IBM Spectrum LSF for SAS
Version 10 Release 1

Release Notes

IBM Spectrum LSF for SAS
Version 10 Release 1

Release Notes

Note
Before using this information and the product it supports, read the information in "Notices" on page 61.

This edition applies to version 10, release 1 of IBM Spectrum LSF (product numbers 5725G82 and 5725L25) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Log in to IBM Knowledge Center with your *IBMid*, and add your comments and feedback to any topic.

Contents

Release notes for IBM Spectrum LSF Version 10.1

Read this document to find out what's new in IBM Spectrum LSF Version 10.1. Learn about product updates, compatibility issues, limitations, known problems, and bugs fixed in the current release. Find LSF product documentation and other information about IBM Spectrum Computing products.

Last modified: 5 June 2018

What's new in IBM Spectrum LSF Version 10.1 Fix Pack 6

The following topics summarize the new and changed behavior in LSF 10.1 Fix Pack 6.

Release date: June 2018

GPU enhancements

The following enhancements affect LSF GPU support.

GPU autoconfiguration

Enabling GPU detection for LSF is now available with automatic configuration. To enable automatic GPU configuration, configure `LSF_GPU_AUTOCONFIG=Y` in the `lsf.conf` file.

When enabled, the **`lsload -gpu`**, **`lsload -gpuload`**, and **`lshosts -gpu`** commands will show host-based or GPU-based resource metrics for monitoring.

Specify additional GPU resource requirements

LSF now allows you to request additional GPU resource requirements to allow you to further refine the GPU resources that are allocated to your jobs. The existing **`bsub -gpu`** command option, **`LSB_GPU_REQ`** parameter in the `lsf.conf` file, and the **`GPU_REQ`** parameter in the `lsb.queues` and `lsb.applications` files now have additional GPU options to make the following requests:

- The `gmodel` option requests GPUs with a specific brand name, model number, or total GPU memory.
- The `gtile` option specifies the number of GPUs to use per socket.
- The `gmem` option reserves the specified amount of memory on each GPU that the job requires.
- The `nvlink` option requests GPUs with NVLink connections.

You can also use these options in the **`bsub -R`** command option or **`RES_REQ`** parameter in the `lsb.queues` and `lsb.applications` files for complex GPU resource requirements, such as for compound or alternative resource requirements. Use the `gtile` option in the `span[]` string and the other options (`gmodel`, `gmem`, and `nvlink`) in the `rusage[]` string as constraints on the `ngpus_physical` resource.

To specify these new GPU options, specify `LSB_GPU_NEW_SYNTAX=extend` in the `lsf.conf` file.

See more information on submitting and monitoring GPU resources in *Administering IBM Spectrum Cluster Foundation*.

Data collection

The following new features affect IBM Spectrum LSF data collection.

IBM Spectrum Scale disk I/O accounting using Elasticsearch

LSF now uses IBM Spectrum LSF Explorer (LSF Explorer) to collect IBM Spectrum Scale disk I/O accounting data which, when combined with LSF job information, allows LSF to provide job-level IBM Spectrum Scale I/O statistics. To use this feature, LSF Explorer must be deployed in your LSF cluster, and LSF must be using IBM Spectrum Scale as the file system. To enable IBM Spectrum Scale disk I/O accounting, configure `LSF_QUERY_ES_FUNCTIONS="gpfsio"` (or `LSF_QUERY_ES_FUNCTIONS="all"`) and `LSF_QUERY_ES_SERVERS="`*`ip`*`:`*`port`*`"` in the `lsf.conf` file.

Use the following commands to display IBM Spectrum Scale disk I/O accounting information:

- **`bacct -l`** displays the total number of read/write bytes of all storage pools on IBM Spectrum Scale.
- **`bjobs -l`** displays the accumulated job disk usage (I/O) data on IBM Spectrum Scale.
- **`bjobs -o "gpfsio"`** displays the job-level disk usage (I/O) data on IBM Spectrum Scale.

Resource Connector enhancements

The following enhancements affect LSF Resource Connector.

LSF resource connector auditing

With this release, LSF will log resource connector VM events along with usage information into a new file **rc.audit.x** (one log entry per line in JSON format). The purpose of the rc.audit.x log file is to provide evidence to support auditing and usage accounting as supplementary data to third party cloud provider logs. The information is readable by the end user as text and is hash protected for security.

LSF also provides a new command-line tool **rclogsvalidate** to validate the logs described above. If the audit file is tampered with, the tool will identify the line which was modified and incorrect.

New parameters have been added to LSF in the lsf.conf configuration file:

- **LSF_ RC_AUDIT_LOG**: If set to Y, enables the resource connector auditor to generate log files.
- **RC_MAX_AUDIT_LOG_SIZE**: An integer to determine the maximum size of the rc.audit.x log file, in MB.
- **RC_MAX_AUDIT_LOG_KEEP_TIME**: An integer that specifies the amount of time that the resource connector audit logs are kept, in months.

Resource connector template prioritizing

In 10.1 Fix Pack 6 Resource Connector prioritize templates.

The ability to set priorities is now provided in the Resource Connector template. LSF will use higher priority templates first (for example, less expensive templates should be assigned higher priorities).

LSF sorts candidate template hosts by template name. However, an administrator might want to sort them by priority, so LSF favors one template to the other. The “Priority” attribute has been added.:

```
{
      "Name": "T2",
      "MaxNumber": "2",
      "Attributes":
      {
        "type": ["String", "X86_64"],
        "ncpus": ["Numeric", "1"],
        "mem": ["Numeric", "512"],
        "template": ["String", "T2"],
        "ostkhost": ["Boolean", "1"]
      },
      "Image": "LSF10.1.0.3_OSTK_SLAVE_VM",
      "Flavor": "t2.nano",
      "UserData": "template=T2",
      "Priority": "10"
    }
```

Note: The example above is for a template in openStack. Other templates may not contain all attributes.

The default value of Priority is “0”, which means the lowest priority. If template hosts have the same priority, LSF sorts them by template name.

Support for a dedicated instance of AWS

One new parameter is added to the Resource Connector template to support a dedicated instance of AWS.

If you do not have a placement group in your AWS account, you must at least insert a placement group with a blank name inside quotation marks, because this is required to specify the tenancy. If you have a placement group, specify the placement group name inside the quotation marks. For example, `"placementGroupName": "",` or `"placementGroupName": "hostgroupA",`.

The values for tenancy can be "default", "dedicated", and "host". However, LSF currently only supports "default" and "dedicated".

The above can be applied for both on-demand and spot instances of AWS.

Full example the template file is as follows:

```
{
    "templates": [
         {
            "templateId": "aws-vm-0",
            "maxNumber": 5,
            "attributes": {
                "type": ["String", "X86_64"],
                "ncores": ["Numeric", "1"],
                "ncpus": ["Numeric", "1"],
                "mem": ["Numeric", "512"],
                "awshost": ["Boolean", "1"],
                "zone": ["String", "us_west_2d"]
            },
            "imageId": "ami-0db70175",
            "subnetId": "subnet-cc0248ba",
            "vmType": "c4.xlarge",
            "keyName": "martin",
            "securityGroupIds": ["sg-b35182ca"],
            "instanceTags": "Name=aws-vm-0",
            "ebsOptimized" : false,
```

```
            "placementGroupName": "",
            "tenancy": "dedicated",
            "userData": "zone=us_west_2d"        }
}
```

HTTP proxy server capability for LSF Resource connector

This feature is useful for customers with strict security requirements. It allows for the use of an HTTP proxy server for endpoint access.

Note: For this release, this feature is enabled only for AWS.

This feature introduces the parameter "scriptOption" for the provider. For example:

```
{
    "providers":[
        {
            "name": "aws1",
            "type": "awsProv",
            "confPath": "resource_connector/aws",
            "scriptPath": "resource_connector/aws",
           "scriptOption": "-Dhttps.proxyHost=10.115.206.146 -Dhttps.proxyPort=8888"
        }
    ]
}
```

The value of `scriptOption` can be any string and is not verified by LSF.

LSF sets the environment variable `SCRIPT_OPTIONS` when launching the scripts. For AWS plugins, the information is passed to java through syntax like the following:

```
java $SCRIPT_OPTIONS -Daws-home-dir=$homeDir -jar $homeDir/lib/AwsTool.jar --getAvailableMachines $h
```

Create EBS-Optimized instances

Creating instances with EBS-Optimized enabled is introduced in this release to archive better performance in cloud storage.

The EBS-Optimized attribute has been added to the Resource Connector template. The AWS provider plugin passes the information to AWS when creating the instance. Only high-end instance types support this attribute. The Resource Connector provider plugin will not check if the instance type is supported.

The "ebsOptimized" field in the Resource Connector template is a boolean value (either true or false). The default value is false. Specify the appropriate vmType that supports ebs_optimized (consult AWS documentation).

```
{
    "templates": [
        {
            "templateId": "Template-VM-1",
            "maxNumber": 4,
            "attributes": {
                "type": ["String", "X86_64"],
                "ncores": ["Numeric", "1"],
                "ncpus": ["Numeric", "1"],
                "mem": ["Numeric", "1024"],
                "awshost1": ["Boolean", "1"]
            },
            "imageId": "ami-40a8cb20",
           "vmType": "m4.large",
            "subnetId": "subnet-cc0248ba",
            "keyName": "martin",
            "securityGroupIds": ["sg-b35182ca"],
            "instanceTags" : "group=project1",
            "ebsOptimized" : true,
```

```
            "userData": "zone=us_west_2a"
        }
    ]
}
```

Resource connector Policy Enhancement

Enhancements have been made for administration of Resource Connector policies:

- A clusterwide parameter **RC_MAX_REQUESTS** has been introduced in the lsb.params file to control the maximum number of new instances that can be required or requested.

  After adding allocated usable hosts in previous sessions, LSF generates total demand requirement. An internal policy entry is created as below:

```
{
      "Name": "__RC_MAX_REQUESTS",
      "Consumer":
       {
        "rcAccount": ["all"],
        "templateName": ["all"],
        "provider": ["all"]
       },
      "StepValue": "$val:0"
    }
```

- The parameter **LSB_RC_UPDATE_INTERVAL** controls how frequent LSF starts demand evaluation. Combining with the new parameter, it plays a cluster wide "step" to control the speed of cluster grow.

Resource management

The following new features affect resource management and allocation.

Running LSF jobs with IBM Cluster Systems Manager

LSF now allows you to run jobs with IBM Cluster Systems Manager (CSM).

The CSM integration allows you to run LSF jobs with CSM features.

See more information on LSF with Cluster Systems Manager in *Administering IBM Spectrum LSF*.

Direct data staging

LSF now allows you to run direct data staging jobs, which uses a burst buffer (for example, IBM CAST burst buffer) instead of the cache to stage in and stage out data for data jobs.

Use the CSM integration to configure LSF to run burst buffer data staging jobs.

See more information on burst bugger data staging jobs in *Administering IBM Spectrum LSF*.

Job scheduling and execution

The following new features affect LSF job scheduling and execution.

Plan-based scheduling and reservations

When enabled, LSF's plan-based scheduling makes allocation plans for jobs based on anticipated future cluster states. LSF reserves resources as needed in order to carry out its plan. This helps to avoid starvation of jobs with special resource requirements.

Plan-based scheduling and reservations addresses a number of issues with the older reservation features in LSF. For example:

- It ensures that reserved resources can really be used by the reserving jobs
- It has better job start-time prediction for reserving jobs, and thus better backfill decisions

Plan-based scheduling aims to replace legacy LSF reservation policies. When **ALLOCATION_PLANNER** is enabled in the `lsb.params` configuration file, then parameters related to the old reservation features (that is **SLOT_RESERVE** and **RESOURCE_RESERVE** in `lsb.queues`), are ignored with a warning.

Automatically extend job run limits

You can now configure the LSF allocation planner to extend the run limit for jobs when the resources that are occupied by the job are not needed by other jobs in queues with the same or higher priority. The allocation planner looks at job plans to determine if there are any other jobs that require the current job's resources.

Enable extendable run limits for jobs submitted to a queue by specifying the **EXTENDABLE_RUNLIMIT** parameter in the `lsb.queues` file. Since the allocation planner decides whether the extend the run limit of jobs, you must also enable plan-based scheduling by enabling the **ALLOCATION_PLANNER** parameter in the `lsb.params` file.

See more information on configuring extendable run limits in *Administering IBM Spectrum LSF*.

Default epsub executable files

Similar to **esub** programs, LSF now allows you to define a default **epsub** program that runs even if you do not define mandatory **epsub** programs with the **LSB_ESUB_METHOD** parameter in the `lsf.conf` file. To define a default **epsub** program, create an executable file named `epsub` (with no application name in the file name) in the `LSF_SERVERDIR` directory.

After the job is submitted, LSF runs the default **epsub** executable file if it exists in the `LSF_SERVERDIR` directory, followed by any mandatory **epsub** executable files that are defined by **LSB_ESUB_METHOD**, followed by the **epsub** executable files that are specified by the `-a` option.

See more information on external job submission and execution controls in *Administering IBM Spectrum LSF*

Restrict users and user groups from forwarding jobs to remote clusters

You can now specify a list of users or user groups that can forward jobs to remote clusters when using the LSF multicluster capability. This allows you to prevent jobs from certain users or user groups from being forwarded to an execution cluster, and to set limits on the submission cluster.

These limits are defined at the queue level in LSF. For jobs that are intended to be forwarded to a remote cluster, users must submit these jobs to queues that have the **SNDJOBS_TO** parameter configured in the `lsb.queues` file. To restrict these queues to specific users or user groups, define the **FWD_USERS** parameter in the `lsb.queues` file for these queues.

See more information on multicluster queues in *Using IBM Spectrum LSF multicluster capability*.

Advance reservations now support the "same" section in resource requirement strings

When using the **brsvadd -R** and **brsvmod -R** options to specify resource requirements for advance reservations, the same string now takes effect, in addition to the select string. Previous versions of LSF only allowed the select string to take effect.

This addition allows you to select hosts with the same resources for your advance reservation.

See more information on specifying resource requirements (and the same string) in *Administering IBM Spectrum LSF*.

Priority factors for absolute priority scheduling

You can now set additional priority factors for LSF to calculate the job priority for absolute priority scheduling (APS). These additional priority factors allow you to modify the priority for the application profile, submission user, or user group, which are all used as factors in the APS calculation. You can also view the APS and fairshare user priority values for pending jobs.

To set the priority factor for an application profile, define the **PRIORITY** parameter in the lsb.applications file. To set the priority factor for a user or user group, define the **PRIORITY** parameter in the **User** or **UserGroup** section of the lsb.users file.

The new **bjobs -prio** option displays the APS and fairshare user priority values for all pending jobs. In addition, the **busers** and **bugroup** commands display the APS priority factor for the specified users or user groups.

See more information on absolute priority scheduling in *Administering IBM Spectrum LSF*.

Job dispatch limits for users, user groups, and queues

You can now set limits on the maximum number of jobs that are dispatched in a scheduling cycle for users, user groups, and queues. This allows you to control the number of jobs, by user, user group, or queue, that are dispatched for execution. If the number of dispatched jobs reaches this limit, other pending jobs that belong to that user, user group, or queue that might have dispatched will remain pending for this scheduling cycle.

To set or update the job dispatch limit, run the **bconf** command on the limit object (that is, run bconf *action_type* limit=*limit_name*) to define the **JOBS_PER_SCHED_CYCLE** parameter for the specific limit. You can only set job dispatch limits if the limit consumer types are **USERS**, **PER_USER**, **QUEUES**, or **PER_QUEUE**.

For example, bconf update limit=L1 "JOBS_PER_SCHED_CYCLE=10"

You can also define the job dispatch limit by defining the **JOBS_PER_SCHED_CYCLE** parameter in the **Limit** section of the lsb.resources file.

See more information on configuring resource allocation limits in *Administering IBM Spectrum LSF*.

Command output formatting

The following enhancements affect LSF command output formatting.

blimits -a option shows all resource limits

The new **blimits -a** command option shows all resource allocation limits, even if they are not being applied to running jobs. Normally, running the **blimits** command with no options displays only resource allocation limits that are being applied to running jobs.

Related concepts:

Display resource allocation limits

View information about resource allocation limits

Related reference:

blimits -a command option

Use bread -w to show messages and attached data files in wide format

LSF allows you to read messages and attached data files from a job in wide format with the new **bread -w** command option. The wide format displays information without truncating fields.

See more information on the **bread -w** command option in *IBM Spectrum LSF Command Reference*.

Other changes to IBM Spectrum LSF

The following changes affect other aspects of LSF behavior.

lsportcheck utility

A new **lsportcheck** utility has been added to LSF. This utility can be used to check the required ports for LSF and include detailed information, whether it is being used or not.

The lsportcheck utility only checks ports on the host for availability. It discovers the ports by reading the configuration files. If the line is commented out or if there is no value, it will use the default values.

