LSF Knowledge Sharing

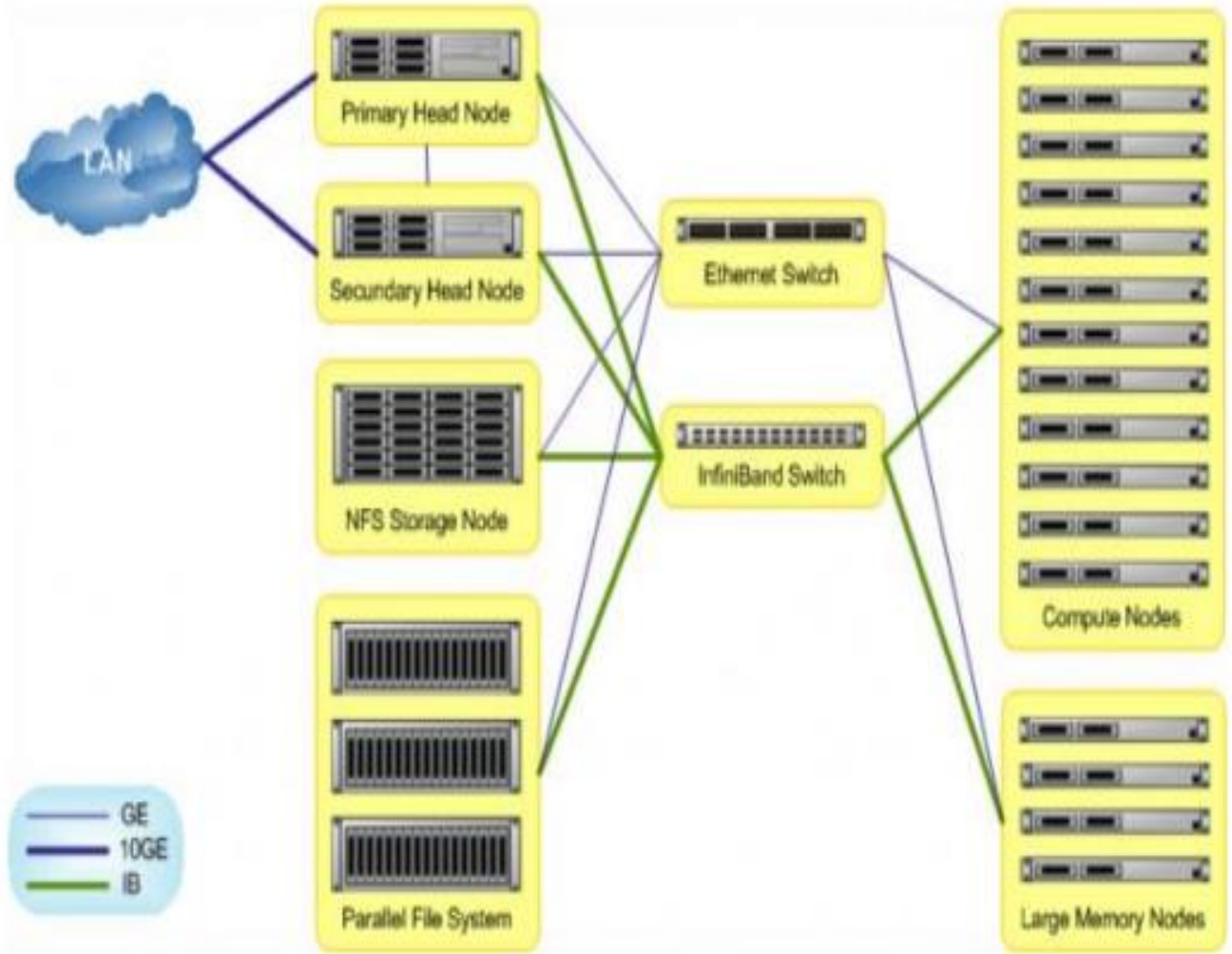
Agenda

- LSF
- Cluster
- Platform LSF
- LSF Basic Structure
- LSF Daemons (Processes)
- LSF Jobs
- LSF Job submission
- LSF User Commands
- LSF Queues
- LSF Hosts
- LSF Resources
- LSF Users
- LSF Scheduling
- LSF Exit Values

Load Share Facility (LSF)

- LSF was based on the Utopia research project at the University of Toronto. In 2007, Platform release Platform LAVA , which is a simplified version of LSF based on an old version of LSF release, licensed under GNU (General Public License). The project was discontinued in 2011, Succeeded by Open LAVA. In January, 2012, Platform Computing was acquired by IBM. The product is called now IBM Spectrum LSF .

Clusters



Clusters

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

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

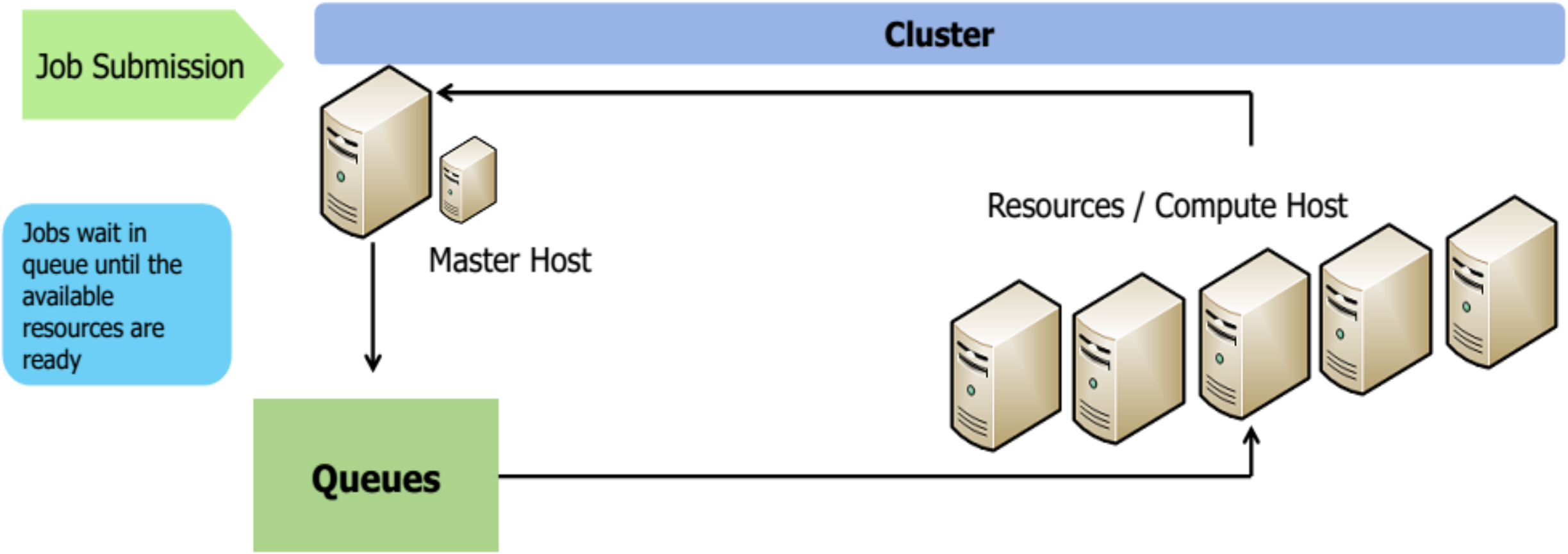
- All the hosts in a single administrative group
- All the hosts on one file server or sub-network
- Hosts that perform similar functions

