IBM Spectrum LSF

What's New in LSF Service Pack 13

July 2022

John Welch
jswelch@us.ibm.com
Technical Specialist
Data & AI

Qi Wang
Senior Software Archtect
Spectrum Computing

Yu-Ching Chen
Advisory Software Developer
Spectrum Computing

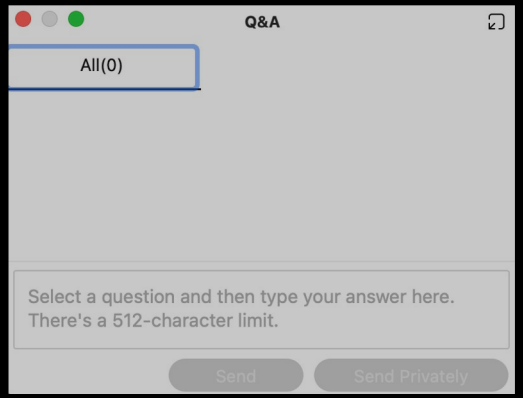
Bohai Zhang
Support Manager
Spectrum Computing

Yi Sun
Technical Account Manager
Spectrum Computing

Bill.McMillan@uk.ibm.com
Principal Product Manager
Spectrum Computing

Agenda

- Continuous Delivery of New Capabilities
- New Enhancements prior to Service Pack 13
- Service Pack 13 Enhancements
- Q&A: Ask questions at any time in the Q&A panel



Release Strategy: Revolution vs Evolution

Major Release (10.1)

Contains:

- Significant Architectural Changes
- Possible incompatibilities.
- Relinking/compiling against API's

Customer Impact

- Significant project
- Slow to upgrade, slow to apply patches
- Dependencies on ISV's to certify
- Delayed value

Service Pack (10.1.0.x)

Contains:

- Significant new functionality
- Cumulative fix roll up
- No incompatibilities, no relinking

Customer Impact

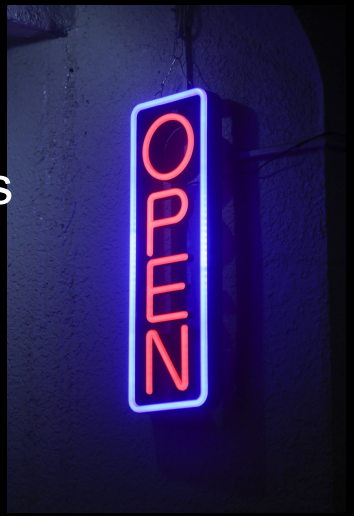
- Applied as a rolling update (e.g. Windows)
- Quick to update
- No dependencies on ISV's to certify
- Accelerated value

Continuous Delivery of New Capabilities

New Enhancements prior to Service Pack 13

- Support EC2 Fleet API in LSF Resource Connector for AWS

  Available via LSF patch
- Cloud provider plug-ins for the LSF Resource Connector
- Operator for LSF integration on OpenShift and Kubernetes



Service Pack 13

- Job Scheduling and Execution
- Resource Connector
- Resource Management
- Container Support
- Command Output Formatting
- Miscellaneous Changes

Job Scheduling and Execution

- Kill jobs by status
- Kill jobs and record jobs as DONE
- Job count based fairshare scheduling
- Delete “job groups” using idle times
- Modify cgroup memory and swap limits for running jobs

*New bkill options

- Kill by job status with

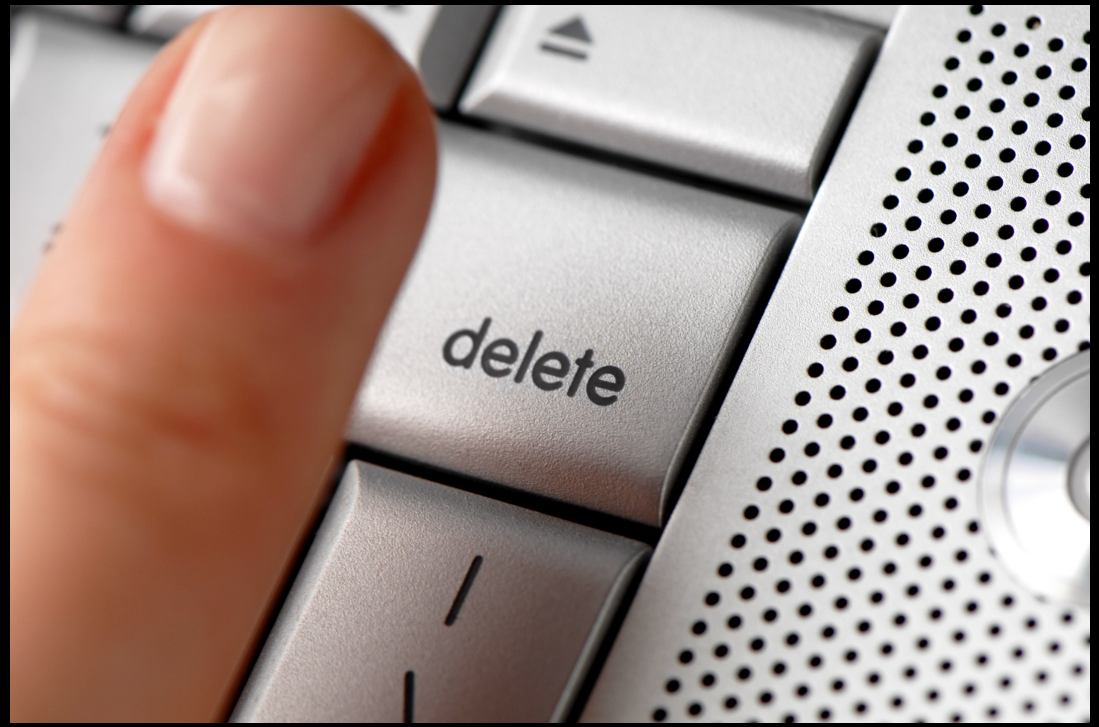
  -stat run|pend|susp

  run: kill jobs in RUN, WAIT, UNKWN
  pend: kill jobs in PEND, PSUSP
  susp: kill jobs in SSUSP, USUSP

