

Load Sharing Facility (LSF)

Minerva Scientific Computing Environment

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

Hyung Min Cho, PhD
The Minerva HPC Team

September 24, 2026



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**
- **Tips for efficient usage of the queuing system**

Minerva cluster @ Mount Sinai



Chimera Partition:

- 4x **login nodes** - Intel Emerald Rapids 8568Y+, 2.3GHz – 96 cores with 512 GB memory per node.
- Compute nodes -
 - 146 **high memory nodes** - Intel Emerald Rapids 8568Y+, 2.3GHz - 96 cores with **1.5 TB** memory per node.)
 - 55 **high memory nodes** - Intel 8358, 2.6 GHz- 64 cores - **1.5 TB** mem/ node
 - 37 **high memory nodes** - Intel 8168/8268, 2.7/2.9GHz - **1.5 TB** mem/node
 - **GPU nodes:**
 - 12 -Intel 6142, 2.6GHz - 384 GB memory - 4x V100-**16 GB** GPU
 - 8 - Intel 8268, 2.9 GHz - 384 GB memory - 4x A100- **40 GB** GPU
 - 2 - Intel 8358,2.6GHz - 2 TB memory - 4x A100- **80 GB** GPU
 - 2 - Intel 8358 2.6 GHz- 500 GB memory - 4x H100- **80 GB** GPU
 - 47 - Intel ER 8568Y+, 2.3GHz - 1.5 TB memory - 4x H100- **80 GB** GPU
 - 4 - AMD Genoa 9334 2.7GHz - 1.5 TB memory - 8x L40S- **48 GB** GPU



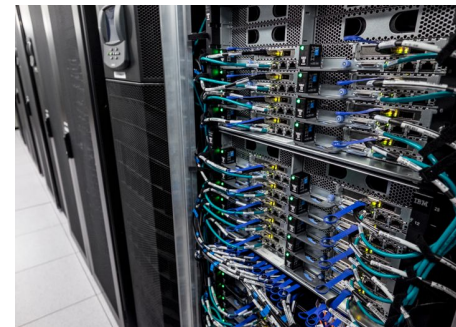
NIH FUNDED NODES

AIMS (AI Mount Sinai) Partition:

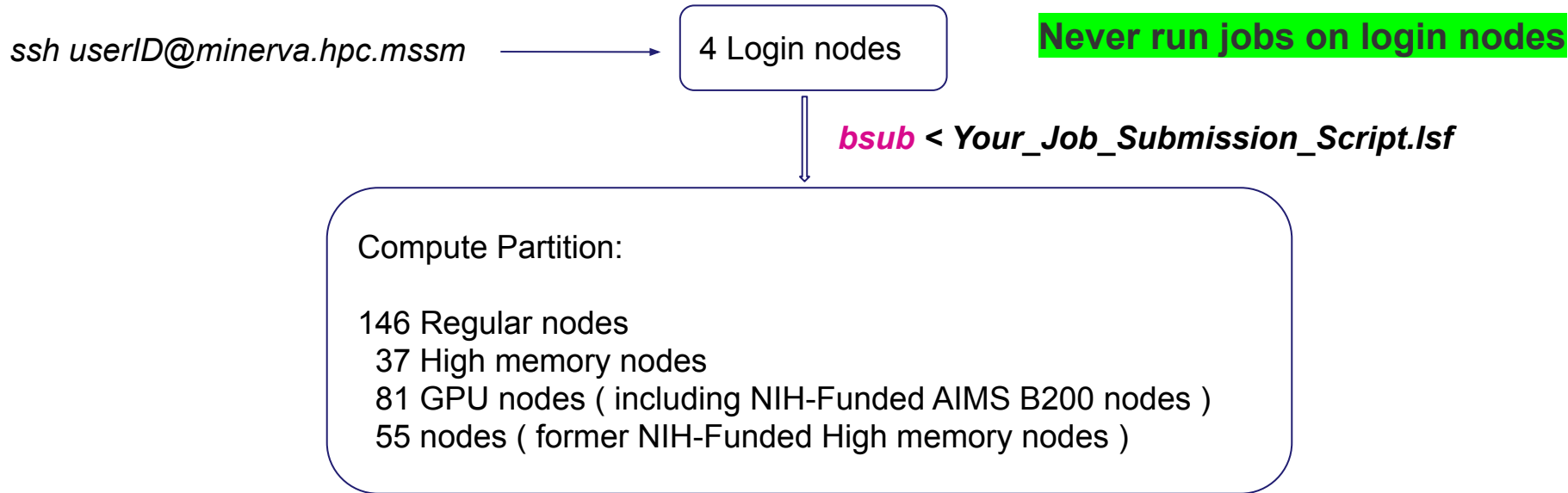
- \$2M AIMS awarded by NIH (Kovatch PI).
- 6 - Intel Xeon Platinum 8570, 2.1 GHz - 112 cores per node - **2 TB** / node
- 8x NVLinked B200 GPUs per node – 192 GB memory per GPU