HPC Cluster

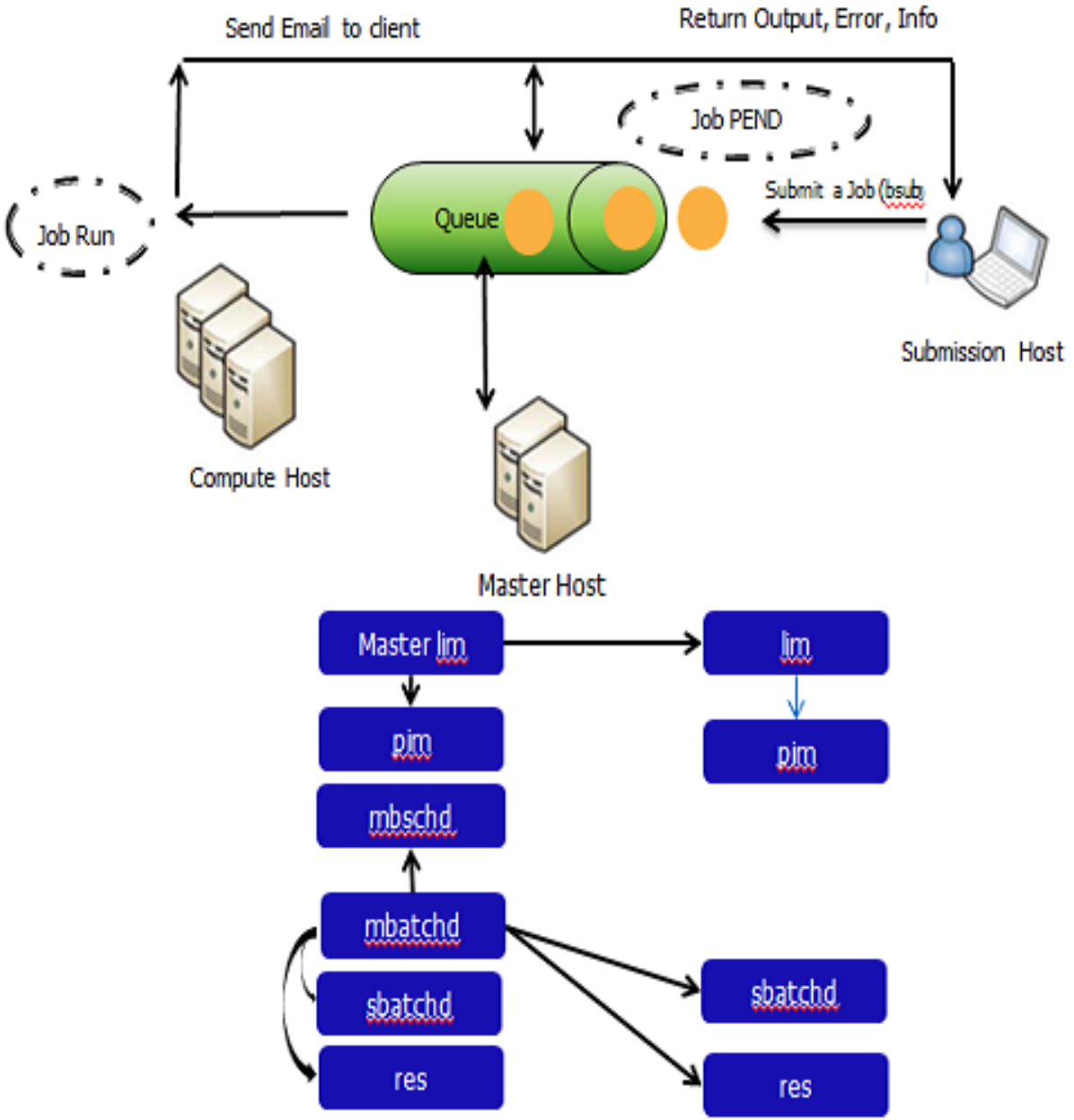
High-performance computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly.

Platform LSF

- IBM Platform LSF software is a industry leading enterprise – class software that distributes work across existing heterogeneous IT resources creating a shared, scalable, and fault-tolerant infrastructure, delivering faster, more reliable workload performance while reducing cost. LSF balances load and allocates resources, while providing access to those resources.
- LSF provides a resource management framework that takes your job requirements, finds the best resources to run the job, and monitors its progress. Jobs always run according to host load and site policies.



LSF Basic Structure



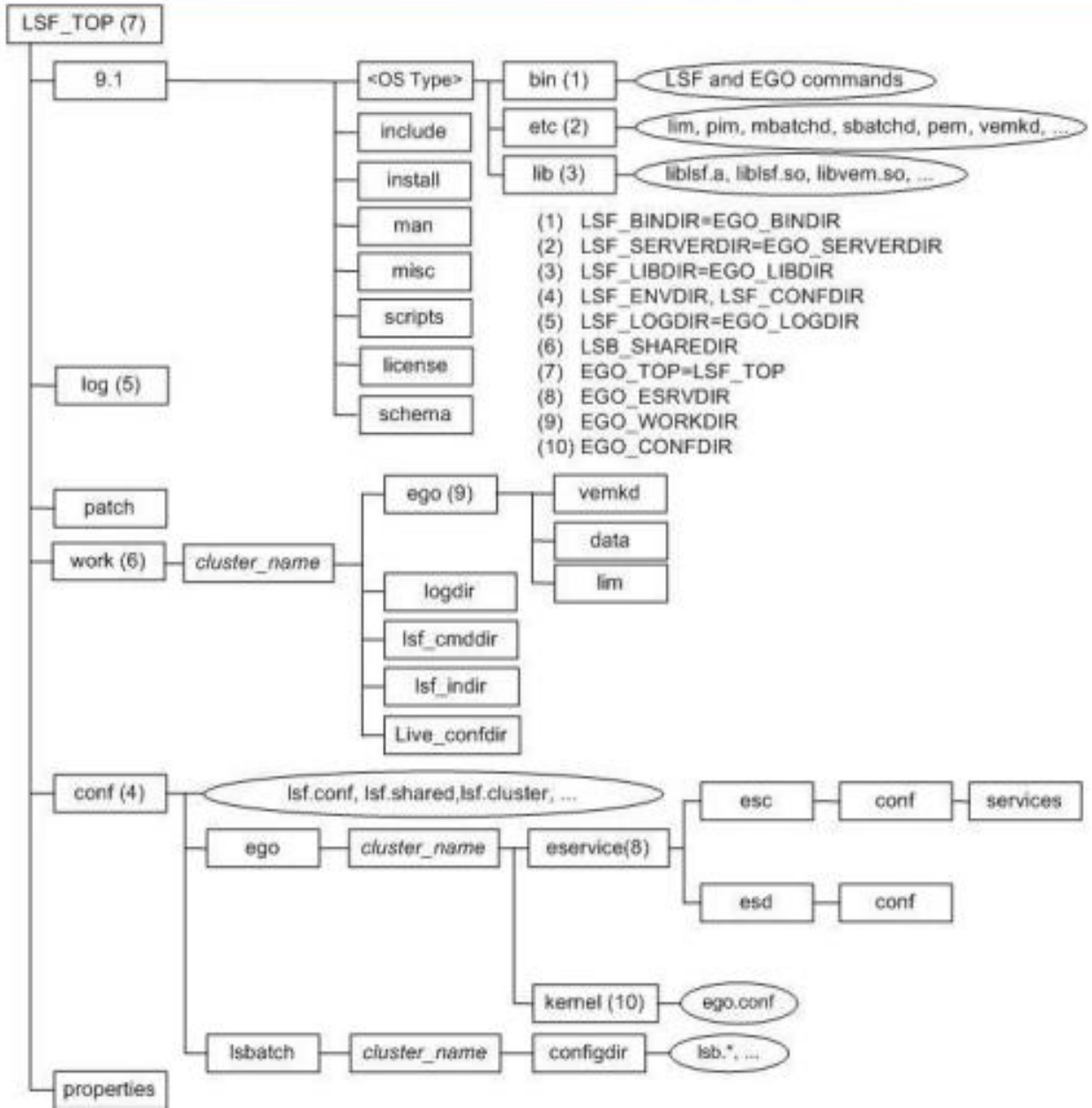
- An LSF Cluster can be divided Into two groups of hosts: Management Hosts And a Compute Hosts . Management Hosts Provide specialized services to the cluster; Compute hosts run the user workload.

LSF Daemons

- Mbatchd
- Mbschd
- Sbatchd
- Res
- Lim
- Master Lim
- ELIM
- Pim

LSF directories

- This diagram illustrates an example directory structure after the LSF installation is complete



LSF Daemons

- Mbatchd
  - Master Batch Daemon running on the master host.
  - Started by **sbatchd**.
  - Responsible for the overall state of jobs in the system.
  - Receives job submission, and information query requests.
  - Manages jobs that are held in queues. Dispatches jobs to hosts as determined by **mbschd**.
- Mbschd
  - Master Batch Scheduler Daemon running on the master host.
  - Works with **mbatchd**.
  - Started by **mbatchd**.
  - Makes scheduling decisions based on job requirements, and policies, and resource availability. Sends scheduling decisions to **mbatchd**.
- Sbatchd
  - Slave Batch Daemon running on each server host.
  - Receives the request to run the job from **mbatchd** and manages local execution of the job.
  - Responsible for enforcing local policies and maintaining the state of jobs on the host.
  - The **sbatchd** forks a child **sbatchd** for every job. The child **sbatchd** runs an instance of **res** to create the execution environment in which the job runs. The child **sbatchd** exits when the job is complete.

