Load Sharing Facility (LSF)

Minerva Scientific Computing Environment

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

Patricia Kovatch
Lili Gai, PhD

Eugene Fluder, PhD
Hyung Min Cho, PhD
Jielin Wu, PhD
Wei Guo, PhD
Kali Mclennan

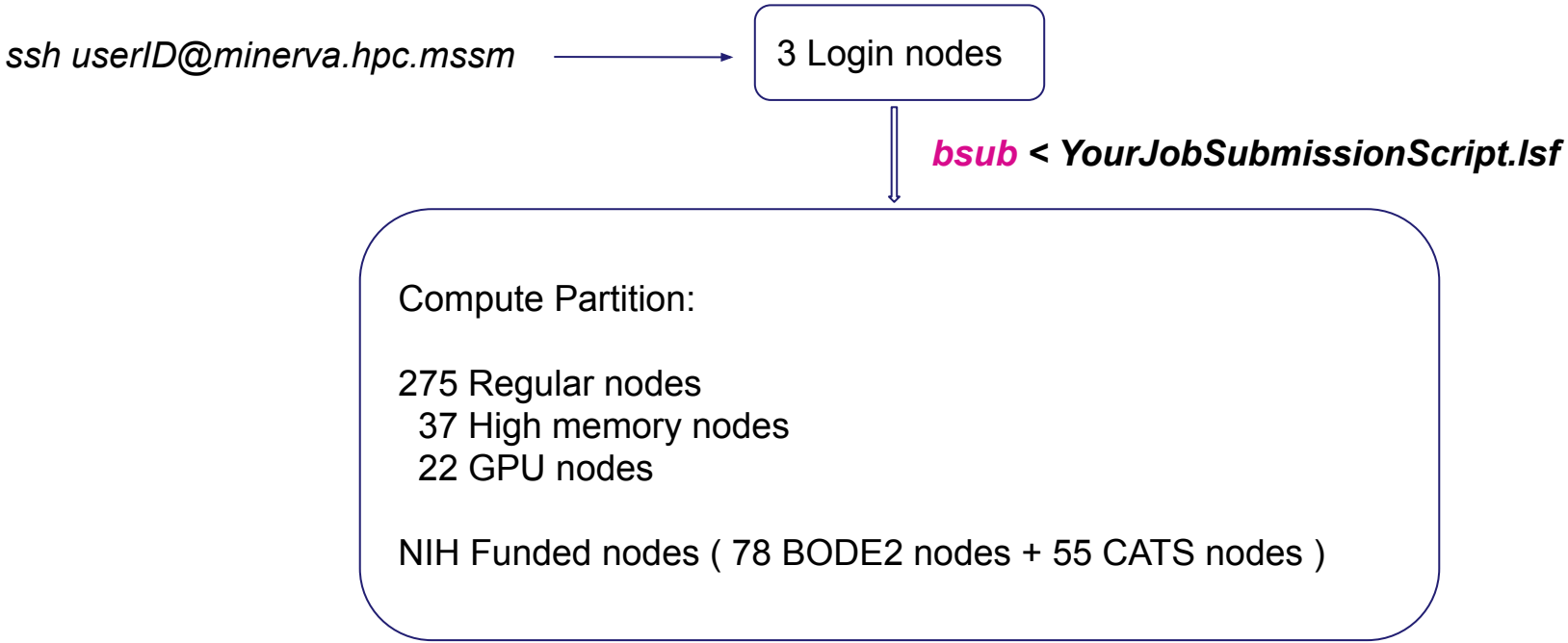
March 29, 2023

Icahn
School of
Medicine at
Mount
Sinai

Outline

- LSF introduction and basic/helpful LSF commands
- Job submission and monitoring
- Interactive job
- Dependent job
- Parallel jobs: parallel processing and GPUs
- Job arrays and Self-scheduler
- Job checkpoint/restart
- Tips for efficient usage of the queuing system

Running Jobs on Minerva Compute Nodes



Access to compute resources and job scheduling are managed by IBM Spectrum LSF (Load Sharing Facility) batch system.

