Load Sharing Facility (LSF)

Minerva Scientific Computing Environment

https://labs.icahn.mssm.edu/minervalab

Patricia Kovatch
Eugene Fluder, PhD
Hyung Min Cho, PhD
Lili Gai, PhD
Dansha Jiang, PhD

March 18, 2020
Outline

- LSF introduction and basic LSF commands
- Dependent job
- Self-scheduler
- Parallel jobs: job arrays, parallel processing and GPUs
- Job restart with checkpoint?
- Tips for efficient usage of the queuing system
Distributed Resource Management System (DRMS)

- Goal is to achieve best utilization of resources and maximize throughput for high-performance computing systems.

- Control usage of hard resources
  - CPU cycles; Memory;

- Can be decomposed into subsystems:
  - Job management; Physical resource management; Scheduling and queuing.

- Widely deployed DRMSs
  - Platform Load Sharing Facility (LSF)
  - Portable Batch Systems (PBS)
  - Simple Linux Utility for Resource Management (Slurm)
  - others such as IBM Load Leveler and Condor.
LSF Job Lifecycle

1. submit a job
2. schedule the job
3. dispatch the job
4. run the job
5. return output
6. send Email to client
   (disabled on Minerva)

https://www.ibm.com/
LSF Useful Commands

**bhosts**: Displays hosts and their static and dynamic resources

- List all the compute nodes on Minerva

```
[gail01@li03c03: ~] $ bhosts

<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>JLU</th>
<th>MAX</th>
<th>NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>dor01-1</td>
<td>ok</td>
<td>-</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fangg03-2</td>
<td>ok</td>
<td>-</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>filizm02-5</td>
<td>closed</td>
<td>-</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>filizm02-6</td>
<td>closed</td>
<td>-</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>greenb08-1</td>
<td>ok</td>
<td>-</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gux01-1</td>
<td>ok</td>
<td>-</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>huckil01-1</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>lc01a05</td>
<td>ok</td>
<td>-</td>
<td>48</td>
<td>31</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>lc01a06</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>lc01a07</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>39</td>
</tr>
<tr>
<td>lc01a08</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>lc01a09</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>lc01a10</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>39</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>lc01a11</td>
<td>ok</td>
<td>-</td>
<td>48</td>
<td>42</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>lc01a12</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>lc01a13</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>lc01a14</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>lc01a15</td>
<td>closed</td>
<td>-</td>
<td>48</td>
<td>48</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>41</td>
</tr>
<tr>
<td>lc01a16</td>
<td>ok</td>
<td>-</td>
<td>48</td>
<td>36</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>
```
bhosts: himem, gpu, bode, nonbode (major nodes), interactive

```bash
gail01@li03c03: ~ $ bhosts himem
HOST_NAME | STATUS | JL/U | MAX | NJOBS | RUN | SSUSP | USUSP | RSV
----------|--------|------|-----|-------|-----|-------|-------|-----
lh03c01   | closed | -    | 48  | 48    | 48  | 0     | 0     | 0   
lh03c02   | closed | -    | 48  | 48    | 29  | 0     | 0     | 19  
lh03c03   | closed | -    | 48  | 48    | 26  | 0     | 0     | 22  
lh03c04   | closed | -    | 48  | 48    | 48  | 0     | 0     | 0   
```

```bash
gail01@li03c03: ~ $ bhosts gpu
HOST_NAME | STATUS | JL/U | MAX | NJOBS | RUN | SSUSP | USUSP | RSV
----------|--------|------|-----|-------|-----|-------|-------|-----
lg03a02   | ok     | -    | 32  | 0     | 32  | 0     | 0     | 0   
lg03a03   | closed | -    | 32  | 32    | 32  | 0     | 0     | 0   
lg03a04   | ok     | -    | 32  | 1     | 1   | 0     | 0     | 0   
lg03a05   | ok     | -    | 32  | 0     | 0   | 0     | 0     | 0   
lg03a06   | ok     | -    | 32  | 0     | 0   | 0     | 0     | 0   
lg03a07   | closed | -    | 32  | 32    | 32  | 0     | 0     | 0   
lg03a08   | ok     | -    | 32  | 0     | 0   | 0     | 0     | 0   
lg03a09   | ok     | -    | 32  | 12    | 12  | 0     | 0     | 0   
lg03a10   | ok     | -    | 32  | 0     | 0   | 0     | 0     | 0   
lg03a11   | ok     | -    | 32  | 0     | 0   | 0     | 0     | 0   
lg03a12   | unavail| -    | 32  | 0     | 0   | 0     | 0     | 0   
```

```bash
gail01@li03c03: ~ $ bhosts bode | head
HOST_NAME | STATUS | JL/U | MAX | NJOBS | RUN | SSUSP | USUSP | RSV
----------|--------|------|-----|-------|-----|-------|-------|-----
lc01g17   | ok     | -    | 48  | 37    | 37  | 0     | 0     | 0   
lc01g18   | closed | -    | 48  | 48    | 48  | 0     | 0     | 0   
lc01g19   | ok     | -    | 48  | 37    | 37  | 0     | 0     | 0   
lc01g20   | ok     | -    | 48  | 37    | 37  | 0     | 0     | 0   
lc01g21   | ok     | -    | 48  | 37    | 37  | 0     | 0     | 0   
lc01g22   | ok     | -    | 48  | 17    | 17  | 0     | 0     | 0   
lc01g23   | ok     | -    | 48  | 17    | 17  | 0     | 0     | 0   
```
bhosts: himem, gpu, bode, nonbode (major nodes), interactive