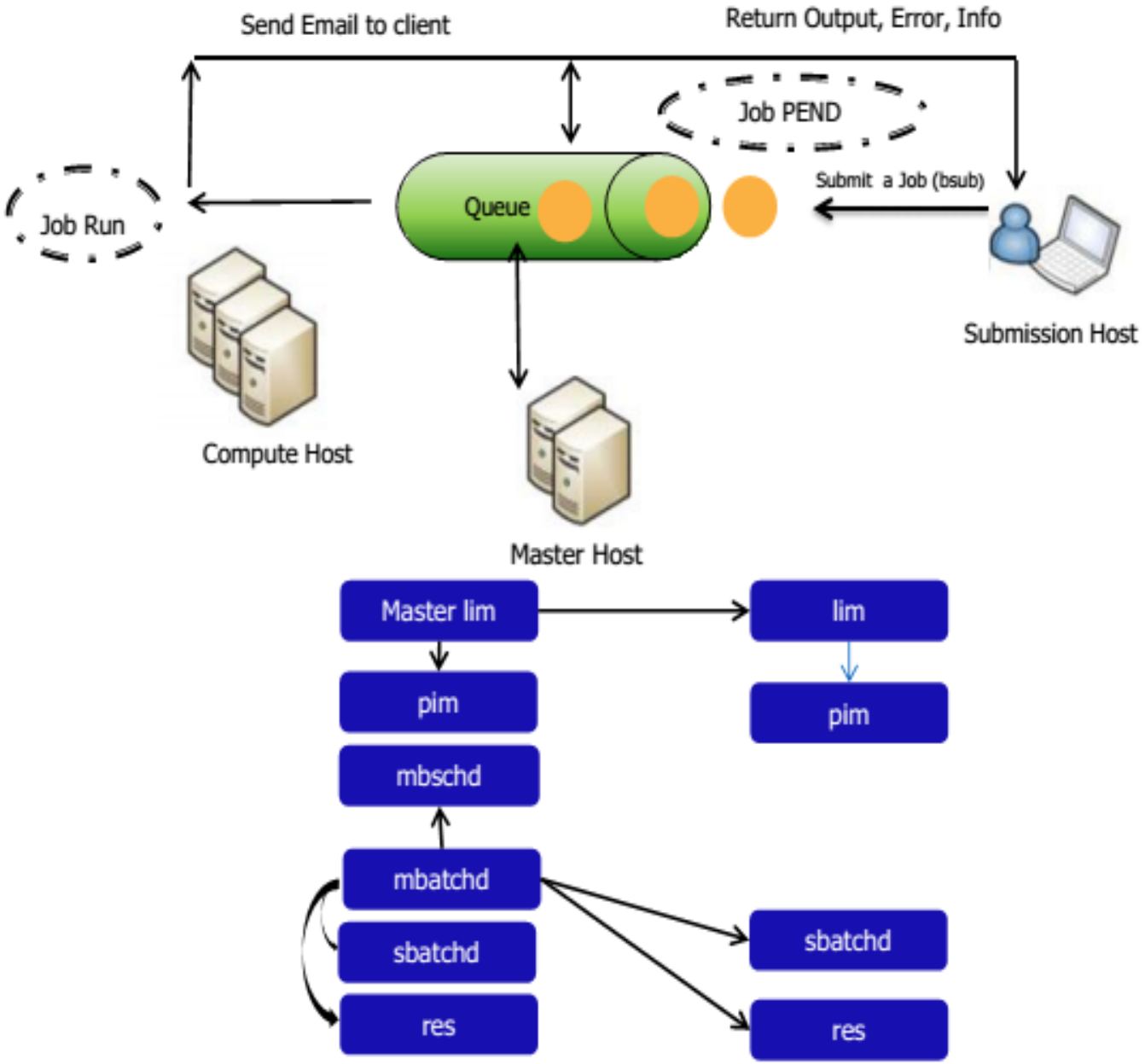
LSF Daemons

- Res
  - Remote Execution Server (**res**) running on each server host.
  - Accepts remote execution requests to provide transparent and secure remote execution of jobs and tasks.
- Lim
  - Load Information Manager (LIM) running on each server host.
  - Collects host load and configuration information and forwards it to the master LIM running on the master host.
  - Reports the information that is displayed by **lsload** and **lshosts**.
  - Static indices are reported when the LIM starts up or when the number of CPUs (ncpus) change. Static indices are:
    - Number of CPUs (ncpus)
    - Number of disks (ndisks)
    - Total available memory (maxmem)
    - Total available swap (maxswp)
    - Total available temp (maxtmp)
    - Dynamic indices for host load collected at regular intervals are:
    - Hosts status (status)
    - 15 second, 1 minute, and 15 minute run queue lengths (r15s, r1m, and r15m)
    - CPU utilization (ut)
    - Paging rate (pg)
    - Number of login sessions (ls)
    - Interactive idle time (it)
    - Available swap space (swp)
    - Available memory (mem)
    - Available temp space (tmp)
    - Disk IO rate (io)

LSF Daemons

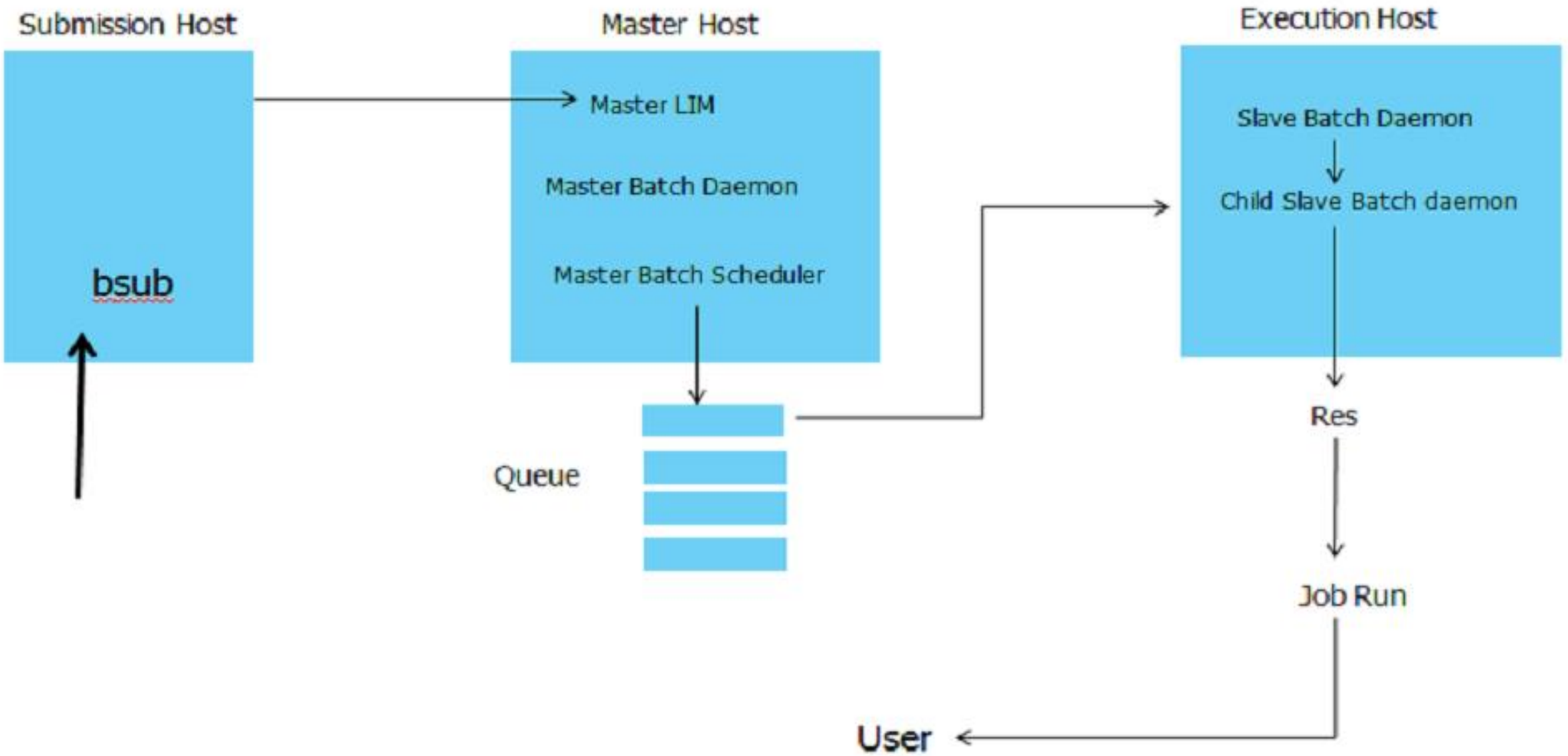
- Master Lim
  - The LIM running on the master host. Receives load information from the LIMs running on hosts in the cluster.
  - Forwards load information to **mbatchd**, which forwards this information to **mbschd** to support scheduling decisions. If the master LIM becomes unavailable, a LIM on another host automatically takes over.
- ELIM
  - External LIM (ELIM) is a site-definable executable that collects and tracks custom dynamic load indices. An ELIM can be a shell script or a compiled binary program, which returns the values of the dynamic resources you define. The ELIM executable must be named elim and located in **LSF_SERVERDIR**.
- Pim
  - Process Information Manager (PIM) running on each server host.
  - Started by LIM, which periodically checks on **pim** and restarts it if it dies.
  - Collects information about job processes running on the host such as CPU and memory that is used by the job, and reports the information to **sbatchd**.