The lsportcheck utility must be executed by the root user, since the tool uses 'netstat' and needs root to get the complete information on the ports of the OS.

Before running this tool, you must source the profile or set the environment variable LSF_TOP.

The utility is installed at <LSF_TOP>/<VERSION>/<PLATFORM>/bin/, for example, /opt/lsf/10.1/linux2.6-glibc2.3-x86_64/bin/

Usage:

lsportcheck

lsportcheck -h

lsportcheck -l[-m | -s]

Description:

Without arguments will output command usage and exit.

`-h` Output command usage and exit.

`-l` List TCP and UDP ports on master.

`-l -m` List TCP and UDP ports on master.

`-l -s` List TCP and UDP ports on slave.

Note: lsportcheck can only be run by root.

Source the relative IBM Spectrum LSF shell script after installation:

For csh or tcsh: 'source $LSF_ENVDIR/cshrc.lsf'

For sh, ksh, or bash: 'source $LSF_ENVDIR/profile.lsf'

Example output:

Example of the output using command **`lsportcheck -l`** or **`lsportcheck -l -m`** on **LSF master:**

```
Checking ports required on host [mymaster1]
-----------------------------------------------------------------------
Program Name  Port Number  Protocol  Binding Address  PID/Status
-----------------------------------------------------------------------
lim           7869         TCP       0.0.0.0          1847
lim           7869         UDP       0.0.0.0          1847
res           6878         TCP       0.0.0.0          1881
sbatchd       6882         TCP       0.0.0.0          1890
mbatchd       6881         TCP       0.0.0.0          1921
mbatchd       6891         TCP       0.0.0.0          1921
pem           7871         TCP       0.0.0.0          1879
vemkd         7870         TCP       0.0.0.0          1880
egosc         7872         TCP       0.0.0.0          3226
-----------------------------------------------------------------------
Optional ports:
-----------------------------------------------------------------------
wsgserver     9090         TCP       0.0.0.0          [Not used]
named         53           TCP       0.0.0.0          [Not used]
named         53           UDP       0.0.0.0          [Not used]
named         953          TCP       0.0.0.0          [In use by another program]
```

Example output:

Example of the output using command **`lsportcheck -l -s`** on **LSF slave:**

```
Checking ports required on host [host1]
-----------------------------------------------------------------------
Program Name  Port Number  Protocol  Binding Address  PID/Status
-----------------------------------------------------------------------
lim           7869         TCP       0.0.0.0          1847
lim           7869         UDP       0.0.0.0          1847
res           6878         TCP       0.0.0.0          1881
sbatchd       6882         TCP       0.0.0.0          1890
pem           7871         TCP       0.0.0.0          1879
```

Increased project name size

In previous versions of LSF, when submitting a job with a project name (by using the **`bsub -P`** option, the **`DEFAULT_PROJECT`** parameter in the `lsb.params` file, or by using the **`LSB_PROJECT_NAME`** or **`LSB_DEFAULTPROJECT`** environment variables), the

maximum length of the project name was 59 characters. The maximum length of the project name is now increased to 511 characters.

This increase also applies to each project name that is specified in the **PER_PROJECT** and **PROJECTS** parameters in the `lsb.resources` file.

Cluster-wide DNS host cache

LSF can generate a cluster-wide DNS host cache file (`$LSF_ENVDIR/.hosts.dnscache`) that is used by all daemons on each host in the cluster to reduce the number of times that LSF daemons directly call the DNS server when starting the LSF cluster. To enable the cluster-wide DNS host cache file, configure `LSF_DNS_CACHE=Y` in the `lsf.conf` file.

Use #include for shared configuration file content

In previous versions of LSF, you can use the `#INCLUDE` directive to insert the contents of a specified file into the beginning of the `lsf.shared` or `lsb.applications` configuration files to share common configurations between clusters or hosts.

You can now use the `#INCLUDE` directive in any place in the following configuration files:

- `lsb.applications`
- `lsb.hosts`
- `lsb.queues`
- `lsb.reasons`
- `lsb.resources`
- `lsb.users`

You can use the `#INCLUDE` directive only at the beginning of the following file:

- `lsf.shared`

For example, you can use `#if ... #endif` Statements to specify a time-based configuration that uses different configurations for different times. You can change the configuration for the entire system by modifying the common file that is specified in the `#INCLUDE` directive.

See more information on shared configuration file content in *IBM Spectrum LSF Advanced Configuration and Troubleshooting*.

Showing the pending reason for interactive jobs

The **bsub -I** command now displays the pending reason for interactive jobs, based on the setting of **LSB_BJOBS_PENDREASON_LEVEL**, if the job is pending.

Showing warning messages for interactive jobs

Interactive jobs can now show exit reasons when the jobs are killed (due to conditions such as reaching the memory or runtime limit). The exit reason is the same as the message shown for the output of the **bhist -l** and **bjobs -l** commands.

Changing job priorities and limits dynamically

Through the introduction of two new parameters, LSF now supports changing job priorities and limits dynamically through an import file. This includes:

- Calling the `eadmin` script at a configured interval, even when a job exception has not occurred through the parameter **`EADMIN_TRIGGER_INTERVAL`** in the `lsb.params` file.
- Allowing job submission during a policy update or cluster restart through the parameter **`PERSIST_LIVE_CONFIG`** in the `lsb.params` file.
- Enhancement of the **`bconf`** command to override existing settings through the `set` action, to support the `-pack` option for reading multiple requests from a file.

Specify a UDP port range for LSF daemons

You can now specify a range of UDP ports to be used by LSF daemons. Previously, LSF binds to a random port number between 1024 and 65535.

To specify a UDP port range, define the **`LSF_UDP_PORT_RANGE`** parameter in the `lsf.conf` file. Include at least 10 ports in this range, and you can specify integers between 1024 and 65535.

What's new in IBM Spectrum LSF Version 10.1 Fix Pack 5

The following topics summarize the new and changed behavior in LSF 10.1 Fix Pack 5. This Fix Pack applies only to IBM POWER9 platforms.

Release date: May 2018

Resource management

The following new features affect resource management and allocation.

Note: LSF 10.1 Fix Pack 5 applies only to IBM POWER9 platforms.

Running LSF jobs with IBM Cluster Systems Manager

LSF now allows you to run jobs with IBM Cluster Systems Manager (CSM).

The CSM integration allows you to run LSF jobs with CSM features.

See more information on LSF with Cluster Systems Manager in *Administering IBM Spectrum LSF*.

Direct data staging

LSF now allows you to run direct data staging jobs, which uses a burst buffer (for example, IBM CAST burst buffer) instead of the cache to stage in and stage out data for data jobs.

Use the CSM integration to configure LSF to run burst buffer data staging jobs.

See more information on burst bugger data staging jobs in *Administering IBM Spectrum LSF*.

Job scheduling and execution

The following new features affect LSF job scheduling and execution.

Note: LSF 10.1 Fix Pack 5 applies only to IBM POWER9 platforms.

Plan-based scheduling and reservations

When enabled, LSF's plan-based scheduling makes allocation plans for jobs based on anticipated future cluster states. LSF reserves resources as needed in order to carry out its plan. This helps to avoid starvation of jobs with special resource requirements.

Plan-based scheduling and reservations addresses a number of issues with the older reservation features in LSF. For example:

- It ensures that reserved resources can really be used by the reserving jobs
- It has better job start-time prediction for reserving jobs, and thus better backfill decisions

Plan-based scheduling aims to replace legacy LSF reservation policies. When **ALLOCATION_PLANNER** is enabled in the `lsb.params` configuration file, then parameters related to the old reservation features (that is **SLOT_RESERVE** and **RESOURCE_RESERVE** in `lsb.queues`), are ignored with a warning.

Automatically extend job run limits

You can now configure the LSF allocation planner to extend the run limit for jobs when the resources that are occupied by the job are not needed by other jobs in queues with the same or higher priority. The allocation planner looks at job plans to determine if there are any other jobs that require the current job's resources.

Enable extendable run limits for jobs submitted to a queue by specifying the **EXTENDABLE_RUNLIMIT** parameter in the `lsb.queues` file. Since the allocation planner decides whether the extend the run limit of jobs, you must also enable plan-based scheduling by enabling the **ALLOCATION_PLANNER** parameter in the `lsb.params` file.

See more information on configuring extendable run limits in *Administering IBM Spectrum LSF*.

Default epsub executable files

Similar to **esub** programs, LSF now allows you to define a default **epsub** program that runs even if you do not define mandatory **epsub** programs with the **LSB_ESUB_METHOD** parameter in the `lsf.conf` file. To define a default **epsub** program, create an executable file named `epsub` (with no application name in the file name) in the `LSF_SERVERDIR` directory.

After the job is submitted, LSF runs the default **epsub** executable file if it exists in the `LSF_SERVERDIR` directory, followed by any mandatory **epsub** executable files that are defined by **LSB_ESUB_METHOD**, followed by the **epsub** executable files that are specified by the `-a` option.

See more information on external job submission and execution controls in *Administering IBM Spectrum LSF*

Restrict users and user groups from forwarding jobs to remote clusters

You can now specify a list of users or user groups that can forward jobs to remote clusters when using the LSF multicluster capability. This allows you to prevent jobs from certain users or user groups from being forwarded to an execution cluster, and to set limits on the submission cluster.

These limits are defined at the queue level in LSF. For jobs that are intended to be forwarded to a remote cluster, users must submit these jobs to queues that have

the **SNDJOBS_TO** parameter configured in the `lsb.queues` file. To restrict these queues to specific users or user groups, define the **FWD_USERS** parameter in the `lsb.queues` file for these queues.

See more information on multicluster queues in *Using IBM Spectrum LSF multicluster capability*.

Advance reservations now support the "same" section in resource requirement strings

When using the **brsvadd -R** and **brsvmod -R** options to specify resource requirements for advance reservations, the `same` string now takes effect, in addition to the `select` string. Previous versions of LSF only allowed the `select` string to take effect.

This addition allows you to select hosts with the same resources for your advance reservation.

See more information on specifying resource requirements (and the same string) in *Administering IBM Spectrum LSF*.

Priority factors for absolute priority scheduling

You can now set additional priority factors for LSF to calculate the job priority for absolute priority scheduling (APS). These additional priority factors allow you to modify the priority for the application profile, submission user, or user group, which are all used as factors in the APS calculation. You can also view the APS and fairshare user priority values for pending jobs.

To set the priority factor for an application profile, define the **PRIORITY** parameter in the `lsb.applications` file. To set the priority factor for a user or user group, define the **PRIORITY** parameter in the **User** or **UserGroup** section of the `lsb.users` file.

The new **bjobs -prio** option displays the APS and fairshare user priority values for all pending jobs. In addition, the **busers** and **bugroup** commands display the APS priority factor for the specified users or user groups.

See more information on absolute priority scheduling in *Administering IBM Spectrum LSF*.

Job dispatch limits for users, user groups, and queues

You can now set limits on the maximum number of jobs that are dispatched in a scheduling cycle for users, user groups, and queues. This allows you to control the number of jobs, by user, user group, or queue, that are dispatched for execution. If the number of dispatched jobs reaches this limit, other pending jobs that belong to that user, user group, or queue that might have dispatched will remain pending for this scheduling cycle.

To set or update the job dispatch limit, run the **bconf** command on the limit object (that is, run `bconf` *`action_type`* `limit=`*`limit_name`*) to define the **JOBS_PER_SCHED_CYCLE** parameter for the specific limit. You can only set job dispatch limits if the limit consumer types are **USERS**, **PER_USER**, **QUEUES**, or **PER_QUEUE**.

For example, `bconf update limit=L1 "JOBS_PER_SCHED_CYCLE=10"`

You can also define the job dispatch limit by defining the **JOBS_PER_SCHED_CYCLE** parameter in the **Limit** section of the `lsb.resources` file.

See more information on configuring resource allocation limits in *Administering IBM Spectrum LSF*.

Command output formatting

The following enhancements affect LSF command output formatting.

Note: LSF 10.1 Fix Pack 5 applies only to IBM POWER9 platforms.

blimits -a option shows all resource limits

The new **blimits -a** command option shows all resource allocation limits, even if they are not being applied to running jobs. Normally, running the **blimits** command with no options displays only resource allocation limits that are being applied to running jobs.

Related concepts:

Display resource allocation limits

View information about resource allocation limits

Related reference:

blimits -a command option

Use bread -w to show messages and attached data files in wide format

LSF allows you to read messages and attached data files from a job in wide format with the new **bread -w** command option. The wide format displays information without truncating fields.

See more information on the **bread -w** command option in *IBM Spectrum LSF Command Reference*.

Other changes to IBM Spectrum LSF

The following changes affect other aspects of LSF behavior.

Note: LSF 10.1 Fix Pack 5 applies only to IBM POWER9 platforms.

Use #include for shared configuration file content

In previous versions of LSF, you can use the `#INCLUDE` directive to insert the contents of a specified file into the beginning of the `lsf.shared` or `lsb.applications` configuration files to share common configurations between clusters or hosts.

You can now use the `#INCLUDE` directive in any place in the following configuration files:

- `lsb.applications`
- `lsb.hosts`
- `lsb.queues`
- `lsb.reasons`
- `lsb.resources`
- `lsb.users`

You can use the `#INCLUDE` directive only at the beginning of the following file:

- `lsf.shared`

For example, you can use `#if` ... `#endif` Statements to specify a time-based configuration that uses different configurations for different times. You can change the configuration for the entire system by modifying the common file that is specified in the `#INCLUDE` directive.

See more information on shared configuration file content in *IBM Spectrum LSF Advanced Configuration and Troubleshooting*.

Showing the pending reason for interactive jobs

The **bsub -I** command now displays the pending reason for interactive jobs, based on the setting of **LSB_BJOBS_PENDREASON_LEVEL**, if the job is pending.

Showing warning messages for interactive jobs

Interactive jobs can now show exit reasons when the jobs are killed (due to conditions such as reaching the memory or runtime limit). The exit reason is the same as the message shown for the output of the **bhist -l** and **bjobs -l** commands.

Changing job priorities and limits dynamically

Through the introduction of two new parameters, LSF now supports changing job priorities and limits dynamically through an import file. This includes:

- Calling the `eadmin` script at a configured interval, even when a job exception has not occurred through the parameter **EADMIN_TRIGGER_INTERVAL** in the `lsb.params` file.
- Allowing job submission during a policy update or cluster restart through the parameter **PERSIST_LIVE_CONFIG** in the `lsb.params` file.
- Enhancement of the **bconf** command to override existing settings through the `set` action, to support the `-pack` option for reading multiple requests from a file.

Specify a UDP port range for LSF daemons

You can now specify a range of UDP ports to be used by LSF daemons. Previously, LSF binds to a random port number between 1024 and 65535.

To specify a UDP port range, define the **LSF_UDP_PORT_RANGE** parameter in the `lsf.conf` file. Include at least 10 ports in this range, and you can specify integers between 1024 and 65535.

What's new in IBM Spectrum LSF Version 10.1 Fix Pack 4

The following topics summarize the new and changed behavior in LSF 10.1 Fix Pack 4

Release date: December 2017

New platform support

The following new features are related to new platform support for LSF.

IBM POWER9

IBM Spectrum LSF 10.1 Fix Pack 4 includes support for IBM POWER9. The package for Linux on IBM Power LE (`lsf10.1_lnx310-lib217-ppc64le`) supports both IBM POWER8 and POWER9.

Performance enhancements

The following enhancements affect performance.

Use IBM Spectrum LSF Explorer to improve the performance of the bacct and bhist commands

The **bacct** and **bhist** commands can now use IBM Spectrum LSF Explorer (LSF Explorer) to get information instead of parsing the `lsb.acct` and `lsb.events` files. Using LSF Explorer improves the performance of the **bacct** and **bhist** commands by avoiding the need for parsing large log files whenever you run these commands.

To use this integration, LSF Explorer, Version 10.2, or later, must be installed and working. To enable this integration, edit the `lsf.conf` file, then define the **LSF_QUERY_ES_SERVERS** and **LSF_QUERY_ES_FUNCTIONS** parameters.

See more information on how to improve the performance of the **bacct** and **bhist** commands in the *Performance Tuning* section of *Administering IBM Spectrum LSF*.

Resource management

The following new feature affects resource management and allocation.

What's new in resource connector for IBM Spectrum LSF

Extended AWS support:
This feature extends LSF the resource connector AWS template to specify an Amazon EBS-Optimized instance. The AWS template also supports LSF exclusive resource syntax (`!resource`) in the instance attributes. LSF considers demand on the template only if a job explicitly asks for the resource in its combined resource requirement.

Launch Google Compute Cloud instances:
LSF clusters can launch instances from Google Compute Cloud to satisfy pending workload. The instances join the LSF cluster. If instances become idle, LSF resource connector automatically deletes them. Configure Google Compute Cloud as a resource provider with the `googleprov_config.json` and `googleprov_templates.json` files.

bhosts -rc and the bhosts -rconly commands show extra host information about provider hosts:
Use the **bhosts -rc** and the **bhosts -rconly** command to see information about resources that are provisioned by LSF resource connector.