Prerequisite

- Must have a project allocation account.
- If you don't have one, ask your PI (or project authorizer) send a request at hpchelp@hpc.mssm.edu
- To see a list of accessible project accounts:

$ **mybalance**

User_ID	Project_name	BODE/CATS
choh07	acc_hpcstaff	Yes
choh07	acc_DGXTrial	No

Basic LSF commands

- **bsub** Batch job submission

- **bjobs** Show your job status. Pending reasons
- **bkill** Kill a batch job
- **bmod** Modify the resource requirement of a **pending** job

- **bpeek** Display the stdout and stderr output of an unfinished job
- **bhist** Display historical information about a job
- **bqueues** Display information about queues
- **bhosts** Display load status information of each compute node

IBM LSF Documentation: https://www.ibm.com/docs/en/spectrum-lsf/10.1.0

Batch job submission example

$ cat myfirst.lsf

```
#!/bin/bash
#BSUB -J myfirstjob                    # Job name
#BSUB -P acc_hpcstaff                  # REQUIRED; To get allocation account, type "mybalance"
#BSUB -q premium                       # queue; default queue is premium
#BSUB -n 1                             # number of compute cores (job slots) needed, 1 by default
#BSUB -W 6:00                          # REQUIRED; walltime in HH:MM
#BSUB -R rusage[mem=4000]              # 4000 MB of memory request per "-n"; 3000 MB by default
#BSUB -oo %J.stdout                    # output log (%J : JobID)
#BSUB -eo %J.stderr                    # error log
#BSUB -L /bin/bash                     # Initialize the execution environment

ml gcc                                 # Commands that you need to run
cd /sc/arion/work/MyID/my/job/dir/
../mybin/serial_executable < testdata.inp > results.log
```

$ bsub < myfirst.lsf

Job <87426883> is submitted to queue <premium>.

Batch job submission example (continue)

$ cat mysecond.lsf

```
#!/bin/bash
#BSUB -q premium                    # queue
#BSUB -R rusage[mem=4000]           # 4000 MB of memory request per "-n"; 3000 MB by default
#BSUB -oo %J.stdout                 # output log (%J : JobID)
#BSUB -eo %J.stderr                 # error log
#BSUB -L /bin/bash                  # Initialize the execution environment
ml gcc                              # Commands that you need to run
cd /sc/arion/work/MyID/my/job/dir/
../mybin/serial_executable < testdata.inp > results.log
```

$ bsub -q **express** -J mysecondjob -P acc_hpcstaff -n 1 -W 30 **< mysecond.lsf**

Job <87426921> is submitted to queue <premium>.

If an option is given on both the bsub command line and in the job script, the command line option overrides the option in the script.

bsub major options

-P accountName	of the form: **acc_**projectName
-q queuename	submission queue
-n ncpu	number of cpu's requested (default: 1)
-W wallClockTime	in form of HH:MM
-R rusage[mem=...]	amount of memory requested **per "-n"** in *MB* *Standard abbreviations (MB, GB, …) can also be used.* max memory per node: ~163GiB (Chimera, BODE compute), ~325GB (GPU), ~1.4TiB (himem, CATS), ~1.9TB (himem-GPU-A100-80GB)
-R span[#-n's per physical node]	span[ptile=4] - 4 cores per node/host span[**hosts=1**] - all cores on **same** node/host
-R himem	Request high memory node

bsub major options

- -o Name of output file (concatenated)
- -oo Name of output file (overwrite)
- -e Name of error file (concatenated)
- -eo Name of error file (overwrite)

NOTE: Default output is mailed to the user BUT since we have disabled mail response, it goes into the bit bucket.

If -o(o) is specified but not -e, error is appended to output file.

Minerva LSF queue structure

Queue structure in Minerva		
Queue	**Wall time limit**	**available resources**
interactive (Dedicated to interactive jobs)	12 hours	4 nodes+2 V100 GPU nodes
premium	6 days	275 nodes + 37 himem nodes+BODE2+CATS
express	12 hours	275 nodes + 4 dedicated nodes (may change)+BODE2+CATS
long	2 weeks	6 dedicated (288 cores) + 12 BODE2
gpu	6 days	40 V100 24 A100 8 A100-80GB
gpuexpress	15 hours	8 A100
private	unlimited	private nodes

bjobs : status of jobs

Check your job: $ **bjobs** *JobID*

JOBID	USER	JOB_NAME	STAT	QUEUE	FROM_HOST	EXEC_HOST	SUBMIT_TIME	START_TIME	TIME_LEFT
87426883	choh07	myfirstjob	PEND	premium	li03c03	-	Mar 27 14:38	-	-

Pending reasons: $ **bjobs** -p *JobID*

JOBID	USER	JOB_NAME	STAT	QUEUE	FROM_HOST	EXEC_HOST	SUBMIT_TIME	START_TIME	TIME_LEFT
87426883	choh07	myfirstjob	PEND	premium	li03c03	-	Mar 27 14:38	-	-

New job is waiting for scheduling;