LSF Jobs Work Flow



1. Submit a job. You submit a job from an LSF client or server with the bsub command. If you do not specify a queue when you submit the job, the job is submitted to the default queue. Jobs are held in a queue and wait to be scheduled. These jobs are in the PEND state.
2. Schedule job. The master batch daemon (mbatchd) looks at jobs in the queue and sends the jobs for scheduling to the master batch scheduler (mbschd) at a preset time interval. mbschd evaluates jobs and makes schedulig decisions that are based on job priority, scheduling policies, and available resources. mbschd selects the best hosts where the job can run and sends its decisions back to mbatchd. Resource information is collected at preset time intervals by the master load information manager (LIM) daemon from LIMs on server hosts. The master LIM communicates this information to mbatchd, which in turn communicates it to mbschd to support scheduling decisions.
3. Dispatch the job. As soon as mbatchd receives scheduling decisions, it immediately dispatches the jobs to hosts.
4. Run job. The slave batch daemon (sbatchd):
    a. Receives the request from mbatchd
    b. Creates a child sbatchd for the job
    c. Creates the execution environment
    d. Starts the job by using a remote execution server (res).
5. Return output. When a job is completed, it is assigned the DONE status if the job completed without any problems. The job is assigned the EXIT status if errors prevented the job from completing. sbatchd communicates job information, including errors and output to mbatchd.
6. Send email to client. mbatchd returns the job output, job error, and job information to the submission host through email.

LSF Daemons process flow



LSF Startup

```
[root@masterlsf ~]# lsfstartup
Starting up all LIMs ...
Do you really want to start up LIM on all hosts ? [y/n]y
Start up LIM on <masterlsf> ...... done
Start up LIM on <slavelsf> ...... done
Start up LIM on <node1> ...... done
Start up LIM on <node2> ...... done

Waiting for Master LIM to start up ...  Master LIM is ok
Starting up all RESes ...
Do you really want to start up RES on all hosts ? [y/n]y
Start up RES on <masterlsf> ...... done
Start up RES on <slavelsf> ...... done rsh: No such file or directory
Start up RES on <node1> ...... done
Start up RES on <node2> ...... done

Starting all slave daemons on LSBATCH hosts ...
Do you really want to start up slave batch daemon on all hosts ? [y/n] y
Start up slave batch daemon on <masterlsf> ...... done
Start up slave batch daemon on <slavelsf> ...... done
Start up slave batch daemon on <node1> ...... done
Start up slave batch daemon on <node2> ...... done

Done starting up LSF daemons on the local LSF cluster ...
```

LSF Shutdown

```
[root@masterlsf ~]# lsfshutdown
Shutting down all slave batch daemons ...

Shut down slave batch daemon on all the hosts? [y/n] y
Shut down slave batch daemon on <masterlsf> ..... done
Shut down slave batch daemon on <node1> ..... done
Shut down slave batch daemon on <node2> ..... done
Shut down slave batch daemon on <slavelsf> ..... done


Shutting down all RESes ...
Do you really want to shut down RES on all hosts? [y/n] y
Shut down RES on <masterlsf> ..... done
Shut down RES on <slavelsf> ..... done
Shut down RES on <node1> ..... done
Shut down RES on <node2> ..... done


Shutting down all LIMs ...
Do you really want to shut down LIMs on all hosts? [y/n] y
Shut down LIM on <masterlsf> ..... done
Shut down LIM on <slavelsf> ..... done
Shut down LIM on <node1> ..... done
Shut down LIM on <node2> ..... done
[root@masterlsf ~]#
```

LSF Jobs

LSF Job

- **Job**

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

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

- **Job slot**

  A job slot is a bucket into which a single unit of work is assigned in the LSF system. Hosts are configured to have a number of job slots available and queues dispatch jobs to fill job slots.

- **Job states**

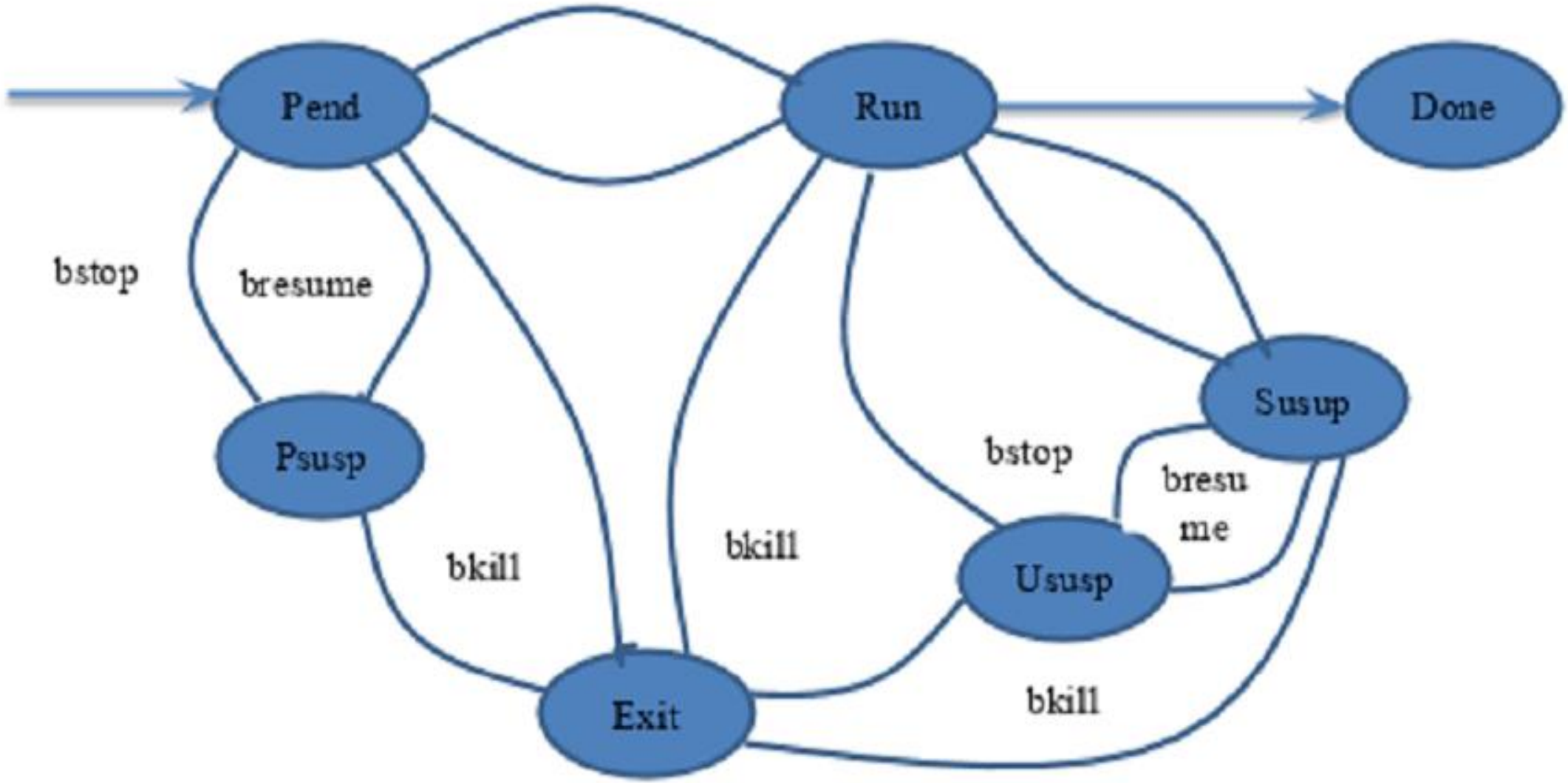
  LSF jobs have the following states:

  - PEND — Waiting in a queue for scheduling and dispatch
  - RUN — Dispatched to a host and running
  - DONE — Finished normally with zero exit value
  - EXIT — Finished with non-zero exit value
  - PSUSP — Suspended while pending
  - USUSP — Suspended by user
  - SSUSP — Suspended by the LSF system
  - POST_DONE — Post-processing completed without errors
  - POST_ERR — Post-processing completed with errors
  - WAIT — Members of a chunk job that are waiting to run

- **Configuration**

  Define job slot limits in lsb.resources.

LSF Batch Job state



LSF batch jobs Continued

- Parallel Job process

When LSF starts running a job LSF assigns a node list to a variable LSB_HOSTS . The first node from the list will be controlling the jobs running on all of the hosts.

- Parallel Job Submission

LSF can select more than one slot for the execution of the job. For parallel job more than one slot is required for execution of job.

  - Steps
    - To submit a parallel job use bsub –n option where " n " is the number of threads you want to run.

```
# bsub -n 16 -o result.out -e result.err /opt/software/openmpi1.6/bin/mpirun ./xhpl
```

LSF batch jobs Continued

- **Specify a minimum and maximum number of processors**

We can specify the minimum and maximum number of processor while submitting a job.