The `-rc` and `-rconly` options make use of the third-party **mosquitto** message queue application to support the additional information displayed by these **bhosts** options. The **mosquitto** binary file is included as part of the LSF distribution. To use the **mosquitto** daemon that is supplied with LSF, you must configure the **LSF_MQ_BROKER_HOSTS** parameter in the `lsf.conf` file to enable LIM to start the **mosquitto** daemon and for **ebrokerd** to send resource provider information to the MQTT message broker.

What's new in data manager for IBM Spectrum LSF

Enhanced LSF multicluster job forwarding:
This feature enhances the LSF data manager implementation for the hybrid cloud environment using job forwarding with IBM Spectrum LSF multicluster capability (LSF multicluster capability). In this implementation, the cluster running in the

public cloud is used as the execution cluster, and this feature enables the submission cluster to push the forwarding job's data requirement to the execution cluster and to receive the output back from the forwarding job. To enable this feature, specify the **SNDJOBS_TO** parameter in the `lsb.queues` file for the data transfer queue in the execution cluster, and specify the **RCVJOBS_FROM** parameter in the `lsb.queues` file for the submission cluster. The path of the **FILE_TRANSFER_CMD** parameter in the `lsf.datamanager` file for the data manager host must exist in the submission cluster.

See more information on configuring the data transfer queue in the *Administering LSF data manager* section of *Using IBM Spectrum LSF Data Manager*.

Specify a folder as the data requirement:
When you specify a folder as a data requirement for a job, LSF generates a single signature for the folder as a whole, and only a single transfer job is required. You can also now use symbolically linked files in a job data requirement, and the colon (:) character can now be used in the path of a job data requirement.

When you submit a job with a data requirement, a data requirement that ends in a slash and an asterisk (/*) is interpreted as a folder. Only files at the top-level of the folder are staged. For example,

```
bsub -data "[host_name:]abs_folder_path/*" job
```

When you use the asterisk character (*) at the end of the path, the data requirements string must be in quotation marks.

A data requirement that ends in a slash (/) is also interpreted also as a folder, but all files including subfolders are staged. For example,

```
bsub -data "[host_name:]abs_folder_path/" job
```

To specify a folder a data requirement for a job, you must have access to the folder and its contents. You must have read and execute permission on folders, and read permission on regular files. If you don't have access to the folder, the submission is rejected.

See more information on configuring the data transfer queue in the *Administering LSF data manager* section of *Using IBM Spectrum LSF Data Manager*.

Container support

The following new feature affects LSF support for containers.

Support for systemd with Docker jobs

When running jobs for Docker containers, you can now use the **systemd** daemon as the Docker **cgroup** driver. This means that you can now run Docker jobs regardless of which **cgroup** driver is used.

To support Docker with the **systemd** cgroup driver and all other cgroup drivers, configure both the **EXEC_DRIVER** and **CONTAINER** parameters. This new configuration provides transparent Docker container support for all cgroup drivers and other container features.

See more information on configuring the Docker application profile in LSF in *Administering IBM Spectrum LSF*.

GPU enhancements

The following enhancements affect LSF GPU support.

NVIDIA Data Center GPU Manager (DCGM) integration updates

LSF, Version 10.1 Fix Pack 2 integrated with NVIDIA Data Center GPU Manager (DCGM) to work more effectively with GPUs in the LSF cluster. LSF now integrates with Version 1.1 of the NVIDIA Data Center GPU Manager (DCGM) API. This update provides the following enhancements to the DCGM features for LSF:

- LSF checks the status of GPUs to automatically filter out unhealthy GPUs when the job allocates GPU resources, and to automatically add back the GPU if it becomes healthy again.
- DCGM provides mechanisms to check the GPU health and LSF integrates these mechanisms to check the GPU status before, during, and after the job is running to meet the GPU requirements. If LSF detects that a GPU is not healthy before the job is complete, LSF requeues the job. This ensures that the job runs on healthy GPUs.
- GPU auto-boost is now enabled for single-GPU jobs, regardless of whether DCGM is enabled. If DCGM is enabled, LSF also enables GPU auto-boost on jobs with exclusive mode that run across multiple GPUs on one host.

Enable the DCGM integration by defining the **LSF_DCGM_PORT** parameter in the `lsf.conf` file.

See more information on the **LSF_DCGM_PORT** parameter in *IBM Spectrum LSF Configuration Reference*.

Job scheduling and execution

The following new feature affects LSF job scheduling and execution.

External job switch control with eswitch

Similar to the external job submission and execution controls (**esub**, **epsub**, and **eexec** programs), LSF now allows you to use external, site-specific binary files or scripts that are associated with a request to switch a job to another queue. By writing external job switch executable files, you can accept, reject, or change the destination queue for any **bswitch** request.

Similar to the **bsub -a** option, the new **bswitch -a** option specifies one or more application-specific external executable files (**eswitch** files) that you want LSF to associate with the switch request.

Similar to the **LSB_ESUB_METHOD** parameter, the new **LSB_ESWITCH_METHOD** environment variable or parameter in the `lsf.conf` file allows you to specify one or more mandatory **eswitch** executable files.

When running any job switch request, LSF first invokes the executable file named **eswitch** (without *.application_name* in the file name) if it exists in the **LSF_SERVERDIR** directory. If an LSF administrator specifies one or more mandatory **eswitch** executable files using the **LSB_ESWITCH_METHOD** parameter in the `lsf.conf` file, LSF then invokes the mandatory executable files. Finally, LSF invokes any application-specific **eswitch** executable files (with *.application_name* in the file name) specified by the **bswitch -a** option. An **eswitch** is run only once, even if it is specified by both the **bswitch -a** option and the **LSB_ESWITCH_METHOD** parameter.

See more information on how to use external job switch controls in *Administering IBM Spectrum LSF*

Advance reservation enhancements

LSF now features enhancements to advance reservations. You can enable LSF to allow jobs to run on advance reservation hosts even if it cannot finish before the advance reservation becomes active (by default, these jobs are suspended when the first advance reservation job starts). The advance reservations can run a pre-script before the advance reservation starts and a post-script when the advance reservation expires. These enhancements are specified in the **brsvadd** and **brsvmod** commands (`-q`, `-nosusp`, `-E`, `-Et`, `-Ep`, and `-Ept` options).

Because the **ebrokerd** daemon starts the advance reservation scripts, you must specify **LSB_START_EBROKERD=Y** in the `lsf.conf` file to enable advance reservations to run pre-scripts and post-scripts.

Related tasks:

Adding an advance reservation

Related reference:

brsvadd command

brsvmod command

LSB_START_EBROKERD parameter in the `lsf.conf` file.

Deleting empty job groups

This enhancement supports the deleting of empty implicit job groups automatically even if they have limits. It adds a new option "all" to the **JOB_GROUP_CLEAN** parameter in `lsb.params`, to delete empty implicit job groups automatically even if they have limits.

Related reference:

JOB_GROUP_CLEAN parameter in the `lsb.params` file

Data collection

The following new features affect IBM Spectrum LSF data collection.

Enhanced energy accounting using Elasticsearch

This enhancement introduces the **lsfbeat** tool, which calls the **ipmitool** to collect the energy data of each host and to send the data to IBM Spectrum LSF Explorer (LSF Explorer). The **bjobs** and **bhosts** commands get the energy data from LSF Explorer and display it. To use this feature, LSF Explorer must be deployed in your LSF cluster. To enable the **lsfbeat** energy service, configure `LSF_ENABLE_BEAT_SERVICE="energy"` in the lsf.conf file, then run the **lsadmin limrestart all** command to start up the **lsfbeat** service. To query energy data with the **bhosts** and **bjobs** commands, configure `LSF_QUERY_ES_FUNCTIONS="energy"` and `LSF_QUERY_ES_SERVERS="`*`ip`*`:`*`port`*`"` in the `lsf.conf` file.

Data provenance tools

LSF now has data provenance tools to trace files that are generated by LSF jobs.

You can use LSF data provenance tools to navigate your data to find where the data is coming from and how it is generated. In addition, you can use data provenance information to reproduce your data results when using the same job input and steps.

When submitting a job with the **bsub** command, enable data provenance by defining `LSB_DATA_PROVENANCE=Y` as an environment variable (`bsub -e`

LSB_DATA_PROVENANCE=Y) or by using the **esub.dprov** application (bsub -a 'dprov(*file_path*)'), and use the **tag.sh** post-execution script to mark the data provenance attributes for the output data files (**-Ep 'tag.sh'**). You can also use the showhist.py script to generate a picture to show the relationship of your data files.

Data provenance requires the use of IBM Spectrum Scale (GPFS) as the file system to support the extended attribute specification of files and Graphviz, which is an open source graph visualization software, to generate pictures from the showhist.py script.

See more information on data provenance in *Administering IBM Spectrum LSF*

Command output formatting

The following enhancements affect LSF command output formatting.

esub and epsub enhancement

LSF users can select different esub (or epsub) applications (or scripts) using **bsub -a** (or **bmod -a**). LSF has a number of different esub applications that users can select, but the **bjobs** and **bhist** commands did not previously show details about these applications in output. This enhancement enables the **bjobs -l**, **bjobs -o esub**, and **bhist -l** commands to show detailed information about esub and epsub applications.

Related reference:

bjobs -l command

bjobs -o esub command

bhist -l command

Energy usage in output of bjobs -l, bjobs -o, and bhosts -l

If enhanced energy accounting using Elasticsearch has been enabled (with **LSF_ENABLE_BEAT_SERVICE** in lsf.conf), output for **bjobs -l** and **bjobs -o energy** shows the energy usage in Joule and kWh. and **bhosts -l** shows the **Current Power** usage in watts and total **Energy Consumed** in Joule and kWh.

Related reference:

LSB_ENABLE_BEAT_SERVICE parameter in lsf.conf

bjobs -l command

Other changes to IBM Spectrum LSF

The following changes affect other aspects of LSF behavior.

Enhance fairshare calculation for job fowarding mode in the LSF multicluster capability

In previous versions of LSF, when calculating the user priority in the fairshare policies, if a job is forwarded to remote clusters while using the LSF multicluster capability, the fairshare counter for the submission host is not updated. For example, if the fairshare calculation determines that a user's job has a high priority and there are no local resources available, that job is forwarded to a remote cluster, but the LSF scheduler still considers the job for forwarding purposes again because the fairshare counter is not updated.

To resolve this issue, LSF now introduces a new forwarded job slots factor (**FWD_JOB_FACTOR**) to account for forwarded jobs when making the user priority calculation for the fairshare policies. To use this forwarded job slots factor, specify the **FWD_JOB_FACTOR** to a non-zero value in the lsb.params file for cluster-wide

settings, or in the `lsb.queues` file for an individual queue. If defined in both files, the queue value takes precedence. In the user priority calculation, the **FWD_JOB_FACTOR** parameter is used for forwarded job slots in the same way that the **RUN_JOB_FACTOR** parameter is used for job slots. To treat remote jobs and local jobs as the same, set **FWD_JOB_FACTOR** to the same value as **RUN_JOB_FACTOR**.

When accounting for forwarded jobs in the fairshare calculations, job usage might be counted twice if global fairshare is used because job usage is counted on the submission cluster, then counted again when the job is running on a remote cluster. To avoid this problem, specify the duration of time after which LSF removes the forwarded jobs from the user priority calculation for fairshare scheduling by specifying the **LSF_MC_FORWARD_FAIRSHARE_CHARGE_DURATION** parameter in the `lsf.conf` file. If you enabled global fairshare and intend to use the new forwarded job slots factor, set the value of **LSF_MC_FORWARD_FAIRSHARE_CHARGE_DURATION** to double the value of the **SYNC_INTERVAL** parameter in the `lsb.globalpolicies` file (approximately 5 minutes) to avoid double-counting the job usage for forwarded jobs. If global fairshare is not enabled, this parameter is not needed.

See more information on how to enhance fairshare calculation to include the job fowarding mode in *Using IBM Spectrum LSF multicluster capability*

Dynamically load the hardware locality (hwloc) library

LSF now allows you to dynamically load the hardware locality (**hwloc**) library from the system library paths whenever it is needed to support newer hardware.

LSF for the following platforms is compiled and linked with the library and header file for **hwloc**, Version 1.11.8, and detects most of the latest hardware without enabling this feature:

- Linux x64 Kernel 2.6, glibc 2.5
- Linux x64 Kernel 3.10, glibc 2.17
- Linux ppc64le Kernel 3.10, glibc 2.17
- Linux ARMv8 Kernel 3.12, glibc 2.17
- Windows

All other platforms use **hwloc**, Version 1.8.

Enable the dynamic loading of the **hwloc** library by defining the **LSF_HWLOC_DYNAMIC** parameter as `Y` in the `lsf.conf` file.

See more information on the **LSF_HWLOC_DYNAMIC** parameter in *IBM Spectrum LSF Configuration Reference*.

What's new in IBM Spectrum LSF Version 10.1 Fix Pack 3

The following topics summarize the new and changed behavior in LSF 10.1 Fix Pack 3

Release date: August 2017

Job scheduling and execution

The following new features affect job scheduling and execution.

View jobs that are associated with an advance reservation

The new **bjobs -U** option allows you to display jobs that are associated with the specified advance reservation.

To view the reservation ID of the advance reservation that is associated with a job ID, use the **bjobs -o** option and specify the `rsvid` column name.

See the information on how to view jobs that are associated with an advance reservation in *IBM Spectrum LSF Parallel Workload Administration*.

Dynamically scheduled reservations

A *dynamically scheduled* reservation accepts jobs based on currently available resources. Use the **brsvsub** command to create a dynamically scheduled reservation and submit a job to to fill the advance reservation when the resources required by the job are available.

Jobs that are scheduled for the reservation run when the reservation is active. Because they are scheduled like jobs, dynamically scheduled reservations do not interfere with running workload (unlike normal advance reservations, which kill any running jobs when the reservation window opens.)

Related concepts:

Advance Reservation

Related reference:

brsvsub

Resource management

The following new feature affects resource management and allocation.

Request additional resources to allocate to running jobs

The new **bresize request** subcommand option allows you to request additional tasks to be allocated to a running resizable job, which grows the resizable job. This means that you can both grow and shrink a resizable job by using the **bresize** command.

See the information on how to work with resizable jobs in *IBM Spectrum LSF Parallel Workload Administration*.

Specify GPU resource requirements for your jobs

Specify all GPU resource requirement as part of job submission, or in a queue or application profile. Use the option `bsub –gpu` to submit jobs that require GPU resources. Specify how LSF manages GPU mode (exclusive or shared), and whether to enable the NVIDIA Multi-Process Service (MPS) for the GPUs used by the job.

The parameter **LSB_GPU_NEW_SYNTAX** in the `lsf.conf` file enables jobs to use GPU resource requirements that are specified with the **bsub -gpu** option or in the queue, application profile.

Use the `bsub -gpu` option to specify GPU requirements for your job or submit your job to a queue or application profile that configures GPU requirements in the **GPU_REQ** parameter.

Set a default GPU requirement by configuring the **LSB_GPU_REQ** parameter in the `lsf.conf` file.

Use the **bjobs -l** command to see the combined and effective GPU requirements that are specified for the job.

What's new in resource connector for IBM Spectrum LSF

Support for new resource providers

LSF resource connector now supports IBM Bluemix (formerly Softlayer) and Microsoft Azure as resource providers. LSF clusters can borrow virtual compute hosts from the IBM Bluemix services or launch instances from Microsoft Azure if the workload demand exceeds cluster capacity. The resource connector generates requests for additional hosts from these providers and dispatches jobs to dynamic hosts that join the LSF cluster. When the demand reduces, the resource connector shuts down the LSF slave daemons and cancels allocated virtual servers.

To specify the configuration for provisioning from Microsoft Azure, use the `azureprov_config.json` and the `azureprov_templates.json` configuration files.

To specify the configuration for provisioning from IBM Bluemix, use the `softlayerprov_config.json` and the `softlayerprov_template.json` configuration files.

Submit jobs to use AWS Spot instances

Use *Spot instances* to bid on spare Amazon EC2 computing capacity. Since Spot instances are often available at a discount compared to the pricing of On-Demand instances, you can significantly reduce the cost of running your applications, grow your application's compute capacity and throughput for the same budget, and enable new types of cloud computing applications.

With Spot instances you can reduce your operating costs by up to 50-90%, compared to on-demand instances. Since Spot instances typically cost 50-90% less, you can increase your compute capacity by 2-10 times within the same budget.

Spot instances are supported on any Linux x86 system that is supported by LSF.

Support federated accounts with temporary access tokens

Resource connector supports *federated accounts* for LSF resource connector as an option instead of requiring permanent AWS IAM account credentials. Federated users are external identities that are granted temporary credentials with secure access to resources in AWS without requiring creation of IAM users. Users are authenticated outside of AWS (for example, through Windows Active Directory).