- Kill jobs and record as DONE

  -d

  Only applies to jobs in RUN, USUSP or SSUSP states



*Job count based fairshare scheduling

- New FAIRSHARE_JOB_COUNT parameter in lsb.params
  - Values Y,y,N,n
  - Defaults to N

User Interface changes

- bgpinfo -l and bqueues -l display new columns: STARTED_JOBS and RESERVED_JOBS
- bgpinfo -l will always display these new columns.
- bqueues -l will only display these columns when FAIRSHARE_JOB_COUNT = Y

```
[root@ib21b01 ~]# bqueues -lr normal
QUEUE: normal
  -- For normal low priority jobs, running only if hosts are lightly loaded.  This is the default queue.
PARAMETERS/STATISTICS
PRIO NICE STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV PJOBS
 30    0  Open:Active       -    -    -    -    16     0    16     0     0    0     0
Interval for a host to accept two jobs is 0 seconds

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

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

SHARE_INFO_FOR: normal/
 USER/GROUP   SHARES  PRIORITY  STARTED  RESERVED  CPU_TIME  RUN_TIME   ADJUST  GPU_RUN_TIME  STARTED_JOBS  RESERVED_JOBS
root            1       0.500     16        0         0.0      352      0.000             0             1              0

USERS: all
HOSTS:  all
REQUEUE_EXIT_VALUES:  2
```

*Delete “job groups” using idle times

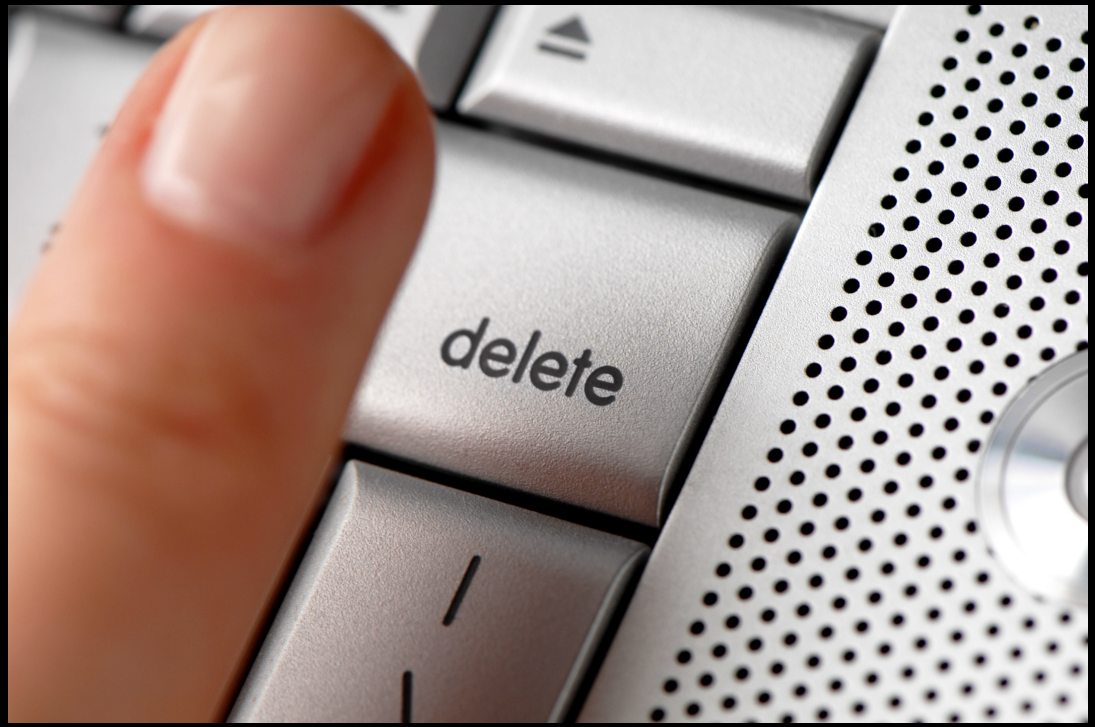
1. New JOB_GROUP_IDLE_TTL parameter in lsb.params

    Idle time-to-live (TTL) is in seconds

2. bgdel has new option “-d” to specify idle time.

    Idle time is in seconds

3. bjgroup has new IDLE_TIME column in output



*cgroup memory and swap limits modifiable when job is running

- Requirements (same as before)
  - Enable LSF cgroup integration in lsf.conf
  - RHEL 6.2 and above or SLES 11 SP and above
  - cgroup subsystem has been enabled on the Linux hosts of the cluster.