Syntax:

bsub –n min_proc , max proc

bsub –n 4,16 myjob

```
[testuser1@masterlsf ~]$ bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1459    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 11:26
                                             node1
                                             node1
                                             node1
                                             node1
                                             node1
                                             node2
                                             node2
                                             node2
                                             node2
                                             node2
                                             node2
                                             node2
```

LSF batch jobs Continued

```
[testuser1@masterlsf ~]$ bsub -n 2,4 -q long sleep 100
Job <1460> is submitted to queue <long>.
[testuser1@masterlsf ~]$ bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1459    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 11:26
                                             node1
                                             node1
                                             node1
                                             node1
                                             node1
                                             node1
                                             node2
                                             node2
                                             node2
                                             node2
                                             node2
                                             node2
                                             node2
1460    testuse RUN   long       masterlsf   node1       sleep 100  Jan 20 11:27
                                             node1
                                             node2
                                             node2
```

LSF batch jobs Continued

- Distribute Processor allocation across hosts
  Sometimes you need to distribute the work across the hosts.

Syntax

The span string supports the following syntax:

Span[hosts=1] [All the processors from same]

Span[ptile=8] [The job has requested 8 processors on available host]

bsub -n 16 -q long -R span[ptile=8] ./myjob

Job Submission Script

```
[testuser1@masterlsf ~]$ cat job1
#!/bin/bash
#BSUB -o outfile          [ Appends the standard output of file specify a outfile name ]
#BSUB -e errorfile        [ specify a error file name ]
#BSUB -q long             [ specify a queue name ]
#BSUB -n 8                [ specify a number of threads ]
#BSUB -sp 3               [ specify a Job Priority ]
#BSUB -J test1            [ specify a Job name]
#BSUB -W 5                [ Run Limit of a Job ]
#BSUB -R "span[ptile=4]"  [ use 4 cpu's on every node ]

/opt/software/openmpi1.6/bin/mpirun ./xhpl

[testuser1@masterlsf ~]$  bsub < job1
```

```
[testuser1@masterlsf ~]$ bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1805    testuse RUN   long       masterlsf   node1       test1      Jan 22 13:02
                                             node1
                                             node1
                                             node1
                                             node2
                                             node2
                                             node2
                                             node2
```

Example: Matlab Job Submission

```
#!/bin/bash
#BSUB -J MySharedMatlab          [ Job Name ]
#BSUB -q long                    [ long     ]
#BSUB -R "rusage[mem=2GB]“       [ Resource Memory  ]
#BSUB -B                         [ Notify when execution begins ]
#BSUB -N                         [ Notify when execution ends ]
#BSUB -u                         [ specify your email address if it is not default ]
#BSUB -o Output.txt              [ Out file name ]
#BSUB -e Error.txt               [ Error file name ]
#BSUB -cwd   /home/user          [ Specify the current working directory for out file  if required ]
#BSUB -W 04:00                   [ Wall clock time for execution ]
#BSUB -n 4                       [ Number of processors requested ]
#BSUB -R “span[hosts=1]”         [ specify the processor distribution ]


matlab -nodisplay -r my_shared_matlab -logfile MySharedMatlabOut
```

Example : WRF Submission script

```
#!/bin/bash
#BSUB -J wrftest              # job name
#BSUB -W 01:00                # wall-clock time (hrs:mins)
#BSUB -n 64                   # number of tasks in job
#BSUB -R "span[ptile=16]"     # run 16 MPI tasks per node
#BSUB -q normal               # queue
#BSUB -e mpi.error.%J         # error file name in which %J is replaced by the job ID
#BSUB -o mpi.output.%J        # output file name in which %J is replaced by the job ID
#BSUB -x                      # Exclusive execution mode. The job is running exclusively on a host
rm -f hostfile
cat $LSB_DJOB_HOSTFILE > hostfile

export SAVE_ALL_TASKS=no
export LD_LIBRARY_PATH=/gpfs1/home/Libs/INTEL/DAPL/dapl-2.1.10/lib:$LD_LIBRARY_PATH
ulimit -c unlimited
export I_MPI_HYDRA_BOOTSTRAP=ssh
export I_MPI_DEBUG=5
export I_MPI_DAPL_PROVIDER=ofa-v2-mlx4_0-1u
export I_MPI_FABRICS=shm:dapl
export DAPL_UCM_REP_TIME=2000
export DAPL_UCM_RTU_TIME=2000
export DAPL_UCM_CQ_SIZE=2000
export DAPL_UCM_QP_SIZE=2000
export DAPL_UCM_RETRY=7
export DAPL_ACK_RETRY=7
export DAPL_ACK_TIMER=20
export I_MPI_DAPL_UD=enable
export I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD=2097152
export I_MPI_FALLBACK=0
export FORT_BUFFERED=yes

/usr/bin/time -p mpiexec.hydra -f ./hostfile -perhost 16 -np 64 -genvall $EXECUTABLE_NAME
```

LSF BSUB Exclusive Feature

- #BSUB –x [Exclusively submits job on the node]

```
[testuser1@masterlsf ~]$ bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1824    testuse RUN   long       masterlsf   node1       test1      Jan 23 12:23
                                             node1
                                             node1
                                             node1
1827    testuse RUN   long       masterlsf   node2       test1      Jan 23 12:26
                                             node2
                                             node2
                                             node2
1828    testuse PEND  long       masterlsf               test1      Jan 23 12:26
1829    testuse PEND  long       masterlsf               test1      Jan 23 12:26
```

```
[root@masterlsf configdir]# bjobs -p 1828
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1828    testuse PEND  long       masterlsf               test1      Jan 23 12:26
 Job's requirement for exclusive execution not satisfied: 1 host;
 Job slot limit reached: 1 host;
 Not specified in job submission: 1 host;
 Load information unavailable: 1 host;
[root@masterlsf configdir]#
```

LSF BSUB Options

```
#!/bin/bash
#BSUB -o outfile                                        [ Appends the standard output of file specify a outfile name ]
#BSUB -e errorfile                                      [ specify a error file name ]
#BSUB -q long                                           [ specify a queue name ]
#BSUB -n 8                                              [ specify a number of threads ]
#BSUB -sp 3                                             [ specify a Job Priority ]
#BSUB -J test1                                          [ specify a Job name]
#BSUB -W 5                                              [ Run Limit set to 5 Minutes  ]
#BSUB -R "span[ptile=4]"                                [ use 4 cpu's on every node ]
#BSUB -R " rusage [mem=20, license=1:duration=2]"       [ License resource requested for 2 minutes ]
#BSUB -c 10                                             [ CPU LIMIT Set to 10 mins ]
#BSUB -D 1024                                           [ Sets per-process (soft) data segment size ]
#BSUB -F 1024                                           [ Sets a per-process (soft) file size limit for each of the processes ]
#BSUB -M 1024                                           [ Memory Limit ]
```

LSF BSUB Options Examples

```
[testuser1@masterlsf ~]$ bsub -c 1   sleep 70
Job <1955> is submitted to default queue <normal>.

 CPULIMIT
 1.0 min of node1
Wed Jan 23 18:40:14: Started 1 Task(s) on Host(s) <node1>, Allocated 1 Slot(s)


 bsub -D 1024   sleep 70
 DATALIMIT
       1 G
Wed Jan 23 18:46:15: Started 1 Task(s) on Host(s) <node1>, Allocated 1 Slot(s)


 bsub -F 1024 sleep 70

 FILELIMIT
   1024 K
Wed Jan 23 18:56:25: Started 1 Task(s) on Host(s) <node1>, Allocated 1 Slot(s)


 bsub -M 1024 sleep 70
 MEMLIMIT
       1 G
Wed Jan 23 19:07:17: Started 1 Task(s) on Host(s) <node1>, Allocated 1 Slot(s)
```

LSF User Commands Examples

- lsid

```
[root@node1 ~]# lsid
IBM Platform LSF Express 9.1.3.0 for IBM Platform HPC, Jul 04 2014
Copyright IBM Corp. 1992, 2014. All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

My cluster name is cluster1
My master name is masterlsf
```

- bqueues

```
[testuser1@masterlsf ~]$ bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
owners            43  Open:Active      -    -    -    -     0     0     0     0
priority          43  Open:Active      -    -    -    -     0     0     0     0
night             40  Open:Active      -    -    -    -     0     0     0     0
chkpnt_rerun_qu   40  Open:Active      -    -    -    -     0     0     0     0
medium            40  Open:Active      -    -    -    -     0     0     0     0
short             35  Open:Active      -    -    -    -     0     0     0     0
license           33  Open:Active      -    -    -    -     0     0     0     0
normal            30  Open:Active      -    -    -    -    16     0    16     0
interactive       30  Open:Active      -    -    -    -     0     0     0     0
small             30  Open:Active      -    -    -    -     0     0     0     0
idle              20  Open:Active      -    -    -    -     0     0     0     0
[testuser1@masterlsf ~]$
```