Use the **AWS_CREDENTIAL_SCRIPT** parameter in the `awsprov_config.json` file to specify a path to the script that generates temporary credentials for federated accounts. For example,

```
AWS_CREDENTIAL_SCRIPT=/shared/dir/generateCredentials.py
```

LSF executes the script as the primary LSF administrator to generate a temporary credentials before it creates the EC2 instance.

Support starting instances within an IAM Role

IAM *roles* group AWS access control privileges together. A role can be assigned to an IAM user or an IAM instance profile. IAM *Instance Profiles* are containers for IAM roles that allow you to associate an EC2 instance with a role through the

profile. The EC2 runtime environment contains temporary credentials that have the access control permissions of the profile role.

To make the roles available for resource connector to create instances, use the `instanceProfile` attribute in the `awsprov_templates.json` file to specify an AWS IAM instance profile to assign to the requested instance. Jobs running in that instance can use the instance profile credentials to access other AWS resources. Resource connector uses that information to request EC2 compute instances with particular instance profiles. Jobs that run on those hosts use temporary credentials provided by AWS to access the AWS resources that the specified role has privileges for.

Tag attached EBS volumes in AWS

The **`instanceTags`** attribute in the `awsprov_templates.json` file can tag EBS volumes with the same tag as the instance. EBS volumes in AWS are persistent block storage volumes used with an EC2 instance. EBS volumes are expensive, so you can use the instance ID that tags the volumes for the accounting purposes.

Note: The tags cannot start with the string `aws:`. This prefix is reserved for internal AWS tags. AWS gives an error if an instance or EBS volume is tagged with a keyword starting with `aws:`. Resource connector removes and ignores user-defined tags that start with `aws:`.

Resource connector demand policies in queues

The **`RC_DEMAND_POLICY`** parameter in the `lsb.queues` file defines threshold conditions to determine whether demand is triggered to borrow resources through resource connector for all the jobs in the queue. As long as pending jobs at the queue meet at least one threshold condition, LSF expresses the demand to resource connector to trigger borrowing.

The demand policy defined by the **`RC_DEMAND_POLICY`** parameter can contain multiple conditions, in an OR relationship. A condition is defined as [*num_pend_jobs*[,*duration*]]. The queue has more than the specified number of eligible pending jobs that are expected to run at least the specified duration in minutes. The num_pend_jobs option is required, and the duration is optional. The default duration is 0 minutes.

View the status of provisioned hosts with the bhosts -rc command

Use the **`bhosts -rc`** or the **`bhosts -rconly`** command to see the status of resources provisioned by LSF resource connector.

To use the `-rc` and `-rconly` options, the **`mosquitto`** binary file for the MQTT broker must be installed in `LSF_SERVERDIR`, and running (check with the `ps -ef | grep mosquitto` command). The the **`LSF_MQ_BROKER_HOSTS`** parameter must be configured in the `lsf.conf` file.

For hosts provisioned by resource connector, the `RC_STATUS`, `PROV_STATUS`, and `UPDATED_AT` columns show appropriate status values and a timestamp. For other hosts in the cluster, these columns are empty.

For example,

```
bhosts -rc
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV RC_STATUS      PROV_STATUS    UPDATED_AT
ec2-35-160-173-192 ok              -      1      0      0      0      0      0 Allocated      running        2017-04-07T1
lsf1.aws.          closed          -      1      0      0      0      0      0
```

The `-l` option shows more detailed information about provisioned hosts.

```
bhosts -rc -l
HOST  ec2-35-160-173-192.us-west-2.compute.amazonaws.com
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV RC_STATUS      PROV_STATUS    UPDATED_AT
ok              60.00     -      1      0      0      0      0      0 Allocated      running        2017-04-07T12:28:46C

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem  slots
 Total           1.0   0.0   0.0    1%   0.0    33    0     3 5504M    0M  385M      1
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M      -
```

The `-rconly` option shows the status of all hosts provisioned by LSF resource connector, no matter if they have joined the cluster or not.

For more information about LSF resource connector, see *Using the IBM Spectrum LSF resource connector*.

Related concepts:

Use AWS Spot instances

Related tasks:

Configuring AWS Spot instances

Advanced configuration for IBM Spectrum LSF resource connector

Configuring AWS access with federated accounts

Related reference:

awsprov_config.json

awsprov_templates.json

policy_config.json

lsf.conf file reference for resource connectory

RC_DEMAND_POLICY in lsb.queues

Container support

The following new feature affects LSF support for containers.

Pre-execution scripts to define container options

When running jobs for Docker, Shifter, or Singularity, you can now specify a pre-execution script that outputs container options that are passed to the container job. This allows you to use a script to set up the execution options for the container job.

See the information on how to configure Docker, Shifter, or Singularity application profiles in *Administering IBM Spectrum LSF*.

Command output formatting

The following new features are related to the LSF command output.

Customize host load information output

Like the **bjobs -o** option, you can now also customize specific fields that the **lsload** command displays by using the `-o` command option. This allows you to create a specific output format, allowing you to easily parse the information by using custom scripts or to display the information in a predefined format.

You can also specify the default output formatting of the **lsload** command by specifying the **LSF_LSLOAD_FORMAT** parameter in the lsf.conf file, or by specifying the **LSF_LSLOAD_FORMAT** environment variable.

See the information on how to customize host load information output in *Administering IBM Spectrum LSF*.

View customized host load information in JSON format

With this release, you can view customized host load information in JSON format by using the new -json command option with the **lsload** command. Since JSON is a customized output format, you must use the -json option together with the -o option.

See the information on how to view customized host load information in JSON format in *Administering IBM Spectrum LSF*.

New output fields for busers -w

With this release, two new fields have been added to the output for **busers -w**: PJOBS and MPJOBS.

The fields shown for **busers -w** now includes:

PEND
The number of tasks in all of the specified users' pending jobs. If used with the -alloc option, the total is 0.

MPEND
The pending job slot threshold for the specified users or user groups. MPEND is defined by the **MAX_PEND_SLOTS** parameter in the lsb.users configuration file.

PJOBS
The number of users' pending jobs.

MPJOBS
The pending job threshold for the specified users. MPJOBS is defined by the **MAX_PEND_JOBS** parameter in the configuration file lsb.users.

Logging and troubleshooting

The following new features are related to logging and troubleshooting.

Diagnose mbatchd and mbschd performance problems

LSF provides a feature to log profiling information for the **mbatchd** and **mbschd** daemons to track the time that the daemons spend on key functions. This can assist IBM Support with diagnosing daemon performance problems.

To enable daemon profiling with the default settings, edit the lsf.conf file, then specify LSB_PROFILE_MBD=Y for the **mbatchd** daemon or specify LSB_PROFILE_SCH=Y for the **mbschd** daemon. You can also add keywords within these parameters to further customize the daemon profilers.

See more information on logging mbatchd and mbschd profiling information in *Administering IBM Spectrum LSF*.

Related concepts:

Logging mbatchd and mbschd profiling information

Related reference:

LSB_PROFILE_MBD parameter in the lsf.conf file

LSB_PROFILE_SCH parameter in the `lsf.conf` file

Other changes to IBM Spectrum LSF

The following changes are related to command options and LSF default behavior.

Changed command options

Specify multiple email addresses with the bsub -u option

You can now specify multiple email addresses with the **bsub -u** option by enclosing the string in quotation marks and using a space to separate each email address. The total length of the address string cannot be longer than 511 characters.

The bpeek -f option now exits when the peeked job is complete

The **bpeek -f** command option now exits when the peeked job is completed.

If the peeked job is requeued or migrated, the **bpeek** command only exits if the job is completed again. In addition, the **bpeek** command cannot get the new output of the job. To avoid these issues, abort the previous **bpeek -f** command and rerun the **bpeek -f** command after the job is requeued or migrated.

Specify remote hosts with the bsub -m option

You can now specify remote hosts by using the **bsub -m** command option when using the job forwarding model with the LSF multicluster capability. To specify remote hosts, use *host_name@cluster_name*.

Changed configuration parameters

New MAX_PEND_SLOTS parameter and change to MAX_PEND_JOBS parameter

With the addition of the new parameter **MAX_PEND_SLOTS**, the concept of **MAX_PEND_JOBS** has changed. **MAX_PEND_JOBS** (in both `lsb.users` and `lsb.params`) has been changed to control the maximum pending "jobs" where previously it controlled the maximum pending "slot" threshold. **MAX_PEND_SLOTS** has been introduced therefore, to control the previous intention of **MAX_PEND_JOBS**.

This means that for customers who previously configured **MAX_PEND_JOBS** (for example, in lsb.users, for a user or group pending job slot limit), they must update the parameter to job count instead of slot count, or replace the parameter with the new **MAX_PEND_SLOTS**, which is meant for backward compatibility.

Changes to default LSF behavior

Improvements to the LSF Integration for Rational ClearCase

Daemon wrapper performance is improved with this release because the daemon wrappers no longer run the **checkView** function to check the ClearCase view (as set by the **CLEARCASE_ROOT** environment variable) under any conditions. In addition, the **NOCHECKVIEW_POSTEXEC** environment variable is now obsolete since it is no longer needed.

If the **cleartool setview** command fails when called by a daemon wrapper, the failure reason is shown in the **bjobs -l**, **bhist -l**, **bstatus**, and **bread** commands if `DAEMON_WRAP_ENABLE_BPOST=Y` is set as an environment variable.

What's new in IBM Spectrum LSF Version 10.1 Fix Pack 2

The following topics summarize the new and changed behavior in LSF 10.1 Fix Pack 2

Performance enhancements

The following new features can improve performance.

Improved mbatchd performance and scalability

Job dependency evaluation is used to check whether each job's dependency condition is satisfied. You can improve the performance and scalability of the **mbatchd** daemon by limiting the amount of time that **mbatchd** takes to evaluate job dependencies in one scheduling cycle. This limits the amount of time that the job dependency evaluation blocks services and frees up time to perform other services during the scheduling cycle. Previously, you could only limit the maximum number of job dependencies, which only indirectly limited the amount of time spent evaluating job dependencies. Job dependency evaluation is a process that is used to check whether each job's dependency condition is satisfied.

See more information on the **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameter in the `lsb.params` file in *IBM Spectrum LSF Configuration Reference*.

Improve performance of LSF daemons by automatically configuring CPU binding

You can now enable LSF to automatically bind LSF daemons to CPU cores by enabling the **LSF_INTELLIGENT_CPU_BIND** parameter in the `lsf.conf` file. LSF automatically creates a CPU binding configuration file for each master and master candidate host according to the automatic binding policy.

See the information on how to automatically bind LSF daemons to specific CPU cores in *Administering IBM Spectrum LSF*.

Reduce mbatchd workload by allowing user scripts to wait for a specific job condition

The new **bwait** command pauses and waits for the specified job condition to occur before the command returns. End users can use this command to reduce workload on the **mbatchd** daemon by including **bwait** in a user script for running jobs instead of using the **bjobs** command in a tight loop to check the job status. For example, the user script might have a command to submit a job, then run **bwait** to wait for the first job to be `DONE` before continuing the script.

The new **lsb_wait()** API provides the same functionality as the **bwait** command.

See more information on the **bwait** command in *IBM Spectrum LSF Command Reference*. See more information about the **EVALUATE_WAIT_CONDITION_TIMEOUT** parameter in *IBM Spectrum LSF Configuration Reference*.

Changes to default LSF behavior

Parallel restart of the mbatchd daemon

The **mbatchd** daemon now restarts in parallel by default. This means that there is always an **mbatchd** daemon handling client commands during the restart to help minimize downtime for LSF. LSF starts a new or child **mbatchd** daemon process to read the configuration files and replace the event file. Previously, the **mbatchd** daemon restarted in serial by default and required the use of the **badmin mbdrestart -p** command option to restart in parallel. To explicitly enable the

mbatchd daemon to restart in serial, use the new **badmin mbdrestart -s** command option.

New default value for caching a failed DNS lookup

The default value of the **LSF_HOST_CACHE_NTTL** parameter in the `lsf.conf` file is increased to the maximum valid value of 60 seconds (from 20 seconds). This reduces the amount of time that LSF takes to repeat failed DNS lookup attempts.

Multithread mbatchd job query daemon

LSF enables the multithread **mbatchd** job query daemon by setting the following parameter values at the time of installation:

- The **LSB_QUERY_PORT** parameter in the `lsf.conf` file is set to `6891`, which enables the multithread **mbatchd** job query daemon and specifies the port number that the **mbatchd** daemon uses for LSF query requests.
- The **LSB_QUERY_ENH** parameter in the `lsf.conf` file is set to `Y`, which extends multithreaded query support to batch query requests (in addition to **bjobs** query requests).

Container support

The following new features affect LSF support for containers.

Running LSF jobs in Shifter containers

LSF now supports the use of Shifter, Version 16.08.3, or later, which must be installed on an LSF server host.

The Shifter integration allows LSF to run jobs in Shifter containers on demand.

See the information on running LSF with Shifter in *Administering IBM Spectrum LSF*.

Running LSF jobs in Singularity containers

LSF now supports the use of Singularity, Version 2.2, or later, which must be installed on an LSF server host.

The Singularity integration allows LSF to run jobs in Singularity containers on demand.

See the information on running LSF with Singularity in *Administering IBM Spectrum LSF*.

GPU

The following new features affect GPU support.

Integration with NVIDIA Data Center GPU Manager (DCGM)

The NVIDIA Data Center GPU Manager (DCGM) is a suite of data center management tools that allow you to manage and monitor GPU resources in an accelerated data center. LSF integrates with NVIDIA DCGM to work more effectively with GPUs in the LSF cluster. DCGM provides additional functionality when working with jobs that request GPU resources by:

- providing GPU usage information for the jobs.
- checking the GPU status before and after the jobs run to identify and filter out unhealthy GPUs.

- synchronizing the GPU auto-boost feature to support jobs that run across multiple GPUs.

Enable the DCGM integration by defining the **LSF_DCGM_PORT** parameter in the `lsf.conf` file.

See more information on the **LSF_DCGM_PORT** parameter in *IBM Spectrum LSF Configuration Reference*.

Related information:

LSF_DCGM_PORT parameter in the `lsf.conf` file

Installation

The following new features affect LSF installation.

Enabling support for Linux cgroup accounting to control resources

Control groups (**cgroups**) are a Linux feature that affects the resource usage of groups of similar processes, allowing you to control how resources are allocated to processes that are running on a host.

With this release, you can enable the **cgroup** feature with LSF by enabling the **ENABLE_CGROUP** parameter in the `install.config` file for LSF installation. The LSF installer sets initial configuration parameters to use the **cgroup** feature.

See more information about the **ENABLE_CGROUP** parameter in the `install.config` file in *IBM Spectrum LSF Configuration Reference* or *Installing IBM Spectrum LSF on UNIX and Linux*.

Automatically enable support for GPU resources at installation

Support for GPU resources in previous versions of LSF required manual configuration of the GPU resources in the `lsf.shared` and `lsf.cluster.`*`cluster_name`* files.

With this release, you can enable LSF to support GPUS automatically by enabling the **ENABLE_GPU** parameter in the `install.config` file for LSF installation. The LSF installer sets initial configuration parameters to support the use of GPU resources.

For more information on the **ENABLE_GPU** parameter in the `install.config` file, see *IBM Spectrum LSF Configuration Reference* or *Installing IBM Spectrum LSF on UNIX and Linux*.

Resource management

The following new features affect resource management and allocation.

Accurate affinity accounting for job slots

Affinity accounting is an extension of HPC allocation feature, where LSF accounts all the slots on the allocated hosts for exclusive jobs. Previous versions of LSF miscalculated the job accounting for job slots when affinity is used in the resource requirement string (in the **bsub -R** option). LSF can now accurately account the number of slots that are consumed by jobs with affinity requirements. LSF calculates the number of slots that are required by affinity jobs when the job task is allocated to the host. The processor unit (PU) that is used for calculating the number of slots is the effective ncpus value on the host. LSF uses this effective ncpus value to calculate the number of slots that are required by affinity jobs when the job task is allocated to the host.

Enable HPC allocation and affinity accounting by defining the **LSB_ENABLE_HPC_ALLOCATION** parameter in the `lsf.conf` file.

See more information on the **LSF_ENABLE_HPC_ALLOCATION** parameter in *IBM Spectrum LSF Configuration Reference*.

Pre-provisioning and post-provisioning in LSF resource connector

Set up pre-provisioning in LSF resource connector to run commands before the resource instance joins the cluster. Configure post-provisioning scripts to run clean up commands after the instance is terminated, but before the host is removed from the cluster.

Configure resource provisioning policies in LSF resource connector

LSF resource connector provides built in policies for limiting the number of instances to be launched and the maximum number of instances to be created. The default plugin framework is a single python script that communicates via `stdin` and `stdout` in JSON data structures. LSF resource connector provides an interface for administrators to write their own resource policy plugin.