nonbode and himem are usually quite busy, while bode and interactive are usually open to jobs in minutes; Availability of gpu queue varies from time to time
**bqueues**: displays information about all the available queues

```
[gail01@li03c03: ~]$ bqueues

<table>
<thead>
<tr>
<th>QUEUE_NAME</th>
<th>PRIORITY</th>
<th>STATUS</th>
<th>MAX</th>
<th>JL/U</th>
<th>JL/P</th>
<th>JL/H</th>
<th>NJOBS</th>
<th>PEND</th>
<th>RUN</th>
<th>SUSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>premium</td>
<td>200</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>363705</td>
<td>356007</td>
<td>2614</td>
<td>0</td>
</tr>
<tr>
<td>private</td>
<td>130</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1012</td>
<td>804</td>
<td>88</td>
<td>0</td>
</tr>
<tr>
<td>express</td>
<td>120</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2928</td>
<td>1768</td>
<td>728</td>
<td>0</td>
</tr>
<tr>
<td>interactive</td>
<td>100</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>long</td>
<td>100</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3781</td>
<td>3685</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>gpu</td>
<td>100</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>48</td>
<td>0</td>
<td>48</td>
<td>0</td>
</tr>
</tbody>
</table>
```

**bqueues -l interactive**

QUEUE: interactive
   -- For interactive jobs

PARAMETERS/STATISTICS

```
PRIO NICE STATUS      MAX JL/U JL/P JL/H NJOBS PEND  RUN  SSUSP USUSP RSV
100   0  Open:Active  -    -    -    -     4     0     4     0     0     0
```

Interval for a host to accept two jobs is 0 seconds

**DEFAULT LIMITS:**

- RUNLIMIT
  - 120.0 min

**MAXIMUM LIMITS:**

- RUNLIMIT
  - 720.0 min

```
.......  
```

**USERS**: all

**HOSTS**: interactive/
# LSF: Queue structure (bqueues)

<table>
<thead>
<tr>
<th>Queue</th>
<th>Wall time limit</th>
<th>available resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>interactive (Dedicated to interactive jobs)</td>
<td>12 hours</td>
<td>4 nodes + 1 GPU node</td>
</tr>
<tr>
<td>premium</td>
<td>6 days</td>
<td>270 nodes + 4 himem nodes</td>
</tr>
<tr>
<td>express</td>
<td>12 hours</td>
<td>270 nodes + 4 dedicated nodes (may change)</td>
</tr>
<tr>
<td>long</td>
<td>2 weeks</td>
<td>4 dedicated (192 cores)</td>
</tr>
<tr>
<td>gpu</td>
<td>6 days</td>
<td>44 V100</td>
</tr>
<tr>
<td>private</td>
<td>unlimited</td>
<td>private nodes</td>
</tr>
</tbody>
</table>