LSF User Commands Examples

- **bhosts [Display Static and Dynamic resource Information]**

```
[root@masterlsf conf]# bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
masterlsf          ok              -      8      0      0      0      0      0
node1              ok              -      8      0      0      0      0      0
node2              ok              -      8      0      0      0      0      0
slavelsf           ok              -      8      0      0      0      0      0
```

- **lshosts [Displays static resource information]**

```
[testuser1@masterlsf ~]$ lshosts
HOST_NAME      type    model  cpuf ncpus maxmem maxswp server RESOURCES
masterlsf    X86_64 Intel_EM  60.0     8  15.9G   7.8G    Yes (mg)
slavelsf     X86_64 Intel_EM  60.0     8  15.9G   7.9G    Yes (mg)
node1        X86_64 Intel_EM  60.0     8  15.9G   7.9G    Yes ()
node2        X86_64 Intel_EM  60.0     8  15.9G   7.9G    Yes ()
```

- **lsload [Display load Information]**

```
lsload
HOST_NAME       status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
masterlsf           ok   0.0   0.0   0.0   0%   0.0   1     0   17G  7.8G   15G
slavelsf            ok   0.0   0.0   0.0   0%   0.0   0  1338   51G  7.9G 15.2G
node1               ok   3.9   2.2   1.3  28%   0.0   0  1183   39G  7.9G 14.7G
node2               ok   4.3   2.2   0.7  25%   0.0   0    19   50G  7.9G 14.7G
[testuser1@masterlsf ~]$
```

LSF User Commands Examples

- **bparams [Display the cluster parameters]**

```
[testuser1@masterlsf ~]$bparams
Default Queues:  dispatch
Default Host Specification:  masterlsf
MBD_SLEEP_TIME used for calculations:  20 seconds
Job Checking Interval:  15 seconds
Job Accepting Interval:  0 seconds
[testuser1@masterlsf ~]$
```

- **bhosts [Display Host Information]**

```
[root@masterlsf configdir]# bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
masterlsf          ok              -      8      0      0      0      0      0
node1              closed          -      8      0      0      0      0      0
node2              ok              -      8      0      0      0      0      0
slavelsf           ok              -      8      0      0      0      0      0
[root@masterlsf configdir]#
```

- **bhosts –l [Display the long information]**

```
[root@masterlsf configdir]# bhosts -l node1
HOST  node1
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
closed_Full     60.00     -      8      8      8      0      0      0         -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem  slots
 Total           0.9   0.1   0.0   60%   0.0    62    0  1160   39G  7.9G 14.3G      0
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M      -


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


[root@masterlsf configdir]#
```

LSF User Commands Examples

- **bjobs -u all [show jobs for all users]**

```
[testuser1@masterlsf ~]$ bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
374     testuse RUN   small      masterlsf   node1       sleep 50   Jan 18 17:45
375     testuse RUN   small      masterlsf   node1       sleep 50   Jan 18 17:46
```

- **bjobs –p [show job pending reasons]**

```
[testuser1@masterlsf ~]$ bjobs -p
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1483    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 11:56
 Not enough hosts to meet the job's spanning requirement;
[testuser1@masterlsf ~]$
```

- **bjobs –u all –m node1 [Check if "node1" is a part of any job]**

```
[root@masterlsf conf]# bjobs -u all -m node1
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1899    testuse RUN   long       masterlsf   node1           *      Jan 22 08:16
                                             node1
```

- **bjobs -l 1816 [Job information in long format]**

```
[testuser1@masterlsf ~]$ bjobs -l

Job <1816>, Job Name <test1>, User <testuser1>, Project <default>, Status <RUN>
                     , Queue <long>, Job Priority <5>, Command <#!/bin/bash ; #
                     BSUB -o outfile;#BSUB -e errorfile;#BSUB -q long ;#BSUB -n
                      16 ;#BSUB -J test1;#BSUB -w 5 ;#BSUB -R "span[ptile=8]";
                     /opt/software/openmpi1.6/bin/mpirun ./xhpl>, Share group c
                     harged </testuser1>
```

LSF User Commands Examples

- **bbot**

```
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1534    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1535    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1536    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1537    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1538    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
1539    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
```

```
[testuser1@masterlsf ~]$ bbot 1538
Job <1538> has been moved to position 1 from bottom.
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1534    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1535    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1536    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1537    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1539    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
1540    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
1541    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
1542    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
1543    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
1538    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
```

LSF User Commands Examples

- btop

```
[testuser1@masterlsf ~]$ btop 1538
Job <1538> has been moved to position 1 from top.
```

```
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1534    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1535    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1536    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1537    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:23
                                             node2
                                             node1
                                             node1
1538    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
1539    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:23
```

LSF User Commands Examples

- **breque**

```
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1544    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1545    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1546    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1547    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1548    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1549    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1550    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1551    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1552    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
```

```
[testuser1@masterlsf ~]$ brequeue 1544
Job <1544> is being requeued
```

```
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1545    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1546    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1547    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1548    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 16:26
                                             node1
                                             node2
                                             node2
1549    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1550    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1551    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1552    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1544    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
```

LSF User Commands Examples

- bmod

```
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1545    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1546    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1547    testuse RUN   long       masterlsf   node2       *un ./xhpl Jan 20 16:26
                                             node2
                                             node1
                                             node1
1548    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 16:26
                                             node1
                                             node2
                                             node2
1549    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1550    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1551    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1552    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1544    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
```

```
[testuser1@masterlsf ~]$ bmod -q medium 1552
Parameters of job <1552> are being changed
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1548    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 16:26
                                             node1
                                             node2
                                             node2
1549    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 16:26
                                             node1
                                             node2
                                             node2
1550    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 16:26
                                             node1
                                             node2
                                             node2
1551    testuse RUN   long       masterlsf   node1       *un ./xhpl Jan 20 16:26
                                             node1
                                             node2
                                             node2
1544    testuse PEND  long       masterlsf               *un ./xhpl Jan 20 16:26
1552    testuse PEND  medium     masterlsf               *un ./xhpl Jan 20 16:26
[testuser1@masterlsf ~]$
```

LSF User Commands Examples

- bmod -cwd /opt/software 1949 [Change of Working Directory]

```
[testuser1@masterlsf ~]$ bjobs -l 1949

Job <1949>, Job Name <test1>, User <testuser1>, Project <default>, Status <PEND
                     >, Queue <long>, Job Priority <2>, Command <#!/bin/bash ;
                     #BSUB -o outfile1;#BSUB -e errorfile1;#BSUB -q long ;#BSUB
                      -n 4 ;#BSUB -J test1;#BSUB -sp 2 ;##BSUB -x ;#BSUB -W 2 ;
                     #BSUB -R "span[ptile=2]"; /opt/
                     software/openmpi1.6/bin/mpirun  ./xhpl>
Wed Jan 23 14:27:34: Submitted from host <masterlsf>, CWD <$HOME>, Output File
                     <outfile1>, Error File <errorfile1>, Re-runnable, 4 Task(s
                     ), Requested Resources <span[ptile=2]>, User Priority <2>;

[testuser1@masterlsf ~]$ bmod -cwd  /opt/software 1949
Parameters of job <1949> are being changed

[testuser1@masterlsf ~]$ bjobs -l 1949

Job <1949>, Job Name <test1>, User <testuser1>, Project <default>, Status <PEND
                     >, Queue <long>, Job Priority <2>, Command <#!/bin/bash ;
                     #BSUB -o outfile1;#BSUB -e errorfile1;#BSUB -q long ;#BSUB
                      -n 4 ;#BSUB -J test1;#BSUB -sp 2 ;##BSUB -x ;#BSUB -W 2 ;
                     ###BSUB -cwd /opt/software;#BSUB -R "span[ptile=2]"; /opt/
                     software/openmpi1.6/bin/mpirun  ./xhpl>
Wed Jan 23 14:27:34: Submitted from host <masterlsf>, CWD <$HOME>, Specified CW
                     D </opt/software>, Output File <outfile1>, Error File <err
                     orfile1>, Re-runnable, 4 Task(s), Requested Resources <spa
                     n[ptile=2]>, User Priority <2>;
```

LSF User Commands Examples

- **Job running in medium queue**

```
[testuser1@masterlsf ~]$ bswitch bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
751     testuse RUN   medium     masterlsf   node1       sleep 100  Jan 18 19:53
752     testuse RUN   medium     masterlsf   node1       sleep 100  Jan 18 19:53
```

- **Job Switch from medium queue to small queue**

```
[testuser1@masterlsf ~]$ bswitch -q medium small 0
Job <751> is switched to queue <small>
Job <752> is switched to queue <small>
[testuser1@masterlsf ~]$ bswitch -q medium small 0jobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
751     testuse RUN   small      masterlsf   node1       sleep 100  Jan 18 19:53
752     testuse RUN   small      masterlsf   node1       sleep 100  Jan 18 19:53
```