Improvements to units for resource requirements and limits

For the **bsub**, **bmod**, and **brestart** commands, you can now use the `ZB` (or `Z`) unit in addition to the following supported units for resource requirements and limits: `KB` (or `K`), `MB` (or `M`), `GB` (or `G`), `TB` (or `T`), `PB` (or `P`), `EB` (or `E`). The specified unit is converted to the appropriate value specified by the **LSF_UNIT_FOR_LIMITS** parameter. The converted limit values round up to a positive integer. For resource requirements, you can specify unit for `mem`, `swp` and `tmp` in `select` and `rusage` section.

By default, the `tmp` resource is not supported by the **LSF_UNIT_FOR_LIMITS** parameter. Use the parameter **LSF_ENABLE_TMP_UNIT=Y** to enable the **LSF_UNIT_FOR_LIMITS** parameter to support limits on the `tmp` resource.

When the **LSF_ENABLE_TMP_UNIT=Y** parameter is set and the **LSF_UNIT_FOR_LIMITS** parameter value is not `MB`, an updated LIM used with old query commands has compatibility issues. The unit for the `tmp` resource changes with the **LSF_UNIT_FOR_LIMITS** parameter in LIM, but query commands still display the unit for the `tmp` resource as `MB`.

Command output formatting

The following new features are related to the LSF command output.

Customize host and queue information output

Like the **bjobs -o** option, you can now also customize specific fields that the **bhosts** and **bqueues** commands display by using the `-o` command option. This allows you to create a specific output format that shows all the required information, which allows you to easily parse the information by using custom scripts or to display the information in a predefined format.

You can also specify the default output formatting of the **bhosts** and **bqueues** commands by specifying the **LSB_BHOSTS_FORMAT** and **LSB_BQUEUES_FORMAT** parameters in the `lsf.conf` file, or by specifying the **LSB_BHOSTS_FORMAT** and **LSB_QUEUES_FORMAT** environment variables.

See the information on how to customize host information output or how to customize queue information output in *Administering IBM Spectrum LSF*.

View customized information output in JSON format

With this release, you can view customized job, host, and queue information in JSON format by using the new `-json` command option with the **bjobs**, **bhosts**, and **bqueues** commands. Since JSON is a customized output format, you must use the `-json` option together with the `-o` option.

See the information on how to view customized host information in JSON format or how to view customized queue information in JSON format in *Administering IBM Spectrum LSF*.

View time in customized job information output in hh:mm:ss format

You can now view times in customized job information in `hh:mm:ss` format by using the new `-hms` command option with the **bjobs** command. Since the `hh:mm:ss` time format is a customized output format, you must use the `-hms` option together with the `-o` or `-o -json` command options.

You can also enable the `hh:mm:ss` time format as the default time format for customized job information by specifying the **LSB_HMS_TIME_FORMAT** parameter in the `lsf.conf` file, or by specifying the **LSB_HMS_TIME_FORMAT** environment variable.

If these parameters or options are not set, the default output time for customized output is in seconds.

See more information on the `-hms` option for the **bjobs** command in the *IBM Spectrum LSF Command Reference*.

See more information on the **LSB_HMS_TIME_FORMAT** parameter in the `lsf.conf` file in the *IBM Spectrum LSF Configuration Reference*.

Security

The following new features affect cluster security.

Improve security and authentication by updating the eauth executable file

LSF now includes an updated version of the **eauth** executable file that automatically generates a site-specific internal key by using 128-bit AES encryption. To use this updated version, you must replace the original **eauth** executable file with the new file.

See more information about how to update the **eauth** executable file in *Administering IBM Spectrum LSF*.

What's new in IBM Spectrum LSF Version 10.1 Fix Pack 1

The following topics summarize the new and changed behavior in LSF 10.1 Fix Pack 1

Release date: November 2016

Simplified affinity requirement syntax

Job submission with affinity requirements for LSF jobs is simplified. An **esub** script that is named `esub.p8aff` is provided to generate optimal affinity requirements based on the input requirements about the submitted affinity jobs. In addition, LSF supports OpenMP thread affinity in the **blaunch** distributed application framework. LSF MPI distributions must integrate with LSF to enable the OpenMP thread affinity.

For the generated affinity requirements, LSF tries to reduce the risk of CPU bottlenecks for the CPU allocation in LSF MPI task and OpenMP thread levels.

For more information, see Submit jobs with affinity resource requirements on IBM POWER8 systems.

bsub and bmod commands export memory and swap values as esub variables

Specifying `mem` and `swp` values in an `rusage[]` string tell LSF how much memory and swap space a job requires, but these values do no limit job resource usage.

The **bsub** and **bmod** commands can export `mem` and `swp` values in the `rusage[]` string to corresponding environment variables for **esub**. You can use these environment variables in your own **esub** to match memory and swap limits with the values in the `rusage[]` string. You also can configure your **esub** to check whether the memory and swap resources are correctly defined for the corresponding limits for the job, queue, or application. If the resources are not correctly defined, LSF rejects the job.

The following environment variables are exported:

- If the **bsub** or **bmod** command has a `mem` value in the `rusage[]` string, the **LSB_SUB_MEM_USAGE** variable is set to the `mem` value in the temporary **esub** parameter file that the **LSB_SUB_PARAM_FILE** environment variable points to. For example, if the **bsub** command has the option `-R "rusage[mem=512]"`, the **LSB_SUB_MEM_USAGE=512** variable is set in the temporary file.
- If the **bsub** or **bmod** command has a `swp` value in the `rusage[]` string, the **LSB_SUB_SWP_USAGE** variable is set to the `mem` value in the temporary **esub** parameter file that the **LSB_SUB_PARAM_FILE** environment variable points to. For example, if the **bsub** command has the option `-R "rusage[swp=1024]"`, the **LSB_SUB_SWP_USAGE=1024** variable is set in the temporary file.

For more information on **LSB_SUB_MEM_USAGE** or **LSB_SUB_SWP_USAGE**, see Configuration to enable job submission and execution controls.

Allow queues to ignore RETAIN and DURATION loan policies

The **LOAN_POLICIES** parameter in the `lsb.resources` file allows other jobs to borrow unused guaranteed resources LSF. You can enable queues to ignore the **RETAIN** and **DURATION** loan policies when LSF determines whether jobs in those queues can borrow unused guaranteed resources. To enable the queue to ignore the **RETAIN** and **DURATION** loan policies, specify an exclamation point (!) before the queue name in the **LOAN_POLICIES** parameter definition.

For more information, see Loaning resources from a pool.

Running LSF jobs in Docker containers

The Docker integration allows LSF to run jobs in Docker containers on demand. LSF manages the entire lifecycle of jobs that run in the container as common jobs.

LSF supports the use of Docker Engine, Version 1.12, or later, which must be installed on an LSF server host.

For more information, see IBM Spectrum LSF with Docker.

Running LSF jobs in Amazon Web Services instances

You can configure LSF to make allocation requests on from Amazon Web Services (AWS). With AWS configured as a resource provider in LSF resource connector, LSF can launch instances from AWS to satisfy pending workload. The AWS instances join the LSF cluster, and are terminated when they become idle.

LSF resource connector with AWS was tested on the following systems:

- LSF10.1 master host - Linux x86 Kernel 3.10, glibc 2.17 RHEL 7.x
- VMs - Linux x86 Kernel 3.10, glibc 2.17 CentOS 7.x

LSF resource connector with AWS is assumed to work on the following systems:

- IBM Spectrum LSF10.1
- Linux x86 Kernel 2.6, glibc 2.5 RHEL 5.x
- Linux x86 Kernel 2.6, glibc 2.11 RHEL 6.x
- Linux x86 Kernel 3.0, glibc 2.11 SLES 11.x
- Linux x86 Kernel 3.11, glibc 2.18 SLES 12.x
- Linux x86 Kernel 4.4, glibc 2.23 Ubuntu 16.04 LTS

For more information, see Using the IBM Spectrum LSF Resource Connector.

Job array performance enhancements

The performance of job array scheduling and execution is improved.

The performance of scheduling, dispatch, and execution of job array elements is affected when array elements are split from their original submitted array under various conditions. For example, if rerunnable array elements are dispatched but fail to run, the elements return to pending state. The LSF scheduler has already split these elements when job was dispatched to execution hosts. The split array elements can remain pending for an excessive amount of time.

For an array jobs with dependency conditions, LSF publishes separate job ready events to the scheduler for each element when the condition is satisfied. The scheduler splits the elements when it handles the job ready events.

The following performance improvements are made:

- Optimized recovery performance in the scheduler for jobs with many separate array elements.
- Improved handling of satisfied dependency conditions for array jobs.
- Improved dependency checking for array jobs to reduce the number of job ready events that are published to the scheduler.

- Improved the processing of events for multiple array elements for job ready event handling.
- Optimized event handling performance in the scheduler for array jobs with many split elements
- Improved handing job stop and resume, and events associated with moving jobs to the top and bottom of the queue with the **bbot** and **btop** commands.

New platform support

LSF supports the following platforms:

- Intel Knights Landing (Linux x86-64 packages)

What's new in IBM Spectrum LSF Version 10.1

The following topics summarize the new and changed behavior in LSF 10.1.

Release date: June 2016

Important: IBM Platform Computing is now renamed to IBM Spectrum Computing to complement IBM's Spectrum Storage family of software-defined offerings. The IBM Platform LSF product is now IBM Spectrum LSF. Some LSF documentation in IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0) does not yet reflect this new product name.

Performance enhancements

The following are the new features in LSF 10.1 that can improve performance.

General performance improvements

Scheduler efficiency

LSF 10.1 includes several binary-level and algorithm-level optimizations to help the scheduler to make faster decisions. These enhancements can make job scheduling less sensitive to the number of job buckets and resource requirement settings.

Daemon communication

LSF 10.1 makes optimizations to **mbatchd**/**sbatchd** communication protocols to ensure a dedicated channel to accelerate messages that are sent and received between the **mbatchd** and **sbatchd** daemons.

Improved scheduling for short jobs

LSF can now allow multiple jobs with common resource requirements to run consecutively on the same allocation. Whenever a job finishes, LSF attempts to quickly replace it with a pending job that has the same resource requirements. To ensure that limits are not violated, LSF selects pending jobs that belong to the same user and have other attributes in common.

Since LSF bypasses most of the standard scheduling logic between jobs, reusing resource allocation can help improve cluster utilization. This improvement is most evident in clusters with several shorter jobs (that is, jobs that run from a few seconds to several minutes) with the same resource requirements.

To ensure that the standard job prioritization policies are approximated, LSF enforces a limit on the length of time that each allocation is reusable. LSF automatically sets this time limit to achieve a high level of resource utilization. By

default, this reuse time cannot exceed 30 minutes. If you specify a maximum reuse time and an optional minimum reuse time with the **ALLOC_REUSE_DURATION** parameter, LSF adjusts the time limit within this specified range to achieve the highest level of resource utilization.

When jobs from job arrays reuse allocations, the dispatch order of these jobs might change. Dispatch order changes because jobs are chosen for allocation reuse based on submission time instead of other factors.

Advance reservations are not considered when the job allocation is reused. A job allocation that is placed on a host with advance reservations enabled cannot be reused. If an advance reservation is created on a host after the job allocation is already made, the allocation can still be reused until the reuse duration is expired or the job is suspended by the advance reservation policy.

To enable LSF to reuse the resource allocation, specify the **RELAX_JOB_DISPATCH_ORDER** parameter in the `lsb.params` file. To enable reuse for a specific queue, specify the **RELAX_JOB_DISPATCH_ORDER** parameter in the `lsb.queues` file. The **RELAX_JOB_DISPATCH_ORDER** parameter is now defined as `Y` at installation.

Use the **badmin perfmon view** command to show the number of jobs that are reordered as a result of this feature.

When the **RELAX_JOB_DISPATCH_ORDER** parameter is specified, changing job group limits is not supported.

Cluster performance improvement with job information cache

LSF has a new job information cache to reduce the load on the work directory file server. LSF caches job information such as job environment variables and data in memory from the command-line and **eexec** in a compressed format. If you have an environment with many commonly used environment variable settings, caching job information can improve job submission and job dispatch performance, especially when the work directory's shared file system is slow or at its limits.

The job information cache is enabled by default in LSF 10.1, and the default size of the `lsb.jobinfo.events` file is 1 GB. New job information is now stored in the new event file instead of individual job files.

The contents of the cache persist in the job information event file, which is located by default at `$LSB_SHAREDIR/cluster_name/logdir/lsb.jobinfo.events`. The location of the `lsb.jobinfo.events` file can be changed with the parameter **LSB_JOBINFO_DIR** in `lsf.conf`.

The amount of memory that is dedicated to the cache is controlled by the `lsb.params` parameter **JOB_INFO_MEMORY_CACHE_SIZE**.

As jobs are cleaned from the system, the `lsb.jobinfo.events` event file needs to be periodically rewritten to discard the unneeded data. By default, the job information event file is rewritten every 15 minutes. This interval can be changed with the parameter **JOB_INFO_EVENT_DUMP_INTERVAL** in the `lsb.params` file.

The values of the parameters **JOB_INFO_MEMORY_CACHE_SIZE** and **JOB_INFO_EVENT_DUMP_INTERVAL** can be viewed with the command **bparams -a** or **bparams -l**

The amount of memory that is used by the job information cache can be viewed with the command **badmin showstatus**.

Job array performance improvements

The algorithm that is used to process large job array operations is enhanced. The time to process multiple array elements in the **mbatchd** daemon and the scheduler is reduced. The processing of job array operations in the **mbatchd** daemon, log events, and publishing job events to the scheduler is more efficient. The performance and behavior of the **bmod**, **bkill**, **bresume**, **bstop**, **bswitch**, **btop**, and **bbot** commands has been improved.

The parameter **JOB_ARRAY_EVENTS_COMBINE** in the `lsb.params` file enables the performance improvements for array jobs. The formats of some event types are changed to include new fields in `lsb.events`, `lsb.acct`, `lsb.stream`, and `lsb.status` files.

The parameter **JOB_ARRAY_EVENTS_COMBINE** makes the parameter **JOB_SWITCH2_EVENT** in the `lsb.params` file obsolete.

Pending job management

The following new features improve the management of pending jobs.

Single pending reason

Previously, a main pending reason or a series of host-based pending reasons was given when a job cannot run. The main reason is given if the job is pending for a reason that is not related to single hosts before or during scheduling, or if it failed to dispatch or run on the allocated host after scheduling. If the job is eligible to be scheduled but no host can be allocated, the pending reason is host-based for every host, to indicate why the host cannot be used. However, this pending reason might mean that the host-based pending reasons are numerous and shown in any random order, making it difficult for users to decipher why their job does not run. This problem is especially true for large clusters.

To make the given pending reason both precise and succinct, this release introduces the option to choose a single key reason for why the job is pending. Host-based pending reasons are classified into categories, and only the top reason in the top category is shown, or a main pending reason.

Host-based pending reasons are now grouped into reasons of candidate hosts and reasons of non-candidate hosts. Reasons for non-candidate hosts are not important to users since they cannot act on them. For example, the reason `Not specified in job submission` might be given for a host that was filtered out by the user with the **bsub -m** command. In contrast, reasons for candidate hosts can be used by the user to get the job to run. For example, with the reason `Job's resource requirement for reserving resource (mem) not satisfied`, you can lower the job's memory requirement.

The new option **bjobs -p1** is introduced in this release to retrieve the single reason for a job. If the single key pending reason is a host-based reason, then the single reason and the corresponding number of hosts is shown. Otherwise, only the single reason is shown.

Note: If the main reason is the only host-based reason, the main reason is shown as the output of the **bjobs -p2** and **bjobs -p3** commands.

Categorized host-based pending reasons

To give users a better understanding of why their jobs are not running, and what they can do about it, LSF groups host-based pending reasons into two categories: reasons of candidate hosts, and reason of non-candidate hosts.

The new options **bjobs -p2** and **bjobs -p3** are introduced in this release.

Option **bjobs -p2** shows the total number of hosts in the cluster and the total number considered. For the hosts considered, the actual reason on each host is shown. For each pending reason, the number of hosts that give that reason is shown. The actual reason messages appear from most to least common.

Option **bjobs -p3** shows the total number of hosts in the cluster and the total number of candidate and non-candidate hosts. For both the candidate and non-candidate hosts, the actual pending reason on each host is shown. For each pending reason, the number of hosts that show that reason is given. The actual reason messages appear from most to least common.

Note: If the main reason is the only host-based reason, the main reason is shown as the output of the **bjobs -p2** and **bjobs -p3** commands.

bjobs -o "pend_reason"

Many customers use the **bjobs –u all** or **bjobs –l –u all** commands to get all information, then use a script to search through the output for the required data. The command **bjobs –o 'fmtspec'** also allows users to request just the fields that they want, and format them so that they are readily consumable.