*default memory: 3000MB / per core*
**bsub - submit a job to LSF (interactive and batch)**

**Interactive jobs:**
- Set up an interactive environment on compute nodes with **internet access**
- Useful for testing and debugging jobs
- **Interactive GPU** is available for job testing

```
bsub -XF -P acc_hpcstaff -q interactive -n 1 -W 2:00 -R rusage[mem=3000] -ls /bin/bash
```
- `-q` : to specify the queue-name from where to get the nodes
- `-ls` : Interactive terminal/shell
- `-n` : to specify the total number of compute cores (job slot) needed
- `-R` : Resource request specifying in a compute node
- `-XF` : X11 forwarding
- `/bin/bash` : the shell to use

```
gail01@li03c03: ~ $ bsub -XF -P acc_hpcstaff -q interactive -n 1 -W 2:00 -R rusage[mem=3000] -ls /bin/bash
Job <2916837> is submitted to queue <interactive>.
<<ssh X11 forwarding job>>
<<Waiting for dispatch ...>>
<<Starting on lc02a29>>
```
bsub - batch job

Create job scripts containing job info and commands to the LSF

bsub [options] < my_batch_job or bsub [options] my_batch_job

- With “<” will interpret the #BSUB cookies in the script.
- Options on the command line override what is in the script

```
gail01@li03c03: ~ $ cat myfirst.lsf
#!/bin/bash
#BSUB -J myfirstjob
# Job name
#BSUB -P acc_hpcstaff
# allocation account
#BSUB -q premium
# queue
#BSUB -n 1
# number of compute cores
#BSUB -W  6:00
# walltime in HH:MM
#BSUB -R rusage[mem=4000]
# 4 GB of memory requested
#BSUB -o %J.stdout
# output log (%J : JobID)
#BSUB -eo %J.stderr
# error log
#BSUB -L /bin/bash
# Initialize the execution environment

module load gcc
which gcc
```

echo “Hello Chimera”
```
gail01@li03c03: ~ $ bsub < myfirst.lsf
Job <2937044> is submitted to queue <premium>.
Submit the script with the bsub command:

bsub < myfirst.lsf
**bjobs - status of jobs**

- **Check your own jobs:** `$bjobs`

  ```
gail01@li03c03: ~ $ bjobs
    JOBID USER JOB_NAME STAT QUEUE FROM_HOST EXEC_HOST SUBMIT_TIME START_TIME TIME_LEFT
    2937044 gail01 myfirstjob PEND premium li03c03 - Sep 10 14:38 - -
  ```

- **Check all jobs:** `$bjobs -u all`

  ```
  JOBID USER JOB_NAME STAT QUEUE FROM_HOST EXEC_HOST SUBMIT_TIME START_TIME TIME_LEFT
  2845103 beckmn01 *>junkK.432 RUN premium regen2 lc02e24 Sep 9 21:19 Sep 10 14:25 23:57 L
  2845113 beckmn01 *>junkK.442 RUN premium regen2 lc02e24 Sep 9 21:19 Sep 10 14:26 23:58 L
  2845088 beckmn01 *>junkK.417 RUN premium regen2 lc04a10 Sep 9 21:18 Sep 10 14:23 23:55 L
  2845089 beckmn01 *>junkK.418 RUN premium regen2 lc04a10 Sep 9 21:18 Sep 10 14:23 23:55 L
  2845090 beckmn01 *>junkK.419 RUN premium regen2 lc04a10 Sep 9 21:18 Sep 10 14:23 23:55 L
  2845091 beckmn01 *>junkK.420 RUN premium regen2 lc04a10 Sep 9 21:18 Sep 10 14:23 23:55 L
  2845092 beckmn01 *>junkK.421 RUN premium regen2 lc04a10 Sep 9 21:18 Sep 10 14:23 23:55 L
  2845093 beckmn01 *>junkK.422 RUN premium regen2 lc04a10 Sep 9 21:18 Sep 10 14:23 23:55 L
  ```

- **Long format with option -l**
**bmod** - modify submission options of pending jobs

bmod takes similar options to bsub

- bmod -R rusage[mem=20000]  <jobID>
- bmod -q express <jobID>