Storage: 32 PB of high-speed online storage as an IBM General Parallel File System (GPFS)

- **Path /sc/arion** : Use the system path environment variable in scripts **\$GPFS**



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	AIMS
-----	-----	---
choh07	acc_bsr2402	No
choh07	acc_KPMP	No
choh07	acc_TSM_TEMP	No
choh07	acc_hpcstaff	Yes
choh07	acc_DGXTrial	No

Basic LSF commands

- **bsub** Batch job submission
- **bjobs** Show your job status. Pending reasons
- **kill** 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
```

```
#BSUB -P acc_hpcstaff
```

```
#BSUB -q premium
```

```
#BSUB -n 1
```

```
#BSUB -W 6:00
```

```
#BSUB -R rusage[mem=4000]
```

```
#BSUB -oo %J.stdout
```

```
#BSUB -eo %J.stderr
```

```
#BSUB -L /bin/bash
```

Job name

*# **REQUIRED**; To get allocation account, type “mybalance”*

queue; default queue is premium

number of compute cores (job slots) needed, 1 by default

*# **REQUIRED**; walltime in HH:MM*

*# 4000 **MB** of memory request per “-n”; 3000 MB by default*

output log (%J : JobID)

error log

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: ~1.4TB (Chimera,GPU H100, L40S),
~325GB (GPU V100, A100) , ~1.9TB (himem-GPU A100-80GB, B200),
~435GB(GPU H100-80GB)

- R span[#-n's per physical node]
 - span[ptile=4] - 4 cores per node/host
 - span[**hosts=1**] - all cores on **same** node/host

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	Description	Max Walltime	Minerva	Minerva- Res
premium	Normal submission queue	144 hrs	✓	✓
express	Rapid turnaround jobs	12 hrs	✓	✓
interactive	Jobs running in interactive mode	12 hrs	✓	—
long	Jobs requiring extended runtime	336 hrs	✓	—
gpu	Jobs requiring GPU resources	144 hrs	✓	—
gpuexpress	Short jobs requiring GPU resources	15 hrs	✓	—
ondemand	Jobs dedicated to Open OnDemand	12 hrs	✓	—
private	Jobs using dedicated resources	Unlimited	✓	—
others	Any other queues are for testing by the Scientific Computing group	N/A	✓	—

Minerva-Res note: only `express` and `premium` are available on Minerva-Res. All other queues in the table above are Minerva-only.

bqueues : information about all the available queues

```
[choh07@li04e02 ~]$ bqueues
```

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
ollama	260	Open:Active	-	-	-	-	8	0	8	0
gpuexpress	240	Open:Active	-	-	-	-	55022	54459	563	0
gpu	230	Open:Active	-	-	-	-	2275	1024	1251	0
premium	200	Open:Active	-	-	-	-	1021390	1013130	5482	
ondemand	200	Open:Active	-	-	-	-	70	0	70	0
express	200	Open:Active	-	-	-	32	1122	31	1091	0
private	200	Open:Active	-	-	-	-	458	0	458	0
sla	200	Open:Active	-	-	-	-	128	0	128	0
interactive	100	Open:Active	-	48	-	-	202	33	166	0
ondemand-network	100	Open:Active	-	-	-	-	4	3	1	0
long	100	Open:Active	-	-	-	-	188	0	188	0

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
lc03e16	ok	-	96	21	21	0	0	0
lc03e17	ok	-	96	10	10	0	0	0
lc04g07	ok	-	96	48	48	0	0	0
lc06e01	ok	-	96	94	94	0	0	0
lc06e02	closed	-	96	96	82	0	0	14
lc06e03	ok	-	96	82	82	0	0	0
lc06e04	closed	-	96	96	96	0	0	0
lc06e05	ok	-	96	79	79	0	0	0
lc06e06	closed	-	96	96	96	0	0	0
lc06e07	closed	-	96	96	96	0	0	0
lc06e08	ok	-	96	92	92	0	0	0
lc06e09	ok	-	96	78	78	0	0	0

bhosts (continue)