Show full details about the job: **bjobs** -ℓ *JobID*

bkill : terminate jobs in the queue

Lots of ways to get away with murder

Kill by JobID	**bkill** 87426883
Kill by JobName	**bkill** -J myjob_1
Kill a bunch of jobs	**bkill** -J myjob_*
Kill all your jobs	**bkill** 0

bpeek: display output of the job produced so far

$ **bpeek** 2937044

<< output from stdout >>

“Hello Minerva”

<< output from stderr >>

bmod: modify submission options of “pending” jobs

bmod takes similar options to **bsub**

- **bmod** -R rusage[mem=20000] *JobID*
  - -R replaces **ALL** R fields not just the one you specify
- **bmod** -q express *JobID*

  $ **bmod** -q express 2937044

  Parameters of job <2937044> are being changed

bhist : historical information

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

Job <2937044>, Job Name <myfirstjob>, User <gail01>, Project <acc_hpcstaff>, Ap
                     plication <default>, Command <#!/bin/bash;#BSUB -J myfirst
                     job;#BSUB -P acc_hpcstaff  ;#BSUB -q premium;#BSUB -n 1;#B
                     SUB -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 <$H
                     OME>, 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 Slo
                     t(s) on Host(s) <lc02a13>, Effective RES_REQ <select[((hea
                     lthy=1)) && (type == local)] order[!-slots:-maxslots] rusa
                     ge[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    UNKWN    TOTAL
  71       0        5        0        0        0        76
```

bhosts : Displays nodes and their load status

- List all the compute nodes on Minerva

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
filizm02-3	ok	-	24	1	1	0	0	0
lc01a05	closed	-	48	48	48	0	0	0
lc01a07	closed	-	48	48	16	0	0	32
lc04a19	unavail	-	48	0	0	0	0	0
lg03a01	ok	-	32	0	0	0	0	0
lg03a02	ok	-	32	17	17	0	0	0
lh03c03	closed	-	48	48	48	0	0	0

.
.
.

bhosts: himem, gpu, bode, nonbode (major nodes), interactive

```
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
gail01@li03c03: ~ $ bhosts gpu
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
lg03a02            ok              -     32      0      0      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
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, cats, nonbode (major nodes), interactive

```
gail01@li03c03: ~ $ bhosts nonbode | head
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
lc01a05            closed          -     48     48     16      0      0     32
lc01a06            closed          -     48     48     18      0      0     30
lc01a07            closed          -     48     48     16      0      0     32
lc01a08            closed          -     48     48     16      0      0     32
lc01a09            closed          -     48     48     30      0      0     18
lc01a10            closed          -     48     48     12      0      0     36
lc01a11            closed          -     48     48     12      0      0     36
lc01a12            closed          -     48     48     14      0      0     34
lc01a13            closed          -     48     45     13      0      0     32
```

```
gail01@li03c03: ~ $ bhosts interactive
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
lc02a27            ok              -     48      1      1      0      0      0
lc02a28            ok              -     48     15     15      0      0      0
lc02a29            ok              -     48      2      1      0      0      1
lc02a30            ok              -     48      1      1      0      0      0
lg03a01            ok              -     32      0      0      0      0      0
lg03a02            ok              -     32      1      1      0      0      0
```

bqueues : information about all the available queues

```
[choh07@li03c03 ~]$ bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
sla2            210  Open:Active       -    -    -    -     0     0     0     0
premium         200  Open:Active       -    -    -    -  4612   606  3776     0
admintest       200  Open:Active       -    -    -    -     0     0     0     0
sla             200  Open:Active       -    -    -    -     0     0     0     0
gputest         130  Open:Active       -    -    -    -    20    12     6     0
gngpu           130  Open:Active       -    -    -    -     1     1     0     0
private         130  Open:Active       -    -    -    -   235    94   141     0
cact            130  Open:Active       -    -    -    -     0     0     0     0
express         120  Open:Active       -    -    -    -   163   121    42     0
interactive     100  Open:Active       -    -    -    -   114     0   114     0
long            100  Open:Active       -    -    -    -  2898  2211   641     0
gpu             100  Open:Active       -    -    -    -   278    82   196     0
gpuexpress      100  Open:Active       -    -    -    -    19     0    19     0
```

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.

- Use ***mybalance*** to see accessible accounts (note BODE/CATS eligible)

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,...