LSF cgroup v1 vs v2

-M mem limit	-v swap limit	cgroup v1 & v2 mem limit	cgroup v1 mem+swap limit	cgroup v2 swap limit
300	Unspecified or unlimited	300	300	0
Unspecified or unlimited	300	300	300	0
300	300	300	600	300

cgroup behaviour when changing limits.

cgroup v1 memory limit reduced example

Job submission with memory and swap limits

```
$ bsub -m qwang-front -M 100 -v 50 ./memoryeater 80

Job <633> is submitted to default queue <normal>.


Mon Jan 24 12:24:36: Resource usage collected.
          MEM: 82 Mbytes;  SWAP: 0 Mbytes;  …
          PGID: 9220;  PIDs: 9220 9221 9223
```

Modify memory limit example 1

```
$ bmod -M 50 633
Parameters of job <633> are being changed

Mon Jan 24 12:25:44: Resource usage collected.
          MEM: 50 Mbytes;  SWAP: 32 Mbytes;  …
          PGID: 9220;  PIDs: 9220 9221 9223
```

Modify memory limit example 2

```
$ bmod -M 20 633
Parameters of job <633> are being changed

…

EXTERNAL MESSAGES:
 MSG_ID FROM       POST_TIME      MESSAGE      ATTACHMENT
 0      root       Jan 24 12:27   Could not modify the job's cgroup m
```

cgroup v2 memory limit reduction - same example 2

```
MEMORY USAGE:
 MAX MEM: 82 Mbytes;  AVG MEM: 53 Mbytes

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

 EXTERNAL MESSAGES:
 MSG_ID FROM       POST_TIME      MESSAGE                              ATTACHMENT
 0      root       Jan 24 12:27   Could not modify the job's cgroup m     N
```

– **cgroup v2**: the job is killed

swap limit reduction rules

- **cgroup v1**: the cgroup subsystem won't modify the mem+swp limit if the new value is less than the current usage.
- **cgroup v2**: accepts the new limit, and leaves the job alone

```
$ bsub -m modesty1 -M 50 -v 50 ./memoryeater 80
Job <631> is submitted to default queue <normal>.
```

```
$ bmod -v 10 631
Parameters of job <631> are being changed
```

Resource Connector Enhancements

- Automatic Selection of Spot template

- Selecting templates for minimum number of servers

*Automatic selection of spot templates

- Configuration information
- Limitations

Configuration information

- Set the allocationStrategy parameter to lowestPrice in awsprov_templates.json.
- Recommended to set spot instance as higher priority template than on demand templates
- Existing configuration for spot instance will work automatically with the new feature

Limitations

- Spot price checks are not guaranteed to be real time. There may be a short delay between seeing a price drop from a primary source before LSF is able to see the change
- Templates are only re-enabled after price goes below the set price, but only after the next demand cycle will jobs be able to be provisioned onto the template

*Template Optimization

- Introduction
- Configuration example
- Algorithm
- Limitations

Introduction

- This feature will allow configuring templates to optimize to other templates
- The admin will configure optimization rules that determine how many VMs of one template is worth another template
- LSF will go through regular demand scheduling and determine how many VMs will be needed on each template
- LSF then will go through each optimization configuration to determine which machines can be optimized and moved to the new VM Templates

Configuration

- New optimization allocRules section in policy_config.json

- The fromTemplate factor determines how many VMs will be worth optimizing toTemplate factor VMs

```
"Optimizations" : {
   "allocRules" : [
      {
         "fromTemplate": {
            "provider" : "aws",
            "templateName" : "templateA",
            "factor" : 4
         },
         "toTemplate" : {
            "provider" : "aws",
            "templateName": "templateC",
            "factor" : 1
         }
      },
      {
         "fromTemplate": {
            "provider" : "aws",
            "templateName" : "templateA",
            "factor" : 2
         },
         "toTemplate" : {
            "provider" : "aws",
            "templateName": "templateB",
            "factor" : 1
         }
      }
   ]
  }
}
```

badmin output

```
$ badmin rc view -c policies
…
Optimizations
        4 hosts (aws:templateA) replaced by 1 hosts (aws:templateC)
        2 hosts (aws:templateA) replaced by 1 hosts (aws:templateB)
```

badmin rc view –c policies show the optimizations from => to

provider:templateID:Factor => provider:templateID:Factor

Limitations

- Enhancement does not guarantee optimal job placement.
- Since LSF does not force jobs to the optimize template nor the related provisioned real host, this may cause some jobs to require a re-provision

Resource Management

- Global Resources
- bwait enhacement
- GPU resource allocation for resizable jobs
- Default values for GPU parameter are changed

*Global Resources

1. Configuration

   lsb.globalpolicies

2. Commands and Usage

Configuration

1) Enable global policy

- Add following parameter in lsf.conf for **all** clusters use same values.

✓ LSB_GPD_PORT=<port>

✓ LSB_GPD_CLUSTER=<**cluster**>

configure the cluster as the submission cluster which startup gpolicyd service

Configuration (Cont.)

2) Example of global resources in lsb.globalpolicies of GPD cluster

```
Begin Resource
RESOURCENAME         TYPE      INTERVAL  INCREASING  CONSUMABLE RELEASE DESCRIPTION
global_res_static    Numeric   ()        N           Y          Y       (global static res)
gres2                Numeric   ()        N           Y          Y       (global static res 2)
global_res_dynamic   Numeric   60        N           Y          Y       (global dynamic res)
End Resource

Begin ResourceMap
RESOURCENAME         LOCATION
global_res_static    (100@[all])
gres2                (10@[all])
global_res_dynamic   ([all])
End ResourceMap
```

Currently only support **NUM@[all] for static resource and [all] for dynamic resource.**

Configuration Dynamic Global Resource

Create gres for dynamic global resource and put it to $LSF_SERVERDIR of GPD cluster.

```
$ cat gres.test
#!/bin/bash
value=500
while true
do
    echo "1 global_res_dynamic ${value}"
    sleep 60
done
```

Global Distribute Policy

The global resource will be distributed among all the connected clusters. There are two kind of policies to control the distribution.