```
[choh07@li04e02 ~]$ bhosts gpuexpress
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
lg03a03	ok	-	32	6	6	0	0	0
lg03a04	ok	-	32	5	5	0	0	0
lg03a05	ok	-	32	0	0	0	0	0
lg03a06	ok	-	32	2	2	0	0	0
lg03a07	ok	-	32	4	4	0	0	0
lg03a08	ok	-	32	13	13	0	0	0
lg03a09	ok	-	32	15	15	0	0	0
lg03a10	ok	-	32	5	5	0	0	0
lg03a11	ok	-	32	8	8	0	0	0
lg05e01	ok	-	96	25	25	0	0	0
lg05e02	ok	-	96	32	32	0	0	0
lg05e03	ok	-	96	26	26	0	0	0
lg05e04	ok	-	96	18	18	0	0	0
lg05e05	ok	-	96	32	32	0	0	0
lg05e06	ok	-	96	32	32	0	0	0
lg05e07	ok	-	96	32	32	0	0	0
lg05e08	ok	-	96	32	32	0	0	0
lg05e09	ok	-	96	32	32	0	0	0
lg05e10	ok	-	96	25	25	0	0	0
lg05e11	ok	-	96	32	32	0	0	0
lg05e12	ok	-	96	32	32	0	0	0
lg05e13	ok	-	96	25	25	0	0	0
lg05e14	ok	-	96	15	15	0	0	0
lg05e15	ok	-	96	32	32	0	0	0
lg05e16	ok	-	96	32	32	0	0	0

```
...
```

bhosts (continue)

```
[choh07@li04e02 ~]$ bhosts interactive
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
lc03e16	ok	-	96	1	1	0	0	0
lc03e17	ok	-	96	10	10	0	0	0
lg03a01	ok	-	32	7	7	0	0	0

```
[choh07@li04e02 ~]$ bhosts long
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
lh05g02	ok	-	64	1	1	0	0	0
lh05g03	ok	-	64	2	2	0	0	0
lh05g04	ok	-	64	4	4	0	0	0
lh05g05	ok	-	64	5	5	0	0	0
lh05g06	ok	-	64	11	11	0	0	0
lh05g07	ok	-	64	0	0	0	0	0
lh05g08	ok	-	64	2	2	0	0	0
lh05g09	ok	-	64	4	4	0	0	0
lh05g10	ok	-	64	3	3	0	0	0