```
gail01@li03c03: ~ $ bmod -q express 2937044
Parameters of job <2937044> are being changed
```

**bpeek** - display output of the job produced so far

bpeek <jobID>

```
gail01@li03c03: ~ $ bpeek 2937044
<< output from stdout >>>
“Hello Chimera”

<< output from stderr >>>
```
**bkill** - kill jobs in the queue

Lots of ways to get away with murder

```
bkill <job ID>
Kill by job id
bkill 765814
```

```
Kill by job name
bkill -J myjob_1
```

```
Kill a bunch of jobs
bkill -J myjob_ *
```

```
Kill all your jobs
bkill 0
```
bhist - historical information

gail01@li03c03: ~ $ bhist -n 1 -l 2937044

Job <2937044>, Job Name <myfirstjob>, User <gail01>, Project <acc_hpcstaff>, Application <default>, Command <#!/bin/bash;#BSUB -J myfirst job;#BSUB -P acc_hpcstaff ;#BSUB -q premium;#BSUB -n 1;#BSUB -W 6:00 ;#BSUB -R rusage[mem=4000];#BSUB -o %J.stdout ;#BSUB -eo %J.stderr;#BSUB -L /bin/bash ; module load gcc ;which gcc;echo "Hello Chimera">

Tue Sep 10 14:38:25: Submitted from host <li03c03>, to Queue <premium>, CWD <$HOME>, Output File <%J.stdout>, Error File (overwrite) <%J.stderr>, Re-runnable, Requested Resources <rusage[mem=4000 ]>, Login Shell </bin/bash>;

RUNLIMIT
360.0 min of li03c03

MEMLIMIT
3.9 G

Tue Sep 10 14:38:40: Parameters of Job are changed:

Job queue changes to : express;

Tue Sep 10 14:39:36: Dispatched 1 Task(s) on Host(s) <lc02a13>, Allocated 1 Slot(s) on Host(s) <lc02a13>, Effective RES_REQ <select[[(healthy=1)] && (type == local)] order[!-slots:-maxslots] rusage[mem=4000.00] same[model] affinity[core(1)*1] >;

Tue Sep 10 14:39:37: Starting (Pid 399431);

Tue Sep 10 14:39:39: Running with execution home </hpc/users/gail01>, Execution CWD </hpc/users/gail01>, Execution Pid <399431>;

Tue Sep 10 14:39:41: Done successfully. The CPU time used is 1.5 seconds;

Tue Sep 10 14:39:41: Post job process done successfully;

MEMORY USAGE:
MAX MEM: 9 Mbytes; AVG MEM: 2 Mbytes

Summary of time in seconds spent in various states by Tue Sep 10 14:39:41
PEND PSUSP RUN USUSP SSUSP UNKNW TOTAL
71 0 5 0 0 0 76
Common errors of batch jobs

1. Valid allocation account needed in the submission script

   Project acc_project is not valid for user gail01

   Request aborted by esub. Job not submitted.

   * $mybalance (note BODE eligible)

   gail01@li03c03: ~ $ mybalance

   User_ID   Project_name   BODE
   -------   ---------------   ------
   gail01    acc_hpcstaff    Yes
   gail01    acc_DGXTrial    No

2. Reach memory limit

   bhist -n 10 -l 107992756

   Fri Jul 27 11:07:33: Completed <exit>; TERM_MEMLIMIT: job killed after reaching LSF memory usage limit;

   * memory based on one core, with 3000MB as default
   * multithreaded applications need to be on the same node, such as STAR, BWA
Wrapper script: LSFqueue module

- We have installed a wrapper script authored by Harm van Bakel, which will make it easier to interact with the LSF job scheduler on Minerva, per user request.
  
  To load them up, $ml LSFqueue
  
  To get more info on the module, $module help LSFqueue; And a detailed readme file at
  
  /hpc/packages/minerva-centos7/LSFqueue/1.0/README.txt
**Dependent Job**

Any job can be dependent on other LSF jobs.

**Syntax**

```bash
bsub -w 'dependency_expression'
```

usually based on the job states of preceding jobs.