- a) ***evenly*** distribution policy

  The available global resource will be divided evenly among all the connected clusters. It's dynamically.

  Example:

  Suppose there is a global resource and its initial value 100 shared among 4 clusters. Each cluster can get 25 available resource first. If cluster1 run used 20 resource, then the total available is 80 now and each cluster can get 20 available resource at this moment.

- b) ***compete*** distribution policy

  Each local cluster will compete to use the global resource. This is the ***default*** behavior.

Configure Global Distribute Policy

- Example of global policies in lsb.globalpolicies file in GPD cluster.

For global resources:

```
Begin DistributePolicy
NAME=Resource_Distribute_Policy
DISTRIBUTE=resource
POLICY=compete
End DistributePolicy
```

For global limits:

```
Begin DistributePolicy
NAME=Limit_Distribute_Policy
DISTRIBUTE=limit
POLICY=evenly
End DistributePolicy
```

Query Global Policy Information

```
$ bgpinfo policy

Global Policy:

Global distribute policy for global resources: compete

Global distribute policy for global limits: evenly
```

Configure Reservation for Global Resource

Example of global resource reservation in lsb.globalpolicies file in GPD cluster.

```
Begin ReservationUsage
RESOURCE        METHOD        RESERVE
gres1           PER_TASK      Y
gres2           PER_HOST      Y
End ReservationUsage
```

Query Global Resource Configuration

```
$ bgpinfo resource -c
RESOURCE_NAME        TYPE    ORDER  INTERVAL  RELEASE CONSUMABLE    METHOD RESERVE
global_res_dynamic   Numeric   Dec        60      Yes        Yes         -      No
global_res_static    Numeric   Dec         0      Yes        Yes         -      No
compete_res          Numeric   Dec         0      Yes        Yes  PER_HOST     Yes
non_consume          Numeric   Dec         0      Yes         No         -      No
```

Using global resource

(same way as local resource)

bsub -R "rusage[global_res_static=10]" sleep 1000

bsub -R "rusage[global_res_static=10]" sleep 1000

bsub -n2 -R "rusage[global_res_static=10/task]" sleep 1000

Check global resource information

(same as local share resource)

```
$ bgpinfo resource
RESOURCE                 TOTAL        RESERVED
policy_change_res          8.0             0.0
global_res_dynamic           -             0.0
global_res_static        200.0             0.0
```

- To check in each execution cluster
  - **bhosts -s**
- To check specific global resource
  - **bgpinfo resource -s <resourceName>**
- To check wide format
  - **-w**

Check global resource information

To check detail resource usage and avail information for each cluster

```
$ bgpinfo resource -l
RESOURCE            CLUSTER              TOTAL      RESERVED
global_res_static   <ALL>                170.0          30.0
                    mcpull-ib22b08-e      44.0           0.0
                    exec_ib15b02          42.0          10.0
                    exec_ib15b03          42.0          20.0
                    mcpull-ib15b01-s      42.0           0.0
gres2               <ALL>                  4.0           6.0
                    mcpull-ib22b08-e       1.0           0.0
                    exec_ib15b02           1.0           2.0
                    exec_ib15b03           1.0           4.0
                    mcpull-ib15b01-s       1.0           0.0
```

*bwait enhancement

- New LSB_BWAIT_IN_JOBS=N in lsf.conf

  Default is Y

- External message posted by bwait to job

```
EXTERNAL MESSAGES:
 MSG_ID FROM       POST_TIME      MESSAGE          ATTACHMENT
 136    _system_   Nov 10 02:34   started(8109)           N
```

- bjobs -l or bhist –l show external message

*Dynamic GPU allocation of resizable jobs

1. In-scope
   - Resizable jobs consider GPU resource requirements
   - Shrink all tasks on a host
   - GPU enforcement of cgroup v1 and v2
   - LSB_GPU_NEW_SYNTAX=extend only
2. Out of scope
   - Jobs with mps, aff, mig enabled in GPU requirements
   - NVIDIA Data Center GPU Manager
   - Docker jobs
   - Shrink first execution host

GPU allocation change when resize

- Grown action
  - When a job grows slots, its GPUs usage changes proportionately:
    - Host-based GPUs usage increases only when the job gains tasks on a **new** host.
    - Task-based GPUs usage increases whenever the job grows
- Shrink action
  - Only support host based shrinking
  - Job releases all tasks on a host or more than one hosts.
  - The first execution host is not allowed to be released for resizable GPU job.

New in the User Interface for GPU resize

Enhancement to commands:

- bjobs/bacct/bhist –gpu –l : Show the new allocated GPUs together with previous allocated GPUs in “GPU_ALLOCATION” section
- bhosts –gpu: GPUs’ job counters will be updated
- bhist –l: new allocated GPUs string will be displayed in resized action event line.

New environment variables (for resize notification command scripts):

- LSB_RESIZE_GPUS:
  Lists the additional GPUs for a grow event or the released GPUs for a shrink event.