- **Job History**

```
[testuser1@masterlsf ~]$ bhist -l 751

Job <751>, User <testuser1>, Project <default>, Command <sleep 100>
Fri Jan 18 19:53:15: Submitted from host <masterlsf>, to Queue <medium>, CWD <$
                     HOME>, Re-runnable;
Fri Jan 18 19:53:16: Dispatched 1 Task(s) on Host(s) <node1>, Allocated 1 Slot(
                     s) on Host(s) <node1>, Effective RES_REQ <select[type == l
                     ocal] order[r15s:pg] >;
Fri Jan 18 19:53:16: Starting (Pid 5420);
Fri Jan 18 19:53:16: Running with execution home </home/testuser1>, Execution C
                     WD </home/testuser1>, Execution Pid <5420>;
Fri Jan 18 19:53:40: Switched to Queue <small> by user or administrator <testus
                     er1>;
Fri Jan 18 19:54:56: Done successfully. The CPU time used is 0.2 seconds;
Fri Jan 18 19:54:56: Post job process done successfully;

MEMORY USAGE:
MAX MEM: 1 Mbytes;  AVG MEM: 1 Mbytes

Summary of time in seconds spent in various states by  Fri Jan 18 19:54:56
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  1        0        100      0        0        0        101
```

LSF Jobs Exit Values

- LSF Job Exit values
  - Exit Value of the jobs can be checked using bhist command.
  - Exit value less than 128 (< 128) is related to Applications
  - Exit value greater than 128 (> 128) means job terminated by system signals.

Example:-

Exit Values	Meaning
255	Job has got exited due to SSH problem
140	The CPU time uses Exceeded
5	SIGTRAP
2	SIGINIT
7	SIGV

LSF Queues

LSF Queues

- **Queue**

  A cluster wide bucket for jobs. All jobs wait in queues until they are scheduled and dispatched to hosts. Queues do not correspond to individual hosts; each queue can use all server hosts in the cluster, or a configured subset of the server hosts.

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

  Queues implement different job scheduling and control policies.

- **Queue Configuration**

  Define queues in lsb.queues

- **Adding a Queue**

  Edit the lsb.queues file to add the new queue definition. Adding a queue does not affect pending or running jobs.

LSF Queues Continued

Example:- **Default queues**

```
[root@masterlsf configdir]# bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
owners           43  Open:Active      -    -    -    -     0     0     0     0
priority         43  Open:Active      -    -    -    -     0     0     0     0
night            40  Open:Inact       -    -    -    -     0     0     0     0
chkpnt_rerun_qu  40  Open:Active      -    -    -    -     0     0     0     0
short            35  Open:Active      -    -    -    -     0     0     0     0
license          33  Open:Active      -    -    -    -     0     0     0     0
normal           30  Open:Active      -    -    -    -     0     0     0     0
interactive      30  Open:Active      -    -    -    -     0     0     0     0
idle             20  Open:Active      -    -    -    -     0     0     0     0
[root@masterlsf configdir]#
```

```
Begin Queue
    Queue_NAME              = long
    DESCRIPTION             = For long jobs
    RERUNNABLE              = YES
    PRIORITY                = 40
    NEW_JOB_SCHED_DELAY     = 1
    HOSTS                   = node1 node2
    EXCLUSIVE               = Y
End Queue
```

LSF Queues Continued

- **Example: Job not switching from queue**

```
[testuser1@masterlsf ~]$ bsub -q medium sleep 200
Job <753> is submitted to queue <medium>.
```

```
[testuser1@masterlsf ~]$ bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
753     testuse RUN   medium     masterlsf   node2       sleep 200  Jan 18 19:57
```

```
[testuser1@masterlsf ~]$ bswitch -q medium small 753
small: Host or host group is not used by the queue
```

- **The node node2 is not belongs to small queue.**

```
Begin Queue
          Queue_NAME                  = small
          DESCRIPTION                 = For small jobs
          RERUNNABLE                  = YES
          PRIORITY                    = 30
          NEW_JOB_SCHED_DELAY         = 1
          HOSTS                       = node1
          EXCLUSIVE                   = Y
End Queue
```

- **Example: When Accidentally Deleted/Removed a queue**

LSF Queues Continued

- **Queue Example**

```
 Begin Queue
QUEUE_NAME   = normal
PRIORITY     = 30
INTERACTIVE  = NO
#RUN_WINDOW   = 5:19:00-1:8:30 20:00-8:30
#r1m          = 0.7/2.0        # loadSched/loadStop
#r15m         = 1.0/2.5
#pg           = 4.0/8
#ut           = 0.2
#io           = 50/240
#CPULIMIT     = 180/hostA      # 3 hours of host hostA
#FILELIMIT    = 20000
#DATALIMIT    = 20000          # jobs data segment limit
#CORELIMIT    = 20000
#TASKLIMIT    = 5              # job task limit
#USERS        = all            # users who can submit jobs to this queue
#HOSTS        = all            # hosts on which jobs in this queue can run
#PRE_EXEC     = /usr/local/lsf/misc/testq_pre >> /tmp/pre.out
#POST_EXEC    = /usr/local/lsf/misc/testq_post |grep -v "Hey"
#REQUEUE_EXIT_VALUES = 55 34 78
#APS_PRIORITY = WEIGHT[[RSRC, 10.0] [MEM, 20.0] [PROC, 2.5] [QPRIORITY, 2.0]] \
#    LIMIT[[RSRC, 3.5] [QPRIORITY, 5.5]] \
#    GRACE_PERIOD[[QPRIORITY, 200s] [MEM, 10m] [PROC, 2h]]
DESCRIPTION  = For normal low priority jobs, running only if hosts are \
lightly loaded.
End Queue
```

LSF Queues Continued

- **bqueues -l** small [also displays the comment text:]

```
[root@masterlsf software]# bqueues -l small

QUEUE: small
  -- For small jobs

PARAMETERS/STATISTICS
PRIO NICE STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV
 30    0  Open:Inact_Adm    -    -    -    -     1     1     0     0     0    0
Schedule delay for a new job is 1 seconds
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:  EXCLUSIVE

USERS: all
HOSTS:  node1
RERUNNABLE :  yes

ADMIN ACTION COMMENT: "testing"
[root@masterlsf software]#
```

LSF Hosts

LSF Hosts

- **Hosts**

  A host is an individual computer in the cluster. Each host may have more than 1 processor. Multiprocessor hosts are used to run parallel jobs. A multiprocessor host with a single process queue is considered a single machine, while a box full of processors that each have their own process queue is treated as a group of separate machines.

- **Submission host**

  The host where jobs are submitted to the cluster. Jobs are submitted using the **bsub** command or from an application that uses the LSF API. Client hosts and server hosts can act as submission hosts.

- **Execution host**

  The host where a job runs. Can be the same as the submission host. All execution hosts are server hosts.

- **Server host**

  Hosts that are capable of submitting and executing jobs. A server host runs sbatchd to execute server requests and apply local policies.

LSF Hosts Continued

- **Client host / Login Node / Submission Node**

  Hosts that are only capable of submitting jobs to the cluster. Client hosts run LSF commands and act only as submission hosts. Client hosts do not execute jobs or run LSF daemons

- **Master host**

  Where the master LIM and **mbatchd** run. An LSF server host that acts as the overall coordinator for that cluster. Each cluster has one master host to do all job scheduling and dispatch. If the master host goes down, another LSF server in the cluster becomes the master host.

  All LSF daemons run on the master host. The LIM on the master host is the master LIM.

LSF Hosts Continued

- **Hosts are Opened and Closed by:**

Close a host

```
[root@masterlsf configdir]# badmin hclose -C "disk problem"  node1
Close <node1> ...... done
[root@masterlsf configdir]#
```

```
[root@masterlsf configdir]# bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
masterlsf          ok              -      8      0      0      0      0      0
node1              closed          -      8      0      0      0      0      0
node2              ok              -      8      0      0      0      0      0
slavelsf           ok              -      8      0      0      0      0      0
[root@masterlsf configdir]#
```

Open a host

```
[root@masterlsf configdir]# badmin hopen node1
Open <node1> ...... done
```

LSF Hosts Continued

- Use **badmin hhist** to display administrator comments for closing and opening hosts:

```
[testuser1@masterlsf ~]$ bjobs -p
No pending job found
[testuser1@masterlsf ~]$

[root@masterlsf configdir]# badmin hhist node1
Fri Jan 18 18:53:49: Host <node1> closed by administrator <root> memory problem.
Fri Jan 18 18:59:05: Host <node1> opened by administrator <root>.
Sun Jan 20 11:54:36: Host <node1> closed by administrator <root>.
Sun Jan 20 11:54:43: Host <node1> opened by administrator <root>.
Sun Jan 20 11:55:02: Host <node1> closed by administrator <root> disk problem.
Sun Jan 20 11:59:17: Host <node1> opened by administrator <root>.
[root@masterlsf configdir]#
```

- **bhosts -l** node1 also displays the comment text

```
[root@masterlsf configdir]# bhosts -l node1
HOST  node1
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
closed_Full     60.00     -      8      8      8      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem  slots
 Total           0.9   0.1   0.0   60%   0.0    62    0  1160   39G  7.9G 14.3G      0
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M      -


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


[root@masterlsf configdir]#
```

LSF Interactive Batch Job & Tasks

- **Interactive batch job**

  A batch job that allows you to interact with the application and still take advantage of LSF scheduling policies and fault tolerance. All input and output are through the terminal that you used to type the job submission command.