3. No suitable hosts for the job

- Requested resource is non-exist : -n 128 -R span[hosts=1]

Interactive access to compute resources

- 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 -P acc_hpcstaff -q interactive -n 4 -W 2:00 -R rusage[mem=4000] -R span[hosts=1] -XF -Is /bin/bash
```

- -Is: Interactive terminal/shell
- -XF: X11 forwarding
- /bin/bash : the shell to use

```
$ bsub -P acc_hpcstaff -q interactive -n 4 -W 2:00 -R rusage[mem=4000] -R span[hosts=1]
-XF -Is /bin/bash
Job <2916837> is submitted to queue <interactive>.
<<ssh X11 forwarding job>>
<<Waiting for dispatch ...>>
<<Starting on lc02a29>>
```

Dependent Job

Any job can be dependent on other LSF jobs.

Syntax
bsub -w 'dependency_expression'
usually based on the job states of preceding jobs.

bsub -J myJ < myjob.lsf

bsub -w '**done**(myJ)' < dependent.lsf

For more details about the dependency_expression:

https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=scheduling-dependency-conditions

Parallel Jobs

- **Distributed memory program**: Message passing between processes (e.g. MPI) Map-reduce(e.g. Spark)
  - 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**: Parallel analysis for multiple instances of the same program
  - Execute on multiple data files simultaneously
  - Each instance running independently

Message Passing Interface (MPI) Jobs

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

```
#!/bin/bash
#BSUB -J myjobMPI
#BSUB -P acc_hpcstaff
#BSUB -q express
#BSUB -n 48
#BSUB -R span[ptile=8]

#BSUB -W 02:00
#BSUB -o %J.stdout
#BSUB -eo %J.stderr
#BSUB -L /bin/bash

cd $LS_SUBCWD
module load openmpi
mpirun -np 48 /my/bin/executable < my_data.in
```

Apache Spark Jobs

- Use **lsf-spark-submit.sh** to launch job. See https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=lsf-apache-spark for full details

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

ml spark

lsf-spark-submit.sh --class "SimpleApp" target/scala-2.10/simple-project_2.10-1.0.jar ../myfile.txt
```

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 has to be written to use multi-threading

```
#!/bin/bash
#BSUB -J myjob
#BSUB -P YourAllocationAccount
#BSUB -q express
#BSUB -n 4
#BSUB -R "span[hosts=1]"
#BSUB -R rusage[mem=12GB]
#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
```

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 (2 GPU node)
  - gpu queue for batch (20 GPU nodes)

- GPU option specification:

-gpu num=<Ngpus>:**gmodel**=<model>

	V100	A100	A100-80GB
# of nodes	10	8	2
GPU card	4 V100	4 A100	4 A100
CPU cores	32	48	64
host memory	384GB	384GB	2TB
GPU memory	16 GB	40GB	80GB

V100: **-gpu num**=Ngpus:**gmodel**=*TeslaV100_PCIE_16GB*
(or **-gpu num**=Ngpus **-R** v100)

A100: **-gpu** num=Ngpus:**gmodel**=*NVIDIAA100_PCIE_40GB*
(or **-gpu num**=Ngpus **-R** a100)

A100-80G: **-gpu num**=Ngpus:**gmodel**=*NVIDIAA100_SXM4_80GB*
(or **-gpu num**=Ngpus **-R** a10080g)

GPGPU (continue)

```
#BSUB -q gpu                                          # submit to gpu queue
#BSUB -n Ncpu                                         # Ncpu is 1~32 on v100

#BSUB -gpu num=4:gmodel=NVIDIAA100_PCIE_40GB          # request 4 GPUs on A100 node
#BSUB -R span[hosts=1]                                # request all gpu card on the same node
                                                      # The number of GPUs requested per node

module purge
module load anaconda3 ( or 2)                         # to access tensorflow
module load cuda                                      # to access the drivers and supporting
source activate tfGPU                                 subroutines

python -c "import tensorflow as tf"
```

GPGPU (continue)

- LSF will set CUDA_VISIBLE_DEVICES to the list of GPU cards assigned to the job.

  E.g: 2,1,3 Most standard packages honor these assignments

  - DO NOT MANUALLY CHANGE THE VALUE OF CUDA_VISIBLE_DEVICES.
- Multiple GPU cards can be requested across different GPU nodes

```
#BSUB -q gpu                    # submit to gpu queue
#BSUB -n 8                      # 8 compute cores requested
#BSUB -R span[ptile=2]          # 2 cores per node, so 4 nodes in total requested
#BSUB -R v100                   # request specified gpu node v100, change to a100 or a10080g
#BSUB -gpu num=2                # 2 GPUs requested per node
```

Note that 2 GPU cards will be reserved on each of 4 nodes for your job. If your job cannot /does not run in distributed mode, you will still lock these resources on the nodes that you are not using and prevent others from being dispatched to those node.

CUDA_VISIBLE_DEVICES may be defined differently on each of the nodes allocated to your job.