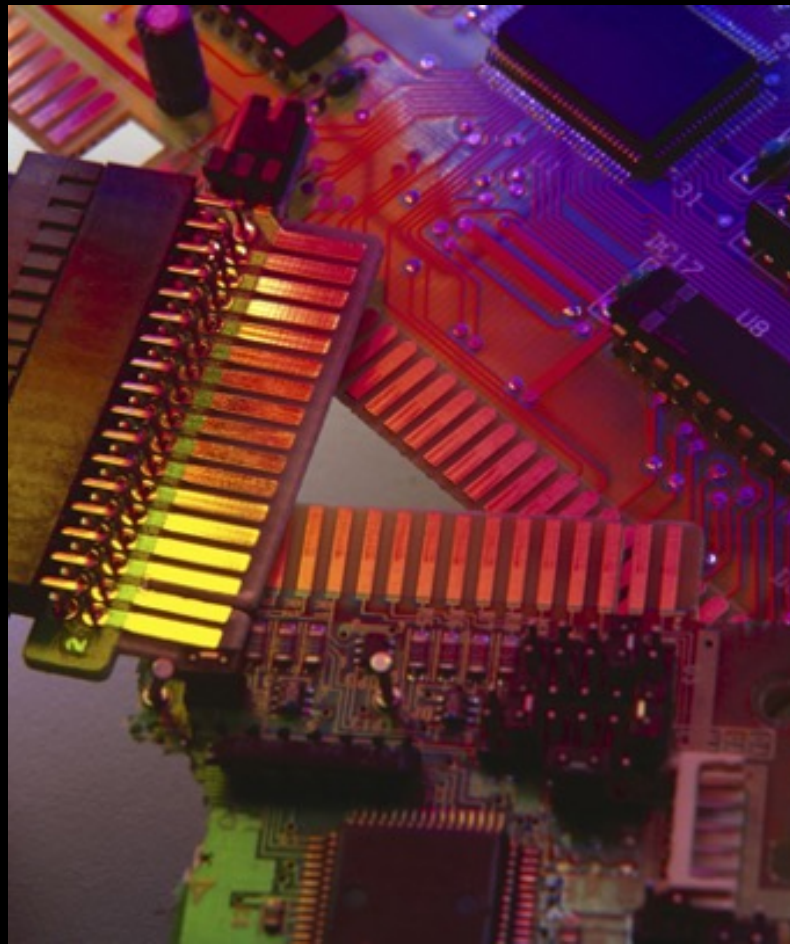
- LSB_ RESIZE_TIME:
  Timestamp for the resize action, which helps identify the exact resized GPU allocation for changed tasks. Export this environment variable for the blaunch command before new tasks grow or shrink. If not set, when the blaunch resizes tasks, LSF uses the latest resized GPU allocation.

*Default values for GPU parameter have changed to

LSF_GPU_AUTOCONFIG=Y

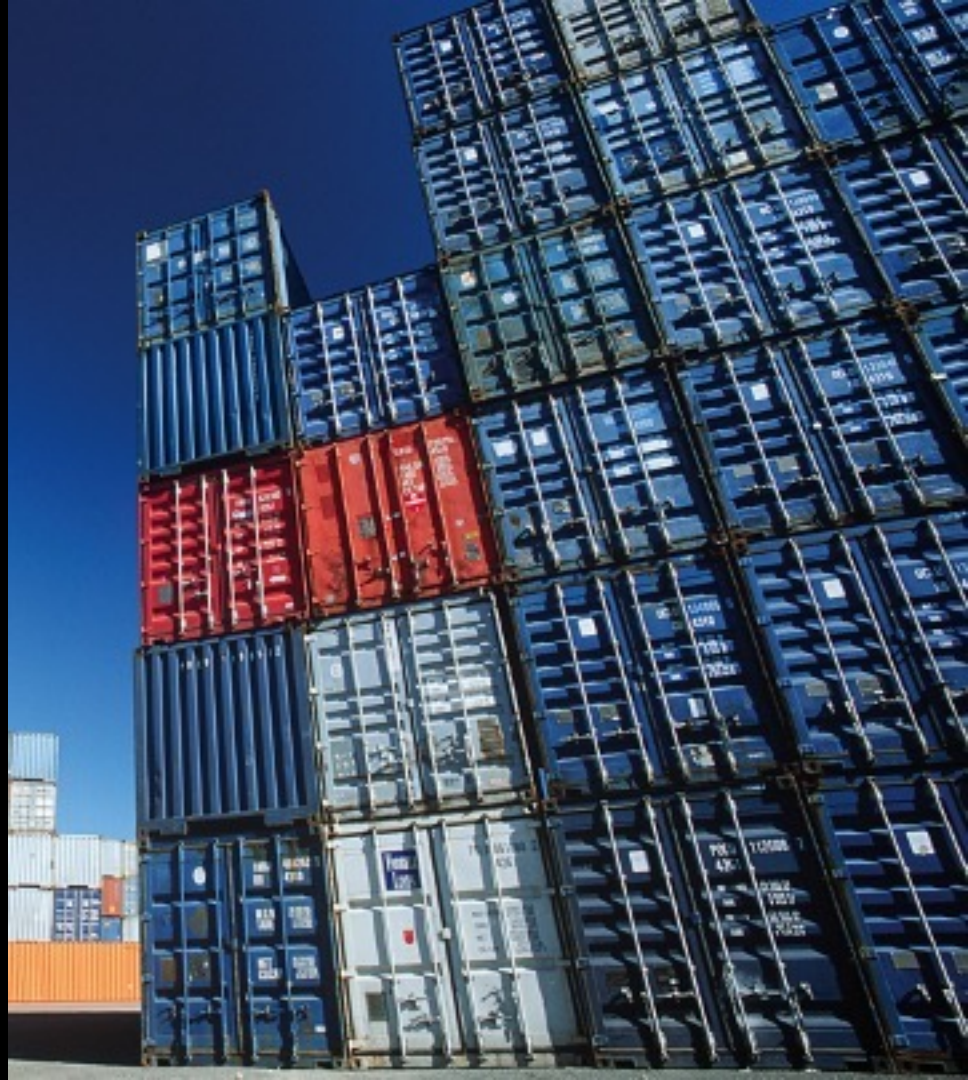
LSB_GPU_NEW_SYNTAX=extend

LSF_GPU_RESOURCE_IGNORE=Y



Container Enhancements

- Podman version 3.3.1 Support
- Apptainer for running LSF Jobs
- New parameter to mount or not mount TMP Directory