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*

bjobs : status of jobs

```
[choh07@li04e01 ~]$ bjobs -l 202030609
```

```
Job <202030609>, User <choh07>, Project <acc_hpcstaff>, Application <default>,  
Status <RUN>, Queue <premium>, Job Priority <50>, Command  
<#!/bin/bash;#BSUB -n 1;#BSUB -W 5;#BSUB -q premium;#BSUB  
-e stderr.test;#BSUB -o stdout.test;#BSUB -P acc_hpcstaff;  
cd /sc/arion/work/choh07/testsuite; module load BN; echo "  
Hello, World!";sleep 180>, Share group charged </choh07>,  
Esub <sinai>  
Tue Sep 23 11:46:34: Submitted from host <li04e02>, CWD </sc/arion/work/choh07/  
testsuite>, Output File <stdout.test>, Error File <stderr.  
test>, Re-runnable, Requested Resources < rusage[mem=3000]  
>;  
Tue Sep 23 11:46:37: Started 1 Task(s) on Host(s) <lh06c27>, Allocated 1 Slot(s  
) on Host(s) <lh06c27>, Execution Home </hpc/users/choh07>  
, Execution CWD </sc/arion/work/choh07/testsuite>;  
Tue Sep 23 11:47:12: Resource usage collected.  
MEM: 5 Mbytes; SWAP: 0 Mbytes; NTHREAD: 5  
PGID: 152866; PIDs: 152866 152867 152871 152877
```

```
RUNLIMIT  
5.0 min
```

```
MEMLIMIT  
2.9 G
```

```
MEMORY USAGE:  
MAX MEM: 12 Mbytes; AVG MEM: 5 Mbytes; MEM Efficiency: 0.40%
```

```
CPU USAGE:  
CPU PEAK: 0.00 ; CPU PEAK DURATION: 0 second(s)  
CPU AVERAGE EFFICIENCY: 0.00% ; CPU PEAK EFFICIENCY: 0.00%
```

```
SCHEDULING PARAMETERS:
```

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

```
RESOURCE REQUIREMENT DETAILS:
```

```
Combined: select[(healthy=1) && (type == local)] order[!-slots:-maxslots] rusa  
ge[mem=3000.00] same[model] affinity[core(1)*1]  
Effective: select[(healthy=1)) && (type == local)] order[!-slots:-maxslots] r  
usage[mem=3000.00] same[model] affinity[core(1)*1]
```

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

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 5 -I 2937044
```

```
Job <2937044>, Job Name <myfirstjob>, User <gail01>, Project <acc_hpcstaff>, Application <default>, Command <#!/bin/bash;#BSUB -J myfirstjob;#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[((headlthy=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	UNKWN	TOTAL
71	0	5	0	0	0	76

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
```

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
```

```
/my/bin/executable < my_data.in
```

#sets the number of threads

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_awesome_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/gpuexpress queue for batch

- GPU option specification:

-gpu num=*Ngpus* **-R** GPU_Model

e.g. **-gpu** num=4 **-R** h100nvl

Ngpus : Number of GPU cards requested **PER NODE**.

To request GPU cards on the same node, “**-R span[hosts=1]**” MUST be added.

GPU_Model	
v100	TeslaV100_PCIE_16GB
a100	NVIDIAA100_PCIE_40GB
a10080g	NVIDIAA100_SXM4_80GB
h10080g	NVIDIAH100_PCIE_80GB
h100nvl	NVIDIAH100_SXM5_80GB
l40s	NVIDIAL40S_PCIE_48GB
b200	NVIDIAB200

GPGPU (continue)

```
[choh07@li04e04 ~]$ bhosts -R h100nvl
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
lg02e05	ok	-	96	0	0	0	0	0
lg02e13	ok	-	96	0	0	0	0	0
lg02e03	ok	-	96	0	0	0	0	0
lg03e06	ok	-	96	4	4	0	0	0
lg03e08	ok	-	96	0	0	0	0	0
lg03e05	ok	-	96	0	0	0	0	0
lg03e04	ok	-	96	0	0	0	0	0
lg02e01	ok	-	96	0	0	0	0	0
lg02e06	ok	-	96	0	0	0	0	0
lg03e02	ok	-	96	0	0	0	0	0
lg02e02	ok	-	96	0	0	0	0	0
lg02e07	ok	-	96	0	0	0	0	0
lg02e04	ok	-	96	0	0	0	0	0
lg02e09	ok	-	96	0	0	0	0	0
lg03e07	ok	-	96	4	4	0	0	0
lg02e11	ok	-	96	8	8	0	0	0
lg03e11	ok	-	96	1	1	0	0	0
lg02e16	ok	-	96	1	1	0	0	0
lg02e15	ok	-	96	1	1	0	0	0
lg05e15	ok	-	96	4	4	0	0	0
lg03e10	ok	-	96	89	89	0	0	0
lg03e15	ok	-	96	28	28	0	0	0
lg03e14	ok	-	96	44	44	0	0	0
lg05e14	ok	-	96	64	64	0	0	0
lg03e12	ok	-	96	47	47	0	0	0
lg05e16	ok	-	96