With the continuing effort to enhance pending reasons, the new field **pend_reason** is introduced in this release to show the single (main) pending reason, including custom messages.

Configurable pending reason message and resource priority with the lsb.reasons file

This release introduces the ability to individually configure pending reason messages. Administrators can make messages clear to inform users on which action they can take to make the job run. Configure custom pending reasons in the new configuration file, **config/lsbatch/<*cluster_name*>/configdir/lsb.reasons**.

Detailed pending reasons

Reasons for why a job is pending are displayed by using the **bjobs** command, but in many cases the **bjobs** command provides only general messages for why the job is pending. The reasons do not include enough details and users might not know how to proceed. For example, the pending reason `The specified job group has reached its job limit` does not clarify which job group limit within the hierarchical tree is at its limit.

Greater detail is added to pending reason messages. Display includes, where applicable, host names, queue names, job group names, user group names, limit name, and limit value as part of the pending reason message.

The enhanced pending reason information is shown by the **bjobs** command with the -p1, -p2, and -p3 options. If the **LSB_BJOBS_PENDREASON_LEVEL** parameter in the lsf.conf file is set to 1, 2, or 3, the new information is shown by the **bjobs -p** command. The pending reason information is not included for the **bjobs -p0** command.

Pending reason summary

A new option, -psum, is introduced to the **bjobs** command. The -psum option displays a summary of current pending reasons. It displays the summarized number of jobs, hosts, and occurrences for each pending reason.

It can be used with the filter options that return a list of pending jobs: -p, -p(0~3), -pi, -pe, -q, -u, -G, -g, -app, -fwd, -J, -Jd, -P, -Lp, -sla, -m

The command bjobs -psum lists the top eligible and ineligible pending reasons in descending order by the number of jobs. If a host reason exists, further detailed host reasons are displayed in descending order by occurrences. Occurrence is a per-job per-host based number, counting the total times that each job hits the reason on every host.

Pending reason performance improvements

With this release, performance problems that are associated with displaying pending reasons are improved. Now, reasons for all jobs in a bucket are published (instead of only the top jobs in the bucket) at every interval that is specified by the **PEND_REASON_UPDATE_INTERVAL** parameter in the lsb.params file. Host-based reasons publishing performance is improved to support up to 20,000 buckets and 7,500 hosts without the need to enable the **CONDENSE_PENDING_REASONS** parameter or to use the **badmin diagnose** command.

Job start time estimation

In clusters with long running parallel jobs (such as HPC environments), a few long running jobs (that is, 100 - 1000 jobs) might be pending in the queue for several days. These jobs might run for several days or weeks.

LSF can now predict an approximate start time for these pending jobs by using a simulation-based job start time estimator that runs on the master host and is triggered by the **mbatchd** daemon. The estimator uses a snapshot of the cluster (including the running jobs and available resources in the cluster) to simulate job scheduling behavior. The estimator determines when jobs finish and the pending jobs start. This snapshot gives users an idea of when their jobs are expected to start.

To use simulation-based estimation to predict start times, jobs must be submitted with either a runtime limit (by using the **bsub -W** option or by submitting to a queue or application profile with a defined **RUNLIMIT** value) or an estimated run time (by using the **bsub -We** option or by submitting to an application profile with a defined **RUNTIME** value). LSF considers jobs without a runtime limit or an estimated run time as never finished after they are dispatched to the simulation-based estimator. If both a runtime limit and an estimated run time are specified for a job, the smaller value is used as the job's run time in the simulation-based estimator.

To enable the simulation-based estimator, define the **LSB_ENABLE_ESTIMATION=Y** parameter in the `lsf.conf` file. When **LSB_ENABLE_ESTIMATION=Y** is set, the estimator starts up 5 minutes after the **mbatchd** daemon starts or restarts. By default, the estimator provides predictions for the first 1000 jobs or for predicted start times up to one week in the future, whichever comes first. Estimation also ends when all pending jobs have prediction job start times.

Optionally, you can control the default values for when **mbatchd** stops the current round of estimation to balance the accuracy of the job start predictions against the computation effort on the master host. **mbatchd** stops the current round of estimation when the estimator reaches any one of the following estimation thresholds that are specified in `lsb.params`:

ESTIMATOR_MAX_JOBS_PREDICTION
: Specifies the number of pending jobs that the estimator predicts, which is 1000 by default.

ESTIMATOR_MAX_TIME_PREDICTION
: Specifies the amount of time into the future, in minutes, that a job is predicted to start before the estimator stops the current round of estimation. By default, the estimator stops after a job is predicted to start in one week (10080 minutes).

ESTIMATOR_MAX_RUNTIME_PREDICTION
: Specifies the amount of time that the estimator runs, up to the value of the **ESTIMATOR_SIM_START_INTERVAL** parameter. By default, the estimator stops after it runs for 30 minutes or the amount of time as specified by the **ESTIMATOR_SIM_START_INTERVAL** parameter, whichever is smaller.

The estimator does not support the following **badmin** subcommands: **mbddebug**, **schddebug**, **mbdtime**, and **schdtime**. The estimator reloads the configurations from the `lsf.conf` file after it starts.

Eligible and ineligible pending jobs

LSF can now determine whether pending jobs are eligible or ineligible for scheduling.

A job that is in an eligible pending state is a job that LSF would normally select for resource allocation, but is pending because its priority is lower than other jobs. It is a job that is eligible for scheduling and runs if sufficient resources are available to run it.

An ineligible pending job is ineligible for scheduling and remains pending even if enough resources are available to run it. A job can remain pending and be ineligible to run for the following reasons:

- The job has a start time constraint (specified with the `-b` option)
- The job is suspended while it is pending (in a PSUSP state).
- The queue of the job is made inactive by the administrator or by its time window.
- The job's dependency conditions are not satisfied.
- The job cannot fit into the runtime window (**RUN_WINDOW** parameter)
- Delayed scheduling is enabled for the job (the **NEW_JOB_SCHED_DELAY** parameter is greater than zero)
- The job's queue or application profile does not exist.

A job that is not under any of the ineligible pending state conditions is treated as an eligible pending job. In addition, for chunk jobs in `WAIT` status, the time that is spent in the `WAIT` status is counted as eligible pending time.

If the **`TRACK_ELIGIBLE_PENDINFO`** parameter in the `lsb.params` file is set to `Y` or `y`, LSF determines which pending jobs are eligible or ineligible for scheduling. LSF uses the eligible pending time instead of total pending time to determine job priority for the following time-based scheduling policies:

- Automatic job priority escalation increases job priority of jobs that are in an eligible pending state instead of pending state for the specified period.
- For absolute priority scheduling (APS), the **`JPRIORITY`** subfactor for the APS priority calculation uses the amount of time that the job spends in an eligible pending state instead of the total pending time.

The **`mbschd`** daemon saves eligible and ineligible pending information to disk every 5 minutes. The eligible and ineligible pending information is recovered when the **`mbatchd`** daemon restarts. When the **`mbatchd`** daemon restarts, some ineligible pending time might be lost since it is recovered from the snapshot file, which is dumped periodically at set intervals. The lost time period is counted as eligible pending time under such conditions. To change this time interval, specify the **`ELIGIBLE_PENDINFO_SNAPSHOT_INTERVAL`** parameter, in minutes, in the `lsb.params` file.

Pending time limits

You can specify pending time limits and eligible pending time limits for jobs.

LSF sends the pending time limit and eligible pending time limit configurations to IBM Spectrum LSF RTM, which handles the alarm and triggered actions such as user notification. For example, RTM can notify the user who submitted the job and the LSF administrator, and take job control actions (for example, killing the job). LSF RTM compares the job's pending time to the pending time limit, and the eligible pending time to the eligible pending time limit. If the job is in a pending state or an eligible pending state for longer than these specified time limits, LSF RTM triggers the alarm and actions. This parameter works without LSF RTM, but LSF does not take any other alarm actions.

To specify a pending time limit or eligible pending time limit at the queue or application level, define the **`PEND_TIME_LIMIT`** or **`ELIGIBLE_PEND_TIME_LIMIT`** parameters in `lsb.queues` or `lsb.applications`. To specify the pending time limit or eligible pending time limit at the job level, use the `-ptl` or `-eptl` options for **`bsub`** and **`bmod`**:

- `PEND_TIME_LIMIT=[`*hour*`:]`*minute*
- `ELIGIBLE_PEND_TIME_LIMIT=[`*hour*`:]`*minute*
- `-ptl [`*hour*`:]`*minute*
- `-eptl [`*hour*`:]`*minute*

The pending or eligible pending time limits are in the form of [*hour*:]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The job-level time limits override the application-level time limits, and the application-level time limits override the queue-level time limits.

LSF does not take any alarm actions. However, LSF users and administrators can track the amount of time that jobs spend in pending or eligible pending states, and whether the jobs reach the pending time limits:

The `-l` option for **bjobs**, **bapp**, and **bqueues** show the job-, application-, and queue-level pending time limits (and eligible pending time limits).

To track the amount of time that current pending jobs spend in the pending and eligible pending states, and to see how much time is remaining before LSF sends an alarm notification, run the **bjobs -p -o** command to get customized output for pending jobs.

- Pending time limit

```
bjobs -p -o "id effective_plimit plimit_remain"
JOBID EFFECTIVE_PLIMIT PLIMIT_REMAIN
101   1800             -60
102   3600              60
```

- Eligible pending time limit

```
bjobs -p -o "id effective_eplimit eplimit_remain"
JOBID EFFECTIVE_EPLIMIT EPLIMIT_REMAIN
101   600              -60
102   900               60
```

The `EFFECTIVE_PLIMIT` and `EFFECTIVE_EPLIMIT` columns indicate the pending and eligible pending time limits for the job. The `PLIMIT_REMAIN` and `EPLIMIT_REMAIN` columns display the amount of time that remains before LSF sends an alarm notification. A negative number indicates that the time limit was reached and shows the amount of time since the limit was reached.

Job scheduling and execution

The following new features affect job scheduling and execution.

Global fairshare scheduling policy

Many LSF customers run clusters in geographic sites that are connected by LSF multicluster capability to maximize resource utilization and throughput. Most customers configure hierarchical fairshare to ensure resource fairness among projects and users. The same fairshare tree can be configured in all clusters for the same organization because users might be mobile and can log in to multiple clusters. But fairshare is local to each cluster and resource usage might be fair in the context of one cluster, but unfair from a more global perspective.

The LSF global fairshare scheduling policy divides the processing power of IBM Spectrum LSF multicluster capability (LSF multicluster capability) and the LSF/XL feature of IBM Spectrum LSF Advanced Edition among users. The global fairshare scheduling policy provides fair access to all resources, making it possible for every user to use the resources of multiple clusters according to their configured shares.

Global fairshare is supported in IBM Spectrum LSF Standard Edition and IBM Spectrum LSF Advanced Edition.

Global fairshare scheduling is based on queue-level user-based fairshare scheduling. LSF clusters that run in geographically separate sites that are connected by LSF multicluster capability can maximize resource utilization and throughput.

Global fairshare supports the following types of fairshare scheduling policies:

- Queue level user-based fairshare
- Cross-queue user-based fairshare
- Parallel fairshare

In cross-queue user-based fairshare policies, you configure the master queue as a participant of global fairshare. Participants can be any queues, users, or user groups that participate in the global fairshare policy. Configuring a slave queue as a participant is not needed, since it does not synchronize data for the global fairshare policy.

For parallel fairshare, LSF can consider the number of CPUs when you use global fairshare scheduling with parallel jobs.

Resource connector for LSF

The resource connector for LSF feature (also called "host factory") enables LSF clusters to borrow resources from supported resource providers (for example, enterprise grid orchestrator or OpenStack based on workload.

The resource connector generates requests for extra hosts from a resource provider and dispatches jobs to dynamic hosts that join the LSF cluster. When the resource provider needs to reclaim the hosts, the resource connector requeues the jobs that are running on the LSF hosts, shuts down LSF daemons, and releases the hosts back to the resource provider.

Use the **bsub** command to submit jobs that require hosts that are borrowed from resource provider. Use the **bhosts** command to monitor the status of borrowed hosts.

LSF with Apache Hadoop

The IBM Spectrum LSF integration with Apache Hadoop provides a connector script that allows users to submit Hadoop applications as regular LSF jobs.

Apache Hadoop ("Hadoop") is a framework for large-scale distributed data storage and processing on computer clusters that uses the Hadoop Distributed File System ("HDFS") for the data storage and MapReduce programming model for the data processing. Since MapReduce workloads might represent only a small fraction of overall workload, but typically requires their own stand-alone environment, MapReduce is difficult to support within traditional HPC clusters. However, HPC clusters typically use parallel file systems that are sufficient for initial MapReduce workloads, so you can run MapReduce workloads as regular parallel jobs that run in an HPC cluster environment. Use the IBM Spectrum LSF integration with Apache Hadoop to submit Hadoop MapReduce workloads as regular LSF parallel jobs.

To run your Hadoop application through LSF, submit it as an LSF job. After the LSF job starts to run, the **blaunch** command automatically provisions and monitors an open source Hadoop cluster within LSF allocated resources, then submits actual MapReduce workloads into this Hadoop cluster. Since each LSF Hadoop job has its own resource (cluster), the integration provides a multi-tenancy environment to allow multiple users to share the common pool of HPC cluster resources. LSF is able to collect resource usage of MapReduce workloads as normal LSF parallel jobs and has full control of the job lifecycle. After the job is complete, LSF shuts down the Hadoop cluster.

By default, the Apache Hadoop integration configures the Hadoop cluster with direct access to shared file systems and does not require HDFS. You can use existing file systems in your HPC cluster without having to immediately invest in a new file system. Through the existing shared file system, data can be stored in common share locations, which avoids the typical data stage-in and stage-out steps with HDFS.

LSF with Apache Spark

The IBM Spectrum LSF integration with Apache Spark provides connector scripts that allow users to submit Spark applications as regular LSF jobs.

Apache Spark ("Spark") is an in-memory cluster computing system for large-scale data processing. Based on Apache Hadoop ("Hadoop"), it provides high-level APIs in Java, Scala and Python, and an optimized engine that supports general execution graphs. It also provides various high-level tools, including Spark SQL for structured data processing, Spark Streaming for stream processing, and Mlib for machine learning.

Spark applications require distributed computed nodes, large memory, a high-speed network, and no file system dependencies, so Spark applications can run in a traditional HPC environment. Use the IBM Spectrum LSF integration with Apache Spark to take advantage of the comprehensive LSF scheduling policies to allocate resources for Spark applications. LSF tracks, monitors, and controls the job execution.

To run your Spark application through LSF, submit it as an LSF job, and the scheduler allocates resources according to the job's resource requirements, while the **blaunch** command starts a stand-alone Spark cluster. After the job is complete, LSF shuts down the Spark cluster.

Resizable jobs with resource requirements

LSF now allows the following resource requirements with resizable jobs:

- Alternative resource requirements
- Compound resource requirements
- Compute unit requirements

When you use the **bresize release** command to release slots from compound resource requirements, you can release only the slots that are represented by the last term of the compound resource requirement. To release slots in earlier terms, run **bresize release** repeatedly to release slots in subsequent last terms.

In addition, autoresizable jobs can now be submitted with compute unit resource requirements. The `maxcus` keyword is enforced across the job's entire allocation as it grows, while the `balance` and `usablecuslots` keywords apply only to the initial resource allocation.

For example,

- `bsub -n 11,60 -R "cu[maxcus=2:type=enclosure]" -app resizable -ar myjob`

  An autoresizable job that spans the fewest possible compute units for a total allocation of at least 11 slots that use at most 2 compute units of type enclosure. If the autoresizable job grows, the entire job still uses at most 2 compute units of type enclosure.

- `bsub -n 64 -R "cu[balance:maxcus=4:type=enclosure]" -app resizable -ar myjob`

  An autoresizable job that spans the fewest possible compute units for a balanced allocation of 64 slots that use 4 or less compute units of type enclosure. If the autoresizable job grows, each subsequent allocation is a balanced allocation. The entire job (that is, the total of the initial and subsequent job allocations) still uses at most 4 compute units of type enclosure, but the job as a whole might not be a balanced allocation.
- `bsub -n 64 -R "cu[excl:maxcus=8:usablecuslots=10]" -app resizable -ar myjob`

  An autoresizable job that allocates 64 slots over 8 or less compute units in groups of 10 or more slots per compute unit. One compute unit possibly uses fewer than 10 slots. If the autoresizable job grows, each subsequent allocation allocates in groups of 10 or more slots per compute unit (with one compute unit possible using fewer than 10 slots). The entire job (that is, the total of the initial and subsequent job allocations) still uses at most 8 compute units. Since each subsequent allocation might have one compute unit that uses fewer than 10 slots, the entire job might have more than one compute unit that uses fewer than 10 slots. The default compute unit type set in the **COMPUTE_UNIT_TYPES** parameter is used, and is used exclusively by `myjob`.