*Podman version 3.3.1 Support

- Configuration example
- Submission example
- Differences between old Podman support and new

Podman Configuration

- **Configuration**

```
CONTAINER=podman[image(image_name) options(podman_run_options)]
EXEC_DRIVER=context[user(default)]
            starter[/path/to/serverdir/docker-starter.py]
            controller[/path/to/serverdir/docker-control.py]
```

```
$ bapp  -l pd

APPLICATION NAME: pd
 -- podman instead of docker for controller

STATISTICS:
   NJOBS     PEND      RUN    SSUSP    USUSP      RSV
       3        0        3        0        0        0

PARAMETERS:

CONTAINER: podman[image(ubuntu) options(--rm)]
EXEC_DRIVER:
    context[user(default)]
    starter[/opt/ibm/10.1/linux3.10-glibc2.17-x86_64/etc/docker-starter.py]
    controller[/opt/ibm/10.1/linux3.10-glibc2.17-x86_64/etc/docker-control.py]
```

Podman Job submission example

- **Usage**

submit job to the application profile or the queue that has podman job configured

```
$ bsub –app pd hostname
Job <1> is submitted to default queue <normal>.
```

Podman Job

What is different from old Podman job?

1. The old configuration of podman is obsolete

Before this project, the configuration of podman in lsb.applications and lsb.queues is

```
CONTAINER=docker[image(image_name) options(podman_run_options)]
```

The reason is that podman is considered as placement of docker. Nowadays, docker becomes less important in container runtime and podman is independent for use. So, we use `podman` instead of `docker` in configuration.

```
CONTAINER=podman[image(image_name) options(podman_run_options)]
```

2. The `docker` boolean resource dependency is not mandatory for `podman` job anymore.

`docker` Boolean resource should be configured before container configuration. For podman, we do not have this configuration dependency anymore.

3. More strict checking

```
EXEC_DRIVER=context[user(default)]
            starter[/path/to/serverdir/docker-starter.py]
            controller[/path/to/serverdir/docker-control.py]
```

`default` user is mandatory for podman job. If it is not configured, `default` is the default value. `starter` and `controller` are mandatory for podman to work.

*Singularity and Apptainer Job Support

Apptainer Job example

- **Configuration**

keyword `apptainer` is introduced in CONTAINER configuration for lsb.applications and lsb.queues.

```
[lsfadmin@dlw14 conf]$ bapp  -l apptainer

APPLICATION NAME: apptainer
 -- apptainer

STATISTICS:
   NJOBS     PEND      RUN    SSUSP    USUSP      RSV
       0        0        0        0        0        0

PARAMETERS:

CONTAINER: apptainer[image(docker://docker.io/ppc64le/centos)]
```

- **Usage**

an Apptainer job should be submitted to an application profile or a queue that configured with `apptainer` container

```
$ bsub -app apptainer hostname
Job <1> is submitted to default queue <normal>.
```

GPU Job for Apptainer examples

- **Configuration**

`LSB_RESOURCE_ENFORCE=gpu` is used to isolate GPUs by cgroups for Singularity/Apptainer job container. It should be enabled for GPU isolation.

- **Usage**

an Apptainer job should be submitted with GPU specifications.

```
$ bsub -I -gpu num=2 -app apptainer nvidia-smi -L
Job <724> is submitted to default queue <interactive>.
<<Waiting for dispatch ...>>
<<Starting on dlw14.aus.stglabs.ibm.com>>
INFO:    Using cached SIF image
GPU 0: Tesla V100-SXM2-16GB (UUID: GPU-bbbe483a-6e5f-721f-c271-fd175f0d8656)
GPU 1: Tesla V100-SXM2-16GB (UUID: GPU-f1c54787-2f6c-c2bd-769d-e7ecb0324207)

$ bsub -I -gpu num=1 -app apptainer nvidia-smi -L
Job <725> is submitted to default queue <interactive>.
<<Waiting for dispatch ...>>
<<Starting on dlw14.aus.stglabs.ibm.com>>
INFO:    Using cached SIF image
GPU 0: Tesla V100-SXM2-16GB (UUID: GPU-bbbe483a-6e5f-721f-c271-fd175f0d8656)
```

*Mount /tmp directory in container jobs

New LSF_DOCKER_MOUNT_TMPDIR in lsf.conf

When this parameter is set to Y or y, LSF mounts the temporary (/tmp) directory to the temporary directory of the host (/tmp) in the container of the Docker job.