```bash
bsub -J myJ < myjob.lsf
bsub -w 'done(myJ)' < dependent.lsf
```
Self-scheduler

- Submit large numbers of independent serial jobs as a single batch
  - It is mandatory for short batch jobs less than ca. 10 minutes
  - These jobs put heavy load on the LSF server and will be killed

```bash
#!/bin/bash
#BSUB -q express
#BSUB -W 00:20
#BSUB -n 12
#BSUB -J selfsched
#BSUB -o test01
module load selfsched # load the selfsched module
mpirun -np 12 selfsched < test.inp # 12 cores, with one master process

$PrepINP < templ.txt > test.inp (InputForSelfScheduler)
$cat templ.txt
1 10000 2 F ← start, end, stride, fixed field length?
/my/bin/path/Exec_# < my_input_parameters_# > output_.log

$cat test.inp (a series of job command)
/my/bin/path/Exec_1 < my_input_parameters_1 > output_1.log
/my/bin/path/Exec_3 < my_input_parameters_3 > output_3.log
...
/my/bin/path/Exec_9999 < my_input_parameters_9999 > output_9999.log
```
Job submission script example: selfsched.lsf

```bash
#!/bin/bash

# Job name
#BSUB -J myMPIjob

# allocation account
#BSUB -P acc_bsr3101

# queue
#BSUB -q express

# number of compute cores
#BSUB -n 64

# 4 cores per node
#BSUB -R span[ptile=4]

# 256 GB of memory (4 GB per core)
#BSUB -R rusage[mem=4000]

# walltime (30 min.)
#BSUB -W 00:20

# output log (%J : JobID)
#BSUB -o %J.stdout

# error log
#BSUB -eo %J.stderr

# Initialize the execution environment
#BSUB -L /bin/bash

echo "Job ID" : $LSB_JOBID

module load python
module load selfsched

mpirun -np 64 selfsched < JobMixPrep.inp > JonMixPrep.out
```
Parallel Job

- **Array job**: Parallel analysis for multiple instances of the same program
  - Execute on multiple data files simultaneously
  - Each instance running independently

- **Distributed memory program**: Message passing between processes (e.g. MPI)
  - Processes execute across multiple CPU cores or nodes

- **Shared memory program** (SMP): multi-threaded execution (e.g. OpenMP)
  - Running across multiple CPU cores on same node

- **GPU programs**: offloading to the device via CUDA
Array Job

- Groups of jobs with the same executable and resource requirements, but different input files.
  - -J “Jobname[index | start-end:increment]”
  - Range of job index is 1~10,000

```bash
#!/bin/bash
#BSUB -P acc_hpcstaff
#BSUB -n 1
#BSUB -W 02:00
#BSUB -q express
#BSUB -J "jobarraytest[1-10]"
#BSUB -o logs/out.%J.%I
#BSUB -e logs/err.%J.%I
Working on file.$LSB_JOBINDEX
```
MPI Jobs

- This example requests 48 cores and 2 hours in the "express" queue.
  - Those 48 cores can be dispatched across multiple nodes

```bash
#!/bin/bash
#BSUB -J myjobMPI
#BSUB -P acc_hpcstaff
#BSUB -q express
#BSUB -n 48
#BSUB -W 02:00
#BSUB -o %J.stdout
#BSUB -e %J.stderr
#BSUB -L /bin/bash

cd $LS_SUBCWD

module load openmpi

mpirun -np 48 /my/bin/executable < my_data.in
```
Multithreaded Jobs - OpenMP

- Multiple CPU cores within one node using shared memory
  - In general, a multithreaded application uses a single process which then spawns multiple threads of execution
  - It’s highly recommended the number of threads is set to the number of compute cores

- Your program needs to be written to use multi-threading