GPGPU - Local SSD

A100	1.8 TB SATA SSD
A100-80GB	7.0 TB NVMe PCIe SSD

- Make your own directory under /ssd and direct your temporary files there.
- Clean up your temporary files after completion.

```
#BSUB -q gpu
#BSUB -gpu num=2:gmodel=NVIDIAA100_SXM4_80GB
#BSUB -R span[hosts=1]
#BSUB -E "mkdir /ssd/YourID_$LSB_JOBID"
#BSUB -Ep "rm -rf /ssd/YourID_$LSB_JOBID"
#BSUB …
```

Array Job

- Groups of jobs with the same executable and resource requirements, but different input files that can be indexed by numbers.
  - -J “Jobname[index | start-end:increment]”
  - Range of job index is **1**~ 10,000
  - **LSB_JOBINDEX** is set to array index

```
#!/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
echo “Working on file.$LSB_JOBINDEX”
```

Array Job (continue)

$ ***bsub < myarrayjob.sh***
Job <2946012> is submitted to queue <express>.

$ ***bjobs***

JOBID	USER	JOB_NAME	STAT	QUEUE	FROM_HOST	EXEC_HOST	SUBMIT_TIME	START_TIME	TIME_LEFT
2946012	gail01	*rraytest[1]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[2]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[3]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[4]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[5]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[6]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[7]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[8]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*rraytest[9]	PEND	express	li03c03	-	Sep 10 14:50	-	-
2946012	gail01	*raytest[10]	PEND	express	li03c03	-	Sep 10 14:50	-	-

Self-scheduler

- Submit large numbers of independent short **serial** jobs as a single batch

```
#!/bin/bash
#BSUB -q express
#BSUB -W 1:00
#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
```

$cat test.inp (test.inp: input for Self-Scheduler; a series of job commands)

```
/my/bin/path/executable < input_jason > output_jason
/my/bin/path/executable < input_tom > output_tom
. . .
/my/bin/path/executable < input_jane > output_jane
```

Job submission script example: selfsched.lsf

```
#!/bin/bash
#BSUB -J myMPIjob                    # Job name
#BSUB -P acc_bsr3101                 # allocation account
#BSUB -q express                     # queue
#BSUB -n 64                          # number of compute cores
#BSUB -R span[ptile=4]               # 4 cores per node
#BSUB -R rusage[mem=4G]        # 256 GB of memory (4 GB per core)
#BSUB -W 2:00                        # walltime (2 hours.)
#BSUB -o %J.stdout                   # output log (%J : JobID)
#BSUB -eo %J.stderr                  # error log
#BSUB -L /bin/bash                   # Initialize the execution environment

echo "Job ID                 : $LSB_JOBID"
echo "Job Execution Host     : $LSB_HOSTS"
echo "Job Sub. Directory     : $LS_SUBCWD"

module load python
module load selfsched
mpirun -np 64 selfsched  <  BunchOfSerialJobs.inp  > BunchOfSerialJobs.out
```

Checkpoint/Restart

- Checkpoint: Save the state of a process at a particular point in the computation
- Restart: Restore the state of a process and continue the computation from the saved state.

Checkpoint/Restart

- The long-time standard BLCR method is no longer supported
- It has been replaced by the more modern method: Checkpoint/Restart In User space (CRIU)

bsub -k "checkpoint_dir [init=initial_checkpoint_period] [check-point_period] [method=method_name]"

E.g.,

bsub -k "chkpntDir init=10 90 method=criu"

More details at
https://labs.icahn.mssm.edu/minervalab/documentation/job-checkpoint/

Checkpoint/Restart

- To restart, use brestart command
- Must restart on same type of machine.
- Can increase memory, change queue, add dependency, etc (see man page)

brestart [options] checkpointFolder jobid

brestart -W 4:00 -R rusage[mem=26000] chkpnt 193876

BONUS: You may be able to checkpoint a process even if you didn't set it up via LSF.

See HPC web site for details.

Tips for efficient usage of the queuing system

- User limitation
  - Max running jobs per user: 4,000
  - Max pending jobs per user: 20,000
  - Max num. of GPUs per user: 10
  - 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 jobs
- 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 = 100GB] -n 20 -R span[hosts=1]
  - Run into PM:

```
NOTE: Because of PM reservations, job may not run
      until after Sat 21 Mar at  8:00PM
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Job <6628109> is submitted to queue <premium>.
```

- 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
- 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

Acknowledgements

- Supported by the Clinical and Translational Science Awards (CTSA) grant UL1TR004419 from the National Center for Advancing Translational Sciences, National Institutes of Health.