Specifying compute unit order by host preference

Previously, the compute unit order was determined only by the compute unit `pref` policies (`cu[pref=config | maxavail | minavail]`). Host preference (specified by `-m` or the **HOSTS** parameter in the `lsb.queues` file) only affected the host order within each compute unit. This release allows the user to specify compute unit order in a more flexible manner, by host preference. LSF now allows use of the host preference to specify compute unit order along with the `cu[pref=config | maxavail | minavail]` policy.

The following example illustrates use of the `-m` preference to specify compute unit order as `cu1>cu2>cu3>cu4`

```
bsub -n 2 -m "cu1+10 cu2+5 cu3+1 cu4" -R "cu[]" ./app
```

Sorting forwarded jobs by submission time

The parameter **MC_SORT_BY_SUBMIT_TIME** is added to the `lsb.params` file. Enabling this parameter in a IBM Spectrum LSF multicluster capability environment allows forwarded jobs on the execution cluster to be sorted and run based on their original submission time (instead of their forwarded time). When the maximum rescheduled time is reached, pending jobs are rescheduled on the execution cluster. Pending jobs are ordered based on their original submission time (the time when the job was first submitted on the submission cluster) and not the forwarding time (the time when the job was reforwarded to the execution cluster).

Compute unit feature functions with the alternative and compound resource requirements

This release now supports compute unit (`cu`) strings in alternative and compound resource requirements except you use the `cu` keywords `excl` or `balance`. Other `cu` keywords (such as `type`, `pref`, `maxcus`, or `usablecuslot`) are fully supported. Jobs are rejected if the merged result of the queue-, application-, and job-level resource requirement is compound or alternative with `cu[excl]` or `cu[balance]`.

External post-submission with epsub

Using the same mechanism for external job submission executable files (**esub**), you can now specify post-submission executable files to run after a job is submitted. An **epsub** is an executable file that you write to meet the post-submission job requirements at your site with information that is not available before job submission. The following are some of the things that you can use an **epsub** to do:

- Pass job information to an external entity
- Post job information to a local log file
- Perform general logic after a job is submitted to LSF

When a user submits a job by using **bsub**, and modifies a job by using the **bmod** command, or restarts a job by using the **brestart** command, LSF runs the **epsub** executable files on the submission host immediately after the job is accepted. The job might or might not be running while **epsub** is running.

For interactive jobs, **bsub** or **bmod** runs **epsub**, then resumes regular interactive job behavior (that is, **bsub** or **bmod** runs **epsub**, then runs the interactive job).

The **epsub** file does not pass information to **eexec**, nor does it get information from **eexec**. **epsub** can read information only from the temporary file that contains job submission options (as indicated by the **LSB_SUB_PARM_FILE** environment variable) and from the environment variables. The following information is available to the **epsub** after job submission:

- A temporary file that contains job submission options, which are available through the **LSB_SUB_PARM_FILE** environment variable. The file that this environment variable specifies is a different file from the one that is initially created by **esub** before the job submission.
- The LSF job ID, which is available through the **LSB_SUB_JOB_ID** environment variable. For job arrays, the job ID includes the job array index.
- The name of the final queue to which the job is submitted (including any queue modifications that are made by **esub**), which is available through the **LSB_SUB_JOB_QUEUE** environment variable.
- The LSF job error number if the job submission failed, which is available through the **LSB_SUB_JOB_ERR** environment variable.

If the **esub** rejects a job, the corresponding **epsub** file does not run.

After job submission, the **bsub** or **bmod** command waits for the **epsub** scripts to finish before it returns. If the **bsub** or **bmod** return time is crucial, do not use **epsub** to perform time-consuming activities. In addition, if **epsub** hangs, **bsub** or **bmod** waits indefinitely for the **epsub** script to finish. This behavior is similar to the **esub** behavior because **bsub** or **bmod** hangs if an **esub** script hangs.

If an LSF administrator specifies one or more mandatory **esub**/**epsub** executable files that use the parameter **LSB_ESUB_METHOD**, LSF starts the corresponding mandatory **epsub** executable files (as specified by using the parameter **LSB_ESUB_METHOD**), followed by any application-specific **epsub** executable files (with *.application_name* in the file name).

If a mandatory program that is specified by the **LSB_ESUB_METHOD** parameter does not have a corresponding **esub** executable file (esub.*application_name*), but has a

corresponding **epsub** executable file (epsub.*application_name*), the job is submitted normally by using the normal external job submission and post-submission mechanisms.

Except for these differences, **epsub** uses the same framework as **esub**.

Save a snapshot of the job scheduler buckets

LSF can now save a snapshot of the current contents of the scheduling buckets to help administrators diagnose problems with the scheduler. Jobs are put into scheduling buckets based on resource requirements and different scheduling policies. Saving the contents into a snapshot file is useful for data analysis by parsing the file or by performing a simple text search on its contents.

This feature is helpful if you want to examine a sudden large performance impact on the scheduler. Use the snapshot file to identify any users with many buckets or large attribute values.

To use this feature, run the **badmin diagnose -c jobreq** command.

This feature enables **mbschd** to write an active image of the scheduler job buckets into a snapshot file as raw data in XML or JSON format. A maximum of one snapshot file is generated in each scheduling cycle.

Use the -f option to specify a custom file name and path and the -t option to specify whether the file is in XML or JSON format.

By default, the name of the snapshot file is jobreq_<*hostname*>_<*dateandtime*>.<*format*>, where <*format*> is xml or json, depending on the specified format of the snapshot file. By default, the snapshot file is saved to the location specified in the **DIAGNOSE_LOGDIR** parameter.

Using logging threads to log messages

The **mbatchd** and **mbschd** daemons now use dedicated threads to write messages to the log files. Using dedicated threads reduces the impact of logging messages on the performance of **mbatchd** and **mbschd**.

Define the LSF_LOG_QUEUE_SIZE=*integer* parameter in the lsf.conf file as an integer between 100 and 500000 to specify the maximum size of the logging queue. The logging queue, which contains the messages to be logged in the log files, is full when the number of entries reaches this number.

Define the **LSF_DISCARD_LOG** parameter in the lsf.conf file to specify the behavior of the logging thread if the logging queue is full. If set to Y, the logging thread discards all new messages at a level lower than **LOG_WARNING** when the logging queue is full. LSF logs a summary of the discarded messages later.

If the **LSF_DISCARD_LOG** parameter is set to N, LSF automatically extends the size of the logging queue if the logging queue is full.

Specifying resource requirements for stopped checkpointable jobs

The **brestart** command now includes the -R option to reserve resources when you restart a stopped checkpointable job. You can specify resources with **brestart -R**

when you restart the job. Specify multiple `-R` options on the **brestart** command for multiple resource requirement strings, compound resource requirements, and alternative resource requirements.

For example, if you submitted the following checkpointable job:

```
bsub -R "select[mem>100] rusage[mem=100]" -M 100 myjob
```

You can restart this checkpointable job by using the **brestart -R** command to specify a new resource requirement:

```
brestart -R "select[mem>5000] rusage[mem=5000]" -M 5000 checkpointdir/pid
```

No size limitations for resource requirement strings

LSF no longer has any size limitations on resource requirement strings. Previously, resource requirement strings were restricted to 512 bytes. You can now submit resource requirement strings with the `-R` option with no limitations on the length of the string.

You must upgrade all hosts in the cluster to LSF 10.1 to submit resource requirement strings with no size limitations. If hosts in the cluster still run earlier versions of LSF, resource requirement strings still have the following limitations:

- In the IBM Spectrum LSF multicluster capability job forwarding mode, if the execution cluster is running an earlier version of LSF:
  - Any jobs with a job-level resource requirement string that is longer than 511 bytes remain pending on the submission cluster.
  - LSF rejects any **bmod** commands that modify a job that is forwarded to the execution cluster with a job-level resource requirement string that is longer than 511 bytes.
- If you run the **bjobs** command from a host with an earlier version of LSF and the job-level resource requirement string is longer than 4096 bytes, the **bjobs -l** command output shows a truncated resource requirement string.
- If you run the **bacct** or **bhist** commands from a host with an earlier version of LSF and the effective resource requirement string is longer than 4096 bytes, the command might fail.

Host-related features

The following new features are related to host management and display.

Condensed host format

When you specify host names or host groups with condensed notation, you can now use colons (`:`) to specify a range of numbers. Colons are used the same as hyphens (`-`) are currently used to specify ranges and can be used interchangeably in condensed notation. You can also use leading zeros to specify host names.

You can now use multiple square brackets (with the supported special characters) to define multiple sets of non-negative integers anywhere in the host name. For example, `hostA[1,3]B[1-3]` includes `hostA1B1`, `hostA1B2`, `hostA1B3`, `hostA3B1`, `hostA3B2`, and `hostA3B3`.

The additions to the condensed notation apply to all cases where you can specify condensed notation, including commands that use the `-m` option or a host list to specify multiple host names, the `lsf.cluster.`*`clustername`* file (in `HOSTNAME` column

of the Hosts section), and the lsb.hosts file (in the HOST_NAME column of the Host section, the GROUP_MEMBER column of the HostGroup section, and the MEMBER column of the ComputeUnit section).

For example, submit a job by using the **bsub -m** command.

- bsub -m "host[1-100].example.com"

  The job is submitted to host1.example.com, host2.example.com, host3.example.com, all the way to host100.example.com.
- bsub -m "host[01-03].example.com"

  The job is submitted to host01.example.com, host02.example.com, and host03.example.com.
- bsub -m "host[5:200].example.com"

  The job is submitted to host5.example.com, host6.example.com, host7.example.com, all the way to host200.example.com.
- bsub -m "host[05:09].example.com"

  The job is submitted to host05.example.com, host06.example.com, all the way to host09.example.com.
- bsub -m "host[1-10,12,20-25].example.com"

  The job is submitted to host1.example.com, host2.example.com, host3.example.com, up to and including host10.example.com. It is also submitted to host12.example.com and the hosts between and including host20.example.com and host25.example.com.
- bsub -m "host[1:10,20,30:39].example.com"

  The job is submitted to host1.example.com, host2.example.com, host3.example.com, up to and including host10.example.com. It is also submitted to host20.example.com and the hosts between and including host30.example.com and host39.example.com.
- bsub -m "host[10-20,30,40:50].example.com"

  The job is submitted to host10.example.com, host11.example.com, host12.example.com, up to and including host20.example.com. It is also submitted to host30.example.com and the hosts between and including host40.example.com and host50.example.com.
- bsub -m "host[01-03,05,07:09].example.com"

  The job is submitted to host01.example.com, up to and including host03.example.com. It is also submitted to host05.example.com, and the hosts between and includinghost07.example.com and host09.example.com.
- bsub -m "hostA[1-2]B[1-3,5].example.com"

  The job is submitted to hostA1B1.example.com, hostA1B2.example.com, hostA1B3.example.com, hostA1B5.example.com, hostA2B1.example.com, hostA2B2.example.com, hostA2B3.example.com, and hostA2B5.example.com.

Register LSF host names and IP addresses to LSF servers

You can now register the IP and host name of your local LSF host with LSF servers so that LSF does not need to use the DNS server to resolve your local host. This addresses previous issues of resolving the host name and IP address of LSF hosts with non-static IP addresses in environments where the DNS server is not able to properly resolve these hosts after their IP addresses change.

To enable host registration, specify LSF_REG_FLOAT_HOSTS=Y in the lsf.conf file on each LSF server, or on one LSF server if all servers have access to the **LSB_SHAREDIR**

directory. This parameter enables LSF daemons to look for records in the `reghostscache` file when it attempts to look up host names or IP addresses.

By default, the `reghostscache` file is stored in the file path as defined by the **LSB_SHAREDIR** parameter in the `lsf.conf` file. Define the **LSB_SHAREDIR** parameter so that the `reghostscache` file can be shared with as many LSF servers as possible. For all LSF servers that have access to the shared directory defined by the **LSB_SHAREDIR** parameter, only one of these servers needs to receive the registration request from the local host. The `reghostscache` file reduces network load by reducing the number of servers to which the registration request must be sent. If all hosts in the cluster can access the shared directory, the registration needs to be sent only to the master LIM. The master LIM records the host information in the shared `reghostscache` file that all other servers can access. If the **LSB_SHAREDIR** parameter is not defined, the `reghostscache` file is placed in the **LSF_TOP** directory.

The following example is a typical record in the `reghostscache` file:

```
MyHost1    192.168.1.2    S-1-5-21-5615612300-9789239785-9879786971
```

Windows hosts that register have their computer SID included as part of the record. If a registration request is received from an already registered host, but its SID does not match with the corresponding record's SID in the `reghostscache` file. This new registration request is rejected, which prevents malicious hosts from imitating another host's name and registering itself as another host.

After you enable host registration, you can register LSF hosts by running the **lsreghost** command from the local host. Specify a path to the `hostregsetup` file:

- On UNIX, **lsreghost -s** *file_path*/**hostregsetup**

  You must run the UNIX command with root privileges. If you want to register the local host at regular intervals, set up a cron job to run this command.
- On Windows, **lsreghost -i** *file_path***hostregsetup**

  The Windows command installs **lsreghost** as a Windows service that automatically starts up when the host starts up.

The `hostregsetup` file is a text file with the names of the LSF servers to which the local host must register itself. Each line in the file contains the host name of one LSF server. Empty lines and `#comment` text are ignored.

The bmgroup command displays leased-in hosts in the resource leasing model for IBM Spectrum LSF multicluster capability

The **bmgroup** command displays compute units, host groups, host names, and administrators for each group or unit. For the resource leasing model, host groups with leased-in hosts are displayed by default as `allremote` in the `HOSTS` column.

You can now expand the `allremote` keyword to display a list of the leased-in hosts in the host group with the **bmgroup**.

By default, the `HOSTS` column now displays a list of leased-in hosts in the form *host_name*@*cluster_name*.

For example, if `cluster_1` defined a host group that is called `master_hosts` that contains only `host_A`, and a host group that is called `remote_hosts` with leased-in hosts as members, and `cluster_2` contains `host_B` and `host_C` that are both being leased in by `cluster_1`:

By default, the HOSTS column displays a list of leased-in hosts:

```
GROUP_NAME   HOSTS
master_hosts host_A
remote_hosts host_B@cluster_2 host_C@cluster_2
```

If the **LSB_BMGROUP_ALLREMOTE_EXPAND=N** parameter is configured in the lsf.conf file or as an environment variable, leased-in hosts are represented by a single keyword allremote instead of being displayed as a list.

```
GROUP_NAME   HOSTS
master_hosts host_A
remote_hosts allremote
```

RUR job accounting replaces CSA for LSF on Cray

In the LSF integration with Cray Linux, Comprehensive System Accounting (CSA) is now deprecated and replaced with Resource Utility Reporting (RUR).

To modify the default RUR settings, edit the following parameters in the lsf.conf file:

LSF_CRAY_RUR_ACCOUNTING
: Specify N to disable RUR job accounting if RUR is not enabled in your Cray environment, or to increase performance. Default value is Y (enabled).

LSF_CRAY_RUR_DIR
: Location of the RUR data files, which is a shared file system that is accessible from any potential first execution host. Default value is LSF_SHARED_DIR/<*cluster_name*>/craylinux/<*cray_machine_name*>/rur.

LSF_CRAY_RUR_PROLOG_PATH
: File path to the RUR prolog script file. Default value is /opt/cray/rur/default/bin/rur_prologue.py.

LSF_CRAY_RUR_EPILOG_PATH
: File path to the RUR epilog script file. Default value is /opt/cray/rur/default/bin/rur_epilogue.py.

RUR does not support host-based resource usage (LSF_HPC_EXTENSIONS="HOST_RUSAGE").

The LSF administrator must enable RUR plug-ins, including output plug-ins, to ensure that the **LSF_CRAY_RUR_DIR** directory contains per-job accounting files (rur.<*job_id*>) or a flat file (rur.output).

Other changes to LSF behavior

See details about changes to default LSF behavior.

General LSF behavior

You cannot use the **bconf** command to define project limits when the cluster has no project limits set.

You cannot delete an advance reservation while jobs are still running in it.

If host preference is specified, compute unit preference is also determined by host preference. Before LSF 10.1, compute unit preference is determined only by the cu preference string (pref=config | maxavail | minavail).

The **JOB_SCHEDULING_INTERVAL** parameter in the `lsb.params` file now specifies the minimal interval between subsequent job scheduling sessions. Specify in seconds, or include the keyword `ms` to specify in milliseconds. If set to 0, subsequent sessions have no minimum interval between them. Previously, this parameter specified the amount of time that **mbschd** sleeps before it starts the next scheduling session.

The job information cache is enabled by default (the **JOB_INFO_MEMORY_CACHE_SIZE** parameter in the `lsb.params` file), and the default size of the `lsb.jobinfo.events` file is 1024 MB (1 GB). New job information is now stored in the new event file instead of individual job files.