```bash
#!/bin/bash
#BSUB -J myjob
#BSUB -P YourAllocationAccount
#BSUB -q express
#BSUB -n 4
#BSUB -R "span[hosts=1]"
#BSUB -R rusage[mem=12000]
#BSUB -W 01:00
#BSUB -o %J.stdout
#BSUB -eo %J.stderr
#BSUB -L /bin/bash

cd $LS_SUBCWD
export OMP_NUM_THREADS=4 #sets the number of threads
/my/bin/executable < my_data.in
```
Job submission script example: star.lsf

```bash
#!/bin/bash
#BSUB -J mySTARjob                      # Job name
#BSUB -P acc_PLK2                       # allocation account
#BSUB -q premium                        # queue
#BSUB -n 8                               # number of compute cores
#BSUB -W 12:00                           # walltime in HH:MM
#BSUB -R rusage[mem=4000]               # 32 GB of memory (4 GB per core)
#BSUB -R span[hosts=1]                  # all cores from one node
#BSUB -o %J.stdout                      # output log (%J : JobID)
#BSUB -eo %J.stderr                     # error log
#BSUB -L /bin/bash                      # Initialize the execution environment

module load star
WRKDIR=/sc/orga/projects/hpcstaff/benchmark_star
STAR --genomeDir $WRKDIR/star-genome --readFilesIn Experiment1.fastq --runThreadN 8 --outFileNamePrefix Experiment1Star
```

Submit the script with the `bsub` command:

`bsub < star.lsf`
Specifying a resource - OpenMP job

Span: define the shape of the slots you ask for:

- `-n 12 -R span[hosts=1]` - allocate all 12 cores to one host
- `-n 12 -R span[ptile=12]` - all 12 slots/cores must be on 1 node
- `-n 24 -R span[ptile=12]` - allocate 12 cores per node = 2 nodes

OMP_NUM_THREADS must be set in script:

- `bsub -n 12 -R span[hosts=1] < my_parallel_job`
  
  `export OMP_NUM_THREADS=12`
- `bsub -n 12 -R span[ptile=12] -a openmp < my_parallel_job`
  
  LSF sets it for you as number of procs per node
- `bsub -n 1 -R "affinity[core(12)]" -R "rusage[mem=12000]" -a openmp < my_parallel_job`
  
  ○ 1 job slot with 12 cores, 12000MB memory to that job slot...not per core
  ○ Advantage: Can vary number of cores and/or memory without making any other changes or calculations
A Bravura Submission - Mixing it all together

Suppose you want to run a combined MPI-openMP job. One mpi process per node, openMP in each MPI Rank:

```
bsub -n 20 -R span[ptile=1] -R affinity[core(8)] -a openmp < my_awsome_job
```

- ptile=1 - one slot on each node
- core(8) - 8 cores per job slot
- openmp - will set OMP_NUM_THREADS on each node to 8
GPGPU (General Purpose Graphics Processor Unit)

- GPGPU resources on Minerva
  - Interactive queue (1 GPU node)
  - gpu queue for batch (11 GPU nodes)
    - Can be quite busy sometimes

<table>
<thead>
<tr>
<th>number of nodes</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU card</td>
<td>4 v100</td>
</tr>
<tr>
<td>CPU cores</td>
<td>32</td>
</tr>
<tr>
<td>host memory</td>
<td>384GB</td>
</tr>
<tr>
<td>GPU memory</td>
<td>16 GB</td>
</tr>
</tbody>
</table>

#BSUB -q gpu
#BSUB -n Ncpu
#BSUB -R v100
#BSUB -R "gpu rusage[ngpus_excl_p=2]"
module purge
module load anaconda3 ( or 2)
module load cuda
source activate tfGPU
python -c "import tensorflow as tf"

# submit to gpu queue
# Ncpu is 1~32 on v100
# request specified gpu node v100
# The number of GPUs requested per node ( 1 by default)
# to access tensorflow
# to access the drivers and supporting subroutines
GPGPU (continue)

- Request multiple GPU cards across different GPU nodes