Default value is Y

Command Output Formatting

- New output fields for bqueues -o
- New –o parameter for blimits
- CPU Peak Efficiency added

*New output fields added to bqueues -o option

Column Name	Width	Alias
MAX_CORELIMIT	8	CORELIMIT
MAX_CPULIMIT	30	CPULIMIT
DEFAULT_CPULIMIT	30	DEF_CPULIMIT
MAX_DATALIMIT	8	DATALIMIT
DEFAULT_DATALIMIT	8	DEF_DATALIMIT
MAX_FILELIMIT	8	FILELIMIT
MAX_MEMLIMIT	8	MEMLIMIT
DEFAULT_MEMLIMIT	8	DEF_MEMLIMIT
MAX_PROCESSLIMIT	8	PROCESSLIMIT
DEFAULT_PROCESSLIMIT	8	DEF_PROCESSLIMIT
MAX_RUNLIMIT	12	RUNLIMIT
DEFAULT_RUNLIMIT	12	DEF_RUNLIMIT
MAX_STACKLIMIT	8	STACKLIMIT
MAX_SWAPLIMIT	8	SWAPLIMIT
MAX_TASKLIMIT	6	TASKLIMIT
MIN_TASKLIMIT	6	-
DEFAULT_TASKLIMIT	6	DEF_TASKLIMIT
MAX_THREADLIMIT	6	THREADLIMIT
DEFAULT_THREADLIMIT	6	DEF_THREADLIMIT
RES_REQ	20	-
HOSTS	50	-

Output example

$ bqueues -o "queue_name runlimit”

```
QUEUE_NAME RUNLIMIT
admin -
owners -
priority -
night -
short -
normal 300.0
mininteractive -
idle -
```

*New -o option for blimits

Field Names and Field Widths supported

Name	Width
NAME	12
CLUSTER	12
USERS	16
QUEUES	16
HOSTS	16
PROJECTS	16
LIC_PROJECTS	20
APPS	8
SLOTS	8
MEM	8
TMP	8
SWP	8
JOBS	8

User defined resources will have a column width of 10.

Example Formatted Output

$ blimits -a -o "Name:10 queues hosts: delimiter=',"

```
INTERNAL RESOURCE LIMITS:

NAME      ,QUEUES,HOSTS
limit1    ,normal,tyandevsvl1
NONAME000 ,normal,-
NONAME001 ,short,-

EXTERNAL RESOURCE LIMITS:

NAME      ,QUEUES,HOSTS
limit1    ,normal,tyandevsvl1
NONAME001 ,short,-
```

*bjobs report CPU peak efficiency

- Background
- Scope of enhancement
- Configuration
- Details of functions
- Feature interactions

Background

Users want to know the peak CPU number a finished job actually used, and the CPU efficiency based on the CPU number the job requested, and also the memory efficiency of the finished jobs.

Actual Peak number of CPUs job used

?

Number of CPU requested in bsub

Scope of enhancement

1. bjobs report peak number of CPUs used
2. bjobs report CPU usage efficiency
3. bjobs report memory efficiency
4. bhist/bacct show CPU/mem efficiency and peak usage for finished jobs after job finished

Configuration

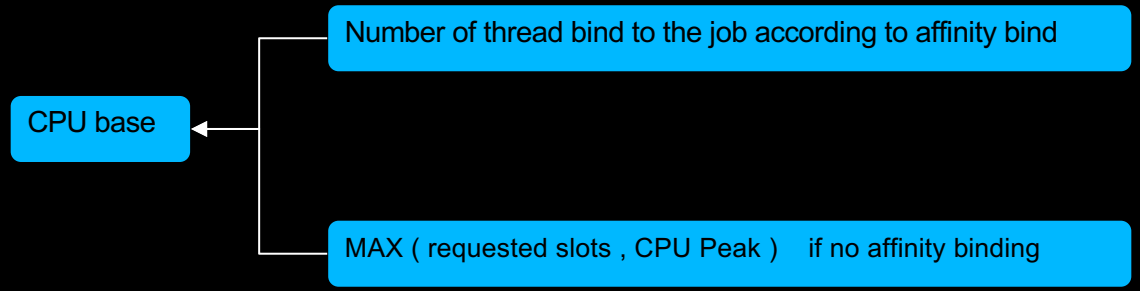
Define new CPU_PEAK_SAMPLE_DURATION parameter in lsb.params to control how offen the CPU/memory efficiency calculation will be triggered.

The default value is 60, which means the calculation is triggered every 60 seconds.

If define the value to 0, the caculation will be triggered only while job finish/suspend/resumed or modified.

Details of function – CPU Efficiency

CPU Efficiency = (CPU Peak / CPU base) * 100%



Details of function – Memory Efficiency

Use current MAX MEM collected in sbd for memory efficiency calculating.

Memery efficiency = (maxMem / rusage mem) * 100%

If the job has not request memory in rusage[] section, memory efficiency = 0%, as we use this memory efficiency to optimize the value of memory we requested for memory.

For parallel job we would use the summary of maxMem and rusage on all execution hosts for calculating the efficiency.

If specify the rusage memory in per_task or per_host in any level, the summary of rusage memory calculation will follow the specification.

Details of function

The CPU and memory efficiency calculation will be triggered every CPU_PEAK_SAMPLE_DURATION reached and the cpuPeak or maxMem is bigger than last duration.

New struct introduced in lsbatch.h to record new info for query jobs:

```
struct jobCpuMemAcct{
    float cpuPeak; /**< Job CPU usage peak */
    float cpuEfficiency; /**< Job CPU usage efficiency */
    float memEfficiency; /**< Job memory usage efficiency */
}
```

Feature interaction

Efficiency calculation is triggered once after
1) a successful bmod based on current effective resreq
2) a successful bresize action

The effective resreq rusage will not be changed by bswitch so no efficiency calculation after bswitch.