The parameter **JOB_SWITCH2_EVENT** in the `lsb.params` file is obsolete in LSF 10.1 and later. To take advantage of enhancements to job array performance, set the **JOB_ARRAY_EVENTS_COMBINE=Y** parameter.

New event replay mechanism writes files to LSF_TMPDIR

On execution hosts, the **sbatchd** daemons write their events to a file under `LSF_TMPDIR` (the default directory is `/tmp`). If the LSF temporary directory becomes full, **sbatchd** cannot write to its event file, and the daemons do not recover normally. You must make sure to maintain enough free space in the `LSF_TMPDIR` directory.

Learn more about IBM Spectrum LSF

Information about IBM Spectrum LSF is available from several sources.

- The IBM Spectrum Computing website www.ibm.com/systems/spectrum-computing/
- The IBM Spectrum LSF product page www.ibm.com/systems/spectrum-computing/products/lsf/
- The LSF area of the IBM Support Portal www.ibm.com/systems/spectrum-computing/support.html
- IBM Spectrum Computing community on IBM developerWorks https://developer.ibm.com/storage/products/ibm-spectrum-lsf
- IBM Spectrum LSF documentation in IBM Knowledge Center www.ibm.com/support/knowledgecenter/SSWRJV

IBM Spectrum Computing community

Connect. Learn. Share. Collaborate and network with the IBM Spectrum Computing experts on IBM developerWorks at https://developer.ibm.com/storage/products/ibm-spectrum-lsf. Join today!

Use IBM developerWorks to learn, develop, and connect:

- Connect to become involved with an ongoing, open engagement among other users, system professionals, and IBM developers of IBM Spectrum Computing products.
- Learn more about IBM Spectrum Computing products on blogs and wikis, and benefit from the expertise and experience of others.
- Share your experience in wikis and forums to collaborate with the broader software defined computing user community.

Product notifications

Subscribe to product notifications on the My notifications page on the IBM Support website.

To receive information about product solution and patch updates automatically, go to the My notifications page on the IBM Support website: www.ibm.com/support/mynotifications. You can edit your subscription settings to choose the types of information you want to get notification about, for example, security bulletins, fixes, troubleshooting, and product enhancements or documentation changes.

IBM Spectrum LSF documentation

IBM Knowledge Center is the home for IBM Spectrum LSF product documentation.

LSF documentation on IBM Knowledge Center

Find the most up-to-date IBM Spectrum LSF documentation on IBM Knowledge Center on the IBM website: www.ibm.com/support/knowledgecenter/SSWRJV.

Search all the content in IBM Knowledge Center for subjects that interest you, or search within a product, or restrict your search to one version of a product. Sign in with your *IBMid* to take full advantage of the customization and personalization features available in IBM Knowledge Center.

Documentation available through IBM Knowledge Center is updated and regenerated frequently after the original release of IBM Spectrum LSF 10.1.

An installable offline version of the documentation is available in IBM Spectrum LSF Application Center Basic Edition, which is packaged with LSF.

We'd like to hear from you

For technical support, contact IBM or your LSF vendor. Or go to the IBM Support Portal: www.ibm.com/support

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

Log in to IBM Knowledge Center with your *IBMid*, and add your comments and feedback to any topic.

Product compatibility

The following sections detail compatibility information for version 10.1 of IBM Spectrum LSF.

Server host compatibility

LSF 9.1 or later servers are compatible with IBM Spectrum LSF 10.1 master hosts. All LSF 9.1 or later features are supported by IBM Spectrum LSF 10.1 master hosts.

Important: To take full advantage of all new features that are introduced in the latest release of IBM Spectrum LSF, you *must* upgrade all hosts in your cluster.

LSF add-on compatibility

IBM Spectrum LSF 10.1 is compatible with LSF family add-ons.

IBM Spectrum LSF RTM and IBM Platform RTM

You can use IBM Platform RTM 8.3 or later to collect data from IBM Spectrum LSF 10.1 clusters. When you add the cluster, select **Poller for LSF 8** or **Poller for LSF 9.1**.

IBM Spectrum LSF License Scheduler and IBM Platform LSF License Scheduler

IBM Platform LSF License Scheduler 8.3 or later are compatible with IBM Spectrum LSF 10.1.

IBM Spectrum LSF Process Manager and IBM Platform Process Manager

IBM Platform Process Manager 9.1 and later, and IBM Spectrum LSF Process Manager is compatible with IBM Spectrum LSF 10.1.

IBM Spectrum LSF Analytics and IBM Platform Analytics

If you use earlier versions of IBM Platform Analytics, do not enable the **JOB_ARRAY_EVENTS_COMBINE** parameter in the `lsb.params` file. The parameter introduces an event format that is not compatible with earlier versions of IBM Platform Analytics.

IBM Platform Analytics 9.1.2.2 is compatible with IBM Spectrum LSF 10.1.

IBM Spectrum LSF Application Center and IBM Platform Application Center

If you upgrade earlier versions of IBM Spectrum LSF to version 10.1, but you do not upgrade IBM Platform Application Center in an existing LSF cluster, IBM Platform Application Center 9.1.3 and later versions are compatible with IBM Spectrum LSF 10.1.

Install a new LSF 10.1 cluster before you install IBM Spectrum LSF Application Center 10.1 to avoid compatibility issues. Versions of IBM Spectrum LSF Application Center that are earlier than 9.1.3 are not compatible with LSF 10.1.

API compatibility

To take full advantage of new IBM Spectrum LSF 10.1 features, recompile your existing LSF applications with IBM Spectrum LSF 10.1.

You must rebuild your applications if they use APIs that changed in IBM Spectrum LSF 10.1.

New and changed Platform LSF APIs

The following APIs or data structures are changed or are new for LSF 10.1:

- struct _limitInfoReq
- struct _lsb_reasonConf

- struct _lsb_reasonMsgConf
- struct _lsb_rsrcConf
- struct _reasonRefEntry
- struct allLevelReasonMsg
- struct appInfoEnt
- struct estimationResults
- struct globalFairshareLoadEnt
- struct globalShareAcctEnt
- struct gpuRusage
- struct hostInfo
- struct hRusage
- struct jobArrayID
- struct jobArrayIndex
- struct jobCleanLog
- struct jobFinishLog
- struct jobFinish2Log
- struct jobForwardLog
- struct jobInfoEnt
- struct jobInfoHead
- struct jobInfoReq
- struct jobModLog
- struct jobMoveLog
- struct jobPendingSummary
- struct jobPendingSummaryElem
- struct jobStartLog
- struct jobStatusLog
- struct jobSwitchLog
- struct jobStatus2Log
- struct jRusage
- struct keyValue
- struct KVPair
- struct packSubmitReply
- struct parameterInfo
- struct participantShareLoad
- struct pendingReasonInfo
- struct perfmonLog
- struct queryInfo
- struct queueInfoEnt
- struct queueKVP
- struct reasonMessage
- struct reasonRefString
- struct reasonRefStrTab
- struct rmtJobCtrlRecord2
- struct sbdAsyncJobStatusReplyLog
- struct sbdAsyncJobStatusReqLog

- struct sbdJobStartAcceptLog
- struct sbdJobStatusLog
- struct shareLoadInfo
- struct signalLog
- struct slotInfoRequest
- struct statusInfo
- struct submit
- union eventLog
- API ls_eligible()

For detailed information about APIs changed or created for LSF 10.1, see the *IBM Spectrum LSF 10.1 API Reference*.

Third-party APIs

The following third-party APIs are tested and supported for this release:

- DRMAA LSF API v 1.1.1
- PERL LSF API v1.0
- Python LSF API v1.0 with LSF 9

Packages for these APIs are available at www.github.com.

For more information about using third-party APIs with LSF 10.1, see the IBM Spectrum Computing community on IBM developerWorks at https://developer.ibm.com/storage/products/ibm-spectrum-lsf.

IBM Spectrum LSF product packages

The IBM Spectrum LSF product consists of distribution packages for supported operating systems, installation packages, and entitlement files.

Supported operating systems

For detailed LSF operating system support information, refer to IBM Spectrum LSF System Requirements (www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/New%20IBM%20Platform%20LSF%20Wiki/page/System%20requirements) at the LSF product wiki on IBM developerWorks.

UNIX and Linux Installer packages

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

lsf10.1.0.6_lsfinstall.tar.Z
: The standard installer package. Use this package in a heterogeneous cluster with a mix of systems other than x86-64. Requires approximately 1 GB free space.

lsf10.1.0.6_lsfinstall_linux_x86_64.tar.Z
: Use this smaller installer package in a homogeneous x86-64 cluster. If you add other non-x86-64 hosts, you must use the standard installer package. Requires approximately 100 MB free space.

lsf10.1.0.6_no_jre_lsfinstall.tar.Z
For all platforms not requiring the JRE. JRE version 1.4 or higher must already be installed on the system. Requires approximately 1 MB free space.

lsf10.1.0.6_lsfinstall_linux_ppc64le.tar.Z

Installer package for Linux on IBM Power 6, 7, and 8 Little-Endian (LE) systems

Entitlement files

The following LSF entitlement configuration files are available:

LSF Standard Edition
`lsf_std_entitlement.dat`

LSF Express Edition
`lsf_exp_entitlement.dat`

LSF Advanced Edition
`lsf_adv_entitlement.dat`

Getting fixes from IBM Fix Central

After you install or upgrade LSF, use IBM Fix Central to find and download the fixes that are recommended by IBM Support for LSF products. From Fix Central, you can search, select, order, and download fix packs and interim fixes for your system with a choice of delivery options.

Before you download a fix from IBM Fix Central (www.ibm.com/support/fixcentral), have the following information at hand:

- Know your IBMid and password. You must log in to the Fix Central website before you can download a fix.
- If you know exactly which fix you need, you can search for it directly from the **Search Fix Central** field on the IBM Fix Central website.
- To get information about the download process, or help during the process, see Fix Central help (`www.ibm.com/systems/support/fixes/en/fixcentral/help/faq_sw.html`).

Note: Fix Packs are only available for the following systems:

- Linux 64-bit
- Linux x86_64
- Linux PPC64LE

Interim fixes are available for the systems that are affected by the fix.

1. On the Fix Central page, decide how you want to select the product information for the fix that you need:
   - Use the **Find product** tab to find fixes by product (for example, IBM Spectrum LSF).
   - Use the **Select product** tab to find fixes by product group (for example, IBM Spectrum Computing).
   a. On the **Find product** tab, enter `IBM Spectrum LSF` in the **Product selector** field.
   b. For **Installed Version**, select the version that is installed on your system. Select **All** to see all available versions.

c. For **Platform**, select the operating system that you run your IBM Spectrum LSF product on. Select **All** to see all available versions.

a. On the **Select product** tab, select **Product group** > **IBM Spectrum Computing**.

Tip:

If you searched for LSF family products before, they are conveniently listed in the **My product history** box.

b. Select your product from the **Product** list. For example, the core LSF product is **IBM Spectrum LSF**. Other IBM Spectrum LSF products, including the IBM Spectrum LSF suites, are listed in the **Select product** list.

c. For **Installed Version**, select the version that is installed on your system. Select **All** to see all available versions.

d. For **Platform**, select the operating system that you run your IBM Spectrum LSF product on. Select **All** to see all available versions.

2. On the Identify fixes page, specify how you want to search for the fix.
   - Browse all the fixes for the specified product, version, and operating system.
   - Enter the APAR or SPR numbers that you want to search for. Enter one or more APAR or SPR numbers, which are separated by a comma; for example, `P101887`.
   - Enter an individual fix ID. Search for updates by entering one or more fix IDs, each separated by a comma; for example, `lsf-10.1-build420903`.
   - Enter text for your search keywords, such as problem area, exception, or message ID, in any order, for example, `lsb_readjobinfo API`.
   - Search a list of the recommended fixes.

   For IBM Power Systems™ fixes, you can use the Fix Level Recommendation Tool (FLRT) (`www.ibm.com/support/customercare/flrt/`) to identify the fixes you want. This tool provides information about the minimum recommended fix levels and compatibility on the key components of IBM Power Systems running the AIX®, IBM i and Linux operating systems. FLRT is especially useful when you plan to upgrade the key components of your system, or you want to verify the current health of the system.
3. On the Select fixes page, browse the list of fixes for your product, version, and operating system.

   Tip: To find the latest fixes, sort the list of fixes by **Release date**.
   - Mark the check box next to any fix that you want to download.
   - To create a new query and a new list of fixes to choose from, clear the list of fixes and return to the Identify fixes page.
   - Filter the content of the Select fixes page by platform, fix status, version, or fix type.
4. On the Download options page, specify how you want to download the fix and any other required information.

   Click **Back** to change your download options.
5. Download the files that implement the fix.

   When you download the file, make sure that the name of the file is not changed. Do not change the name of the file yourself, and check that the web browsers or download utility did not inadvertently change the file name.
6. To apply the fix, follow the instructions in the readme file that is downloaded with the fix.

7. Optional: Subscribe to notifications about LSF fixes on Fix Central.

   To receive information about product solution and patch updates automatically, go to the My notifications page on the IBM Support website: (`www.ibm.com/support/my notifications`). You can edit your subscription settings to choose the types of information you want to get notification about, for example, security bulletins, fixes, troubleshooting, and product enhancements or documentation changes.

Bugs fixed

LSF Version 10.1 releases and Fix Packs contain bugs that were fixed since the general availability of LSF.

Fix Pack 6

LSF Version 10.1 Fix Pack 6 contains all bugs that were fixed before 24 May 2018.

Fix Pack 5

LSF Version 10.1 Fix Pack 5, which only applies to IBM POWER 9 platforms, contains all bugs that were fixed before 27 March 2018.

Fix Pack 4

LSF Version 10.1 Fix Pack 4 contains all bugs that were fixed before 20 November 2017.

Fix Pack 3

LSF Version 10.1 Fix Pack 3 contains all bugs that were fixed before 6 July 2017.

Fix Pack 2

LSF Version 10.1 Fix Pack 2 contains all bugs that were fixed before 15 February 2017.

Fix Pack 1

LSF Version 10.1 Fix Pack 1 contains all bugs that were fixed before 20 October 2016.

June 2016 release

The June 2016 release of LSF Version 10.1 contains all bugs that were fixed before 29 April 2016.

Lists of fixed bugs for all releases of LSF are available on IBM developerWorks on the LSF family wiki Troubleshooting page: http://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/New%20IBM%20Platform%20LSF%20Wiki/page/Troubleshooting.

Known issues

LSF 10.1 has the following known issues.

- On AIX, a TCL parser issue causes jobs to pend when the **LSF_STRICT_RESREQ=N** parameter is set in the `lsf.conf` file, even though AIX hosts are available. To avoid the problem, make sure that **LSF_STRICT_RESREQ=Y**.
- While running a job, a RedHat 7.2 server host may fail with the following error messages in the system log file or the system console:

  ```
  INFO: rcu_sched self-detected stall on CPU { number}
  INFO: rcu_sched detected stalls on CPUs/tasks:
  BUG: soft lockup - CPU#number stuck for time! [res:16462]
  ```

  This is an issue with RedHat 7.2 kernel-3.10.0-327.el7. To resolve this issue, download and apply a RedHat kernel security update. For more details, refer to https://rhn.redhat.com/errata/RHSA-2016-2098.html.

Limitations

LSF 10.1 has the following limitations.

Job start time prediction

Job start time prediction has limited support for guaranteed SLA. The estimator cannot schedule the jobs that borrow the resources in the guarantee pool. The estimator scheduler bypasses backfilling scheduling, which calls the guarantee reserve plug-in to schedule loan jobs.

GPU MPS solution

The MPS Server supports up to 16 client CUDA contexts concurrently. And this limitation is per user per job. That means MPS can handle at most 16 CUDA processes at one time even though LSF allocated multiple GPUs.

Registering dynamic LSF host IP address or name into master LIM

In shared LSF environments that frequently change IP addresses, client hosts need to register with the master host only. If client hosts do not register, the cache file is overwritten by other LIM hosts and the cache file becomes inaccurate. Windows client hosts with the same IP address and a new SID, administrators must manually remove old records from cache file and restart the master LIM to reregister.

Simplified affinity requirement syntax

The **esub.p8aff** script cannot modify the environment variables when called by the **bmod** command. The **SMT** argument (the **OMP_NUM_THREADS** environment variable) cannot be applied to the execution hosts, but the **cpus_per_core** and **distribution_policy** arguments can be modified. Therefore, when calling the **esub.p8aff** script from the **bmod** command, you must ensure that the specified **SMT** argument is the same as the **SMT** argument in the original job submission. Otherwise, the generated affinity string might not match the effective SMT mode on execution hosts, which might produce unpredictable affinity results.

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 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.

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.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

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

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

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

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

Part Number: CNC26EN

Printed in USA

(1P) P/N: CNC26EN