```bash
#BSUB -q gpu
#BSUB -n 8
#BSUB -R span[ptile=2]
#BSUB -R v100
#BSUB -R "gpu rusage[ngpus_excl_p=2]"
# submit to gpu queue
# 8 compute cores requested
# 2 cores per node, so 4 nodes in total requested
# request specified gpu node v100
# 2 GPUs requested per node
```
Checkpoint/Restart

- The long-time standard BLCR method is no longer supported
- We are investigating a more modern method: Checkpoint/Restart In User space (CRIU)
- In final stages of testing. Watch for the Announcement.
LSF: summary on job submission examples

Interactive session:

# interactive session for 1 core
$ bsub -P acc_hpcstaff -q interactive -n 1 -R rusage[mem=4000] -W 00:10 -ls /bin/bash

# interactive session for multiple cores on the same node, default memory 3GB/core
$ bsub -P acc_hpcstaff -q interactive -n 12 -R "span[hosts=1]" -W 00:10 -ls /bin/bash

# interactive session for multiple cores on different node node
$ bsub -P acc_hpcstaff -q interactive -n 50 -W 00:10 -ls /bin/bash

# interactive GPU nodes, flag “-R v100” is required
$ bsub -P acc_hpcstaff -q interactive -n 1 -R v100 -R rusage[ngpus_excl_p=1] -W 01:00 -ls /bin/bash

Batch jobs submission:

# simple standard job submission
$ bsub -P acc_hpcstaff -q express -n 1 -W 00:10 echo “Hello World”

# GPU job submission if you don’t mind the GPU card model
$ bsub -P acc_hpcstaff -q gpu -n 1 -R rusage[ngpus_excl_p=2] -W 00:10 echo “Hello World”

# flag “-R v100” is required if you want to use certain GPU card (v100/p100)
$ bsub -P acc_hpcstaff -q gpu -n 1 -R v100 -R rusage[ngpus_excl_p=1] -W 00:10 echo “Hello World”

# himem job submission, flag “-R himem” is required
$ bsub -P acc_hpcstaff -q premium -n 1 -R himem -W 00:10 echo “Hello World”
More bsub [options]

Tips for efficient usage of the queuing system

- **User limitation**
  - Max pending job per user: 20,000
  - Heavy users: depending on the resource requested

- **Find appropriate queue and nodes**
  - Use `-q interactive` for debug (both CPU and GPU with internet access)
  - Use `-q express` if `walltime < 12h`
  - Use himem node for memory intensive job

- **Request reasonable resource**
  - Prior knowledge needed (run test program and use top or others to monitor)
  - Keep it simple

- **Job not start after a long pending time**
  - Whether the resource requested is non-exist: `-R rusage[mem = 10000] -n 20`
  - Run into PM: `NOTE: Because of PM reservations, job may not run until after Sat 21 Mar at 8:00PM`

- **If you see memory not enough**
  - Think about shared memory vs distributed memory job………
  - Use `-R span[hosts=1]` where needed
Final Friendly Reminder

- Never run jobs on login nodes
  - For file management, coding, compilation, etc., purposes only
- Never run jobs outside LSF
  - Fair sharing
  - Scratch disk not backed up, efficient use of limited resources
  - Job temporary dir configured to /local/JOBS instead of /tmp.
- Logging onto compute nodes is no longer allowed

- Follow us by visiting https://labs.icahn.mssm.edu/minervalab, weekly update and twitter
- Acknowledge Scientific Computing at Mount Sinai should appear in your publications
  - This work was supported in part through the computational resources and staff expertise provided by Scientific Computing at the Icahn School of Medicine at Mount Sinai.
  - If you are using BODE: “Research reported in this paper was supported by the Office of Research Infrastructure of the National Institutes of Health under award numbers S10OD026880. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.
Last but not Least

▶ Got a problem? Need a program installed? Send an email to:

hpchelp@hpc.mssm.edu