation.
-
- **bmod** cannot change the memory requirement for a running job if a MEM general resource limit is defined for the user in `lsb.resources`.
- Application checkpointing is not supported on 64-bit Windows 7.
- MacOS X is supported only as LSF slave hosts. MacOS X hosts cannot be LSF master or master candidate hosts.
- LSF 8.3 **blimits** does not work with 9.1 binaries.
- For GSLA, a job may pend or receive fewer slots than expected when you ask for a range of slots.

Limitations

- Parallel restart cannot be used if the **mbatchd** is configured to use duplicate event logging (LSB_LOCALDIR is configured in `lsf.conf`).
- Processor number is not detected correctly on POWER7 Linux machines
- NUMA topology may be incorrect after bringing cores offline.

Bugs fixed since December 2012 (Platform LSF 9.1)

Bugs fixed in the March 2013 release (LSF 9.1.1) since December 2012 are listed in the document *Fixed Bugs for Platform LSF 9.1.1*.

Bugs fixed since April 2012 (Platform LSF 8.3)

Bugs fixed in the December 2012 release (LSF 9.1) since April 2012 until November 22, 2012 are listed in the document *Fixed Bugs for Platform LSF 9.1*.

Platform LSF 9.1.1 Installation Packages

The LSF installation consists of the following packages and files:

- Installer packages:
  - `lsf9.1.1_lsfinstall.tar.Z`

    This is the standard installation package. Use this package in a heterogeneous cluster with a mix of systems other than x86-64.
  - `lsf9.1.1_lsfinstall_linux_x86_64.tar.Z`

Use this smaller installation package in a homogeneous x86-64 clsuter. If you add other non x86-64 hosts you must use the standard installation package.

Use this installer package to install LSF 9.1.1 from a Linux host or a supported platform that already has JRE 1.4+ installed.

The same installer packages are used for LSF Express Edition, LSF Standard Edition, and LSF Advanced Edition.

Note: Allow approximately 1 GB for `lsf9.1.1_lsfinstall.tar.Z`, and approximately 100 MB for `lsf9.1.1_lsfinstall_linux_x86_64.tar.Z`.

- Product packages
- Documentation packages:
  - `lsf9.1.1_documentation.tar.Z`
  - `lsf9.1.1_documentation.zip`
- Entitlement configuration files:
  - LSF Standard Edition: `platform_lsf_std_entitlement.dat`
  - LSF Express Edition: `platform_lsf_exp_entitlement.dat`.
  - LSF Advanced Edition: `platform_lsf_adv_entitlement.dat`.

Separate product packages are available for the following operating systems:

Operating system	Product package
AIX 6 and 7 on POWER	lsf9.1.1_aix-64.tar.Z
HP UX B.11.31 on PA-RISC	lsf9.1.1_hppa11i-64.tar.Z
HP UX B.11.31 on IA64	lsf9.1.1_hpuxia64.tar.Z
Solaris 10 and 11 on Sparc	lsf9.1.1_sparc-sol10-64.tar.Z
Solaris 10 and 11 on x86-64	lsf9.1.1_x86-64-sol10.tar.Z
Linux on x86-64 Kernel 2.6 and 3.*x*	lsf9.1.1_linux2.6-glibc2.3-x86_64.tar.Z
Linux on POWER Kernel 2.6 and 3.*x*	lsf9.1.1_linux2.6-glibc2.3-ppc64.tar.Z
Windows 2003/2008/2012/XP/7/8 32-bit	lsf9.1.1_win32.msi
Windows 2003/2008/2012/XP/7/8 64-bit	lsf9.1.1_win-x64.msi
Mac OS 10.x	lsf9.1.1_macosx.tar.Z
Cray Linux	lsf9.1.1_lnx26-lib23-x64-cray.tar.Z

Download the Platform LSF 9.1.1 Distribution Packages

Download the LSF distribution and documentation packages from the IBM Support Portal:

www.ibm.com/support

Install Platform LSF 9.1.1

Installing Platform LSF involves the following steps:

1. Get an LSF entitlement file
   - `platform_lsf_exp_entitlement.dat` for LSF Express Edition
   - `platform_lsf_std_entitlement.dat` for LSF Standard Edition

- `platform_lsf_adv_entitlement.dat` for LSF Advanced Edition

2. Run the installation programs.

Run the UNIX and Linux installation

Use the **lsfinstall** installation program to install a new LSF 9.1.1 cluster, or upgrade from an earlier LSF version.

See *Installing IBM Platform LSF on UNIX and Linux* for new cluster installation steps.

See the *IBM Platform LSF Command Reference* for detailed information about **lsfinstall** and its options.

Run the Windows installation

Platform LSF on Windows Server 2003, Windows Server 2008, Windows XP, Windows Vista, Windows 7, Windows 8, Windows HPC Server 2008, and Windows Server 2012 is distributed in the following packages:

- `lsf9.1.1_win32.msi`
- `lsf9.1.1_win-x64.msi`

See *Installing Platform LSF on Windows* for new cluster installation steps.

Learn About Platform LSF 9.1.1

Information about Platform LSF is available from the following sources:

- World Wide Web
- Platform LSF documentation

World Wide Web

Information about Platform LSF 9.1.1 is available in the LSF area of the IBM Support Portal:

www.ibm.com/support

For the latest information about Platform LSF9.1.1, visit the IBM Platform web site:

www-03.ibm.com/systems/technicalcomputing/platformcomputing/index.html

Platform LSF documentation

The Platform LSF Documentation page is your entry point for all LSF documentation. If you have installed IBM Platform Application Center, you can access and search the LSF documentation through the Help link in the user interface.

Platform LSF documentation is also available in PDF format by searching the IBM Publications Center:

`www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss`

Get Technical Support

Contact IBM

Contact IBM or your LSF vendor for technical support.

Or go to the IBM Support Portal: www.ibm.com/support

We'd like to hear from you

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

Mail

IBM Canada Ltd.
Information Development
E6/5F9/3600 /MKM
3600 Steeles Avenue East
Markham
Ontario Canada L3R 9Z7

Be sure to include the following information:

- The title and order number of the publication you are commenting on
- The version of the product you are using
- The format of the manual (HTML or PDF)
- If applicable, the specific location of the information about which you have comments (for example, a page number)

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

© Copyright IBM Corp. 1992, 2013

sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Intellectual Property Law
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application

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.

Printed in USA

GI13-3413-01