  When you submit an interactive job, a message is displayed while the job is awaiting scheduling. A new job cannot be submitted until the interactive job is completed or terminated.

  The **bsub** command stops display of output from the shell until the job completes, and no mail is sent to you by default. Use Ctrl-C at any time to terminate the job.

  - **Commands**
  - **bsub -I** — Submit an interactive job

```
[testuser1@masterlsf ~]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1486    testuse RUN   interactiv masterlsf   node2       * 8 ./xhpl Jan 20 12:03
[testuser1@masterlsf ~]$
```

- **Interactive task**

  A command that is not submitted to a batch queue and scheduled by LSF, but is dispatched immediately. LSF locates the resources needed by the task and chooses the best host among the candidate hosts that has the required resources and is lightly loaded. Each command can be a single process, or it can be a group of cooperating processes.

  Tasks are run without using the batch processing features of LSF but still with the advantage of resource requirements and selection of the best host to run the task based on load.

  - **Commands**
  - **lsrun** — Submit an interactive task

LSF Interactive Batch Job & Tasks

```
[testuser1@masterlsf ~]$ lsrun -m node1 /opt/software/openmpi1.6/bin/mpirun -n 8 ./xhpl
================================================================================
HPLinpack 2.1  --  High-Performance Linpack benchmark  --   October 26, 2012
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :   10000
NB     :       4
PMAP   : Row-major process mapping
P      :       4
Q      :       2
PFACT  :   Right
NBMIN  :       4
NDIV   :       2
RFACT  :   Right
BCAST  :   1ring
DEPTH  :       0
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
ALIGN  : 8 double precision words
```

LSF Host types and host models

- **Host types**

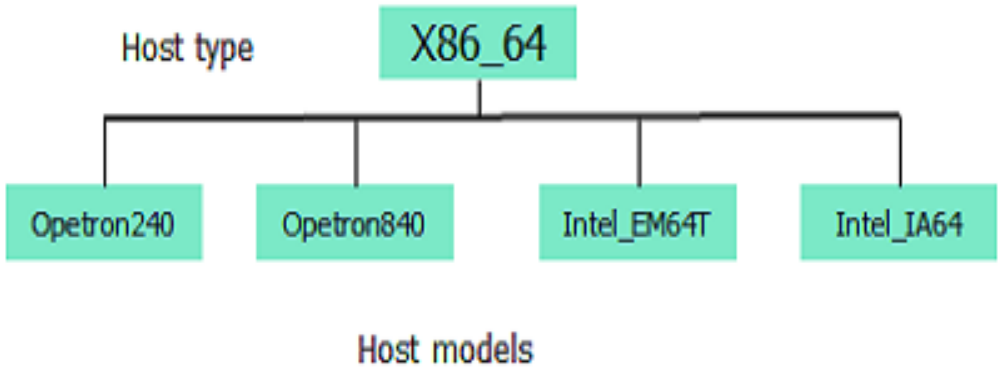
  Hosts in LSF are characterized by host type and host model.

  The Combination of operating system version and host CPU architect

  All computers that run the same operating system on the same computer architecture are of the same type in other words, binary-compatible with each other.

  Each host type usually requires a different set of LSF binary files.

  Commands: lsinfo -t — View all host types



- **Host model**

  The combination of host type and CPU speed (CPU factor) of the computer.

  All hosts of the same relative speed are assigned the same host model.

  The CPU factor is taken into consideration when jobs are being dispatched.

  Commands:

  lsinfo -m — View a list of currently running models

  lsinfo -M — View all models

LSF Resources

LSF Resources

- **Resources**

LSF resources are objects in the LSF system resources that LSF uses track job requirements and schedule jobs according to their availability on individual hosts.

- **Resource usage**

The LSF system uses built-in and configured resources to track resources availability and usage. Jobs are scheduled according to the resources available on individual hosts.

LSF Collects information such as:

- Total CPU time consumed by all processes in the job.
- Total resident memory usage in KB of all currently running processes in a job
- Total virtual memory usage in KB of all currently running processes in a job
- Currently active process group ID in a job
- Currently active processes in a job On UNIX, job-level resource usage is collected through PIM.

Commands:

lsinfo : View the resources available in your cluster

bjobs –l : View current resource usage of a job

LSF Resource

- **Load indices**

  Load indices measure the availability of dynamic, non-shared resources on hosts in the cluster. Load indices built into the LIM are updated at fixed time intervals

  Commands:

  lsload –l : View all load indices

  bhosts –l View load levels on a host

- **External load indices**

  Defined and configured by the LSF administrator and collected by an External Load Information Manager (ELIM) program. The ELIM also updates LIM when new values are received.

  Commands:

  lsinfo : View external load indices

- **Static resources**

  Built-in resources that represent host information that does not change over time, such as the maximum RAM available to user processes or the number of processors in a machine. Most static resources are determined by the LIM at startup.

- Load thresholds Two types of load thresholds can be configured by your LSF administrator to schedule jobs in queues. Each load threshold specifies a load index value:

LSF Resource usage Continued

Commands

bhosts –l : View suspending conditions for hosts

bqueues –l : View suspending conditions for queues

bjobs –l : View suspending conditions for a particular job and the scheduling thresholds that control when a job is resumed

- **Runtime resource usage limits**

  Limit the use of resources while a job is running. Jobs that consume more than the specified amount of a resource are signaled or have their priority lowered.

- **Hard and soft limits**

  Resource limits specified at the queue level are hard limits while those specified with job submission are soft limits.

  Resource allocation limits

  Restrict the amount of given resource that must be available during job scheduling for different classes of jobs to start, and which resource consumers the limits apply to. If all of the resource has been consumed, no more jobs can be started until some of the resource is released.

- **Resource requirements (bsub –R)**

  Restrict which hosts the job can run on. Hosts that match the resource requirements are the candidate hosts. When LSF schedules a job, it collects the load index values of all the candidate hosts and compares them to the scheduling conditions. Jobs are only dispatched to a host if all load values are within the scheduling thresholds.

  Commands

  bsub -R — Specify resource requirement string for a job

LSF Users

LSF Users

- **LSF user**

  A user account that has permission to submit jobs to the LSF cluster.

  LSF administrator

  In general, you must be an LSF administrator to perform operations that will affect other LSF users. Each cluster has one primary LSF administrator, specified during LSF installation. You can also configure additional administrators at the cluster level and at the queue level.

  Primary LSF administrator

  The first cluster administrator specified during installation and first administrator listed in lsf.cluster.cluster_name . The primary LSF administrator account owns the configuration and log files. The primary LSF administrator has permission to perform cluster wide operations, change configuration files, reconfigure the cluster, and control jobs submitted by all users.

  Queue administrator

  An LSF administrator user account that has administrative permissions limited to a specified queue.

LSF Scheduling

IBM LSF Scheduling and dispatch

- LSF allows mltiple schedulling policies in the same cluster. LSF has several queue scheduling policies such as, preemptive, fairshare, and hierarchical fairshare.
  - First-come,first-served(FCFS) scheduling

    By default, jobs in a queue are dispatched in FCFS order. This means that jobs are dispatched according to their order in the queue.
  - Service level agreement(SLA) scheduling

    An SLA in LSF is a "just-in-time" scheduling policy that schedules the services agreed to between LSF administrators and LSF users. The SLA scheduling policy defines how many jobs should be run from each SLA to met the configured goals.
  - Fairshare Scheduling

    If you specify a fairshare scheduling policy for the queue or if host partitions have been configured, LSF dispatches jobs between users based on assigned user shares, resources usage, or other factors.
  - Preemption

    You can specify desired behavior so that when two or more jobs compete for the same resources, one job preempts the other. Preemption can apply to not only job slots, but also to advance reservation)reserving hosts for particular jobs) and licenses(using IBM Platform License Scheduler).
  - Backfill

    Allows small jobs to run on job slots reserved for other jobs, provided the backfilling job completes before the reservation time expires and resource usage is due.

IBM LSF Scheduling and dispatch continued

- **Scheduling and dispatch**

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

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

- **Dispatch Order**

Job are not necessarily dispatched in order of submission.

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

LSF considers jobs for dispatch in the following order:

  - For each queue, from highest to lowest priority. If multiple queues have the same priority, LSF schedules all the jobs from these queues in first-come, first-serverd order.
  - For each job in the queue, according to FCFS order.
  - If any host is eligible to run this job, start the job on the best eligible host, and mark that host ineligible to start any other job until JOB_ACCEPT_INTERVAL has passed.

Thank You