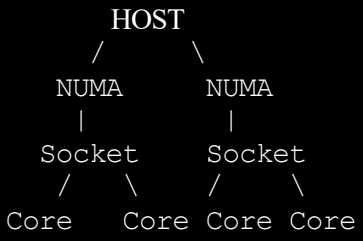
Miscellaneous Enhancements

- Updated support for hardware locality library
- Limit jobs and tasks in Multiclsuter receive queues
- Honoring the preferred host for host group members
- Host group support for commands that support host names
- New platform support

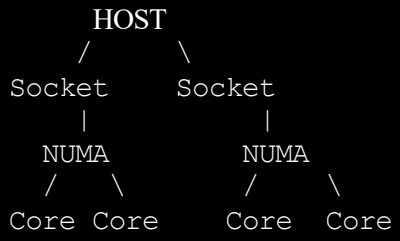
*Hardware locality (hwloc) 2.6 support

Change hierarchy for hosts with one NUMA per socket:

Previously in hwloc v1:



Now in hwloc v2:



Update to lim –T/lshosts –T to include physical index for socket & core

```
$ lshosts -T bjhc01
Host[125.7G] bjhc01
    NUMA[0: 62.7G]
        Socket
            core(0 16)
            core(1 17)
            core(2 18)
            core(3 19)
            core(4 20)
            core(5 21)
            core(6 22)
            core(7 23)
    NUMA[1: 62.9G]
        Socket
            core(8 24)
            core(9 25)
            core(10 26)
            core(11 27)
            core(12 28)
            core(13 29)
            core(14 30)
            core(15 31)
```

Change to

```
$lshosts -T bjhc01
Host[125.7G] bjhc01
    Socket0
        NUMA[0: 62.7G]
            core0(0 16)
            core1(1 17)
            core2(2 18)
            core3(3 19)
            core4(4 20)
            core5(5 21)
            core6(6 22)
            core7(7 23)
    Socket1
        NUMA[1: 62.9G]
            core0(8 24)
            core1(9 25)
            core2(10 26)
            core3(11 27)
            core4(12 28)
            core5(13 29)
            core6(14 30)
            core7(15 31)
```

*Queue level remote jobs/tasks running limit

- New parameter in lsb.queues for RCVJOBS_FROM queues

  IMPT_JOBLIMIT – defaults to unlimited

  IMPT_TASKLIMIT – defaults to unlimited

- For example

```
Begin Queue

QUEUE_NAME = example

…

IMPT_JOBLIMIT = 100      # how many remote jobs can be started

IMPT_TASKLIMIT = 200     # how many tasks from remote jobs can be started

RCVJOBS_FROM = remote_cluster1 remote_cluster2

End Queue
```

*Host Group member preference

- For example, in lsb.hosts

```
hgroup1 (host1 host2)                # no preference
hgroup2 (host3+2 host4+1 host5)      # preference on hosts
hgroup3 (hgroup1+2 hgroup2+1 host6) # preference on subgroups
```

- Rules
  1. The preference defined in subgroup will be overwritten by its parent
  2. hostgroup preference take effects only if asked host is a single hostgroup

- Submission examples

```
$ bsub –m hgroup3 …
LSF converts to “host1+2 host2+2 host3+1 host4+1 host5+1 host6” for scheduling.
$ bsub –m “hgroup2 host1” …
LSF converts to “host3 host4 host5 host1” for scheduling.
```

Restrictions & Limitations

Admin can **not** configure preference at hostgroup in the following cases:

- An exclamation mark (!) indicates an externally defined host group (egroup).
- Use a tilde (~) to exclude specified hosts or host groups from the list.
- Dynamic group.
  - Use an asterisk (*) as a wildcard character.
  - Use square bracket with a hyphen (host[integer1-integer2]) or a colon (host[integer1:integer2]), or with commas (host[integer1, integer2, …]).
- MC Lease-in hosts, like ALLREMOTE

Admin can **not** dynamically define preference with “bconf” or “badmin hghostadd”

bmgroup

- Use bmgroup to verify preferences

```
% bmgroup -l hgroup3
GROUP_NAME  HOSTS
hgroup3             hgroup1/+2 hgroup2/+1 host6

% bmgroup -r hgroup3
GROUP_NAME  HOSTS
hgroup3             host1+2 host2/+2 hosts3+1 host4+1 host5+1 host6
```

*Enhanced hostgroup support

Configuration

- New LSF_HOSTGROUP_INFO = Y in lsf.conf. Default is N

Commands

- lshosts [host_name | **host_group**]
- lsload -m [host_name | **host_group**]
- battr [create | show | delete] –m [host_name | **host group**]
- brsvs [–p [all | host_name | host_group]] | [–z [all | host_name | **host_group**]]
- bresume –m [host_name | **host_group**]

*New platform support

- RHEL 8.5 and 8.6 on x64 and Power, kernel 4.18.0, glibc 2.28
- RHEL 9.0 on x64 and Power, kernel 5.14.0, glibc 2.34
- RHEL 8.*x*, RHEL 9.0, and IBM AIX 7.*x* on IBM Power 10

- Full platform support

References

Release Notes of FP13:

https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=wn-whats-new-in-lsf-101-fix-pack-13

Download link of FP13 from IBM Fix Central:

https://www.ibm.com/support/fixcentral/swg/selectFixes?product=ibm/Other+software/IBM+Spectrum+LSF&release=All&platform=All&function=fixId&fixids=lsf-10.1.0.13-spk-2022-Apr-build601088&includeSupersedes=0

Thank you

John Welch
Technical Specialist
—
jswelch@us.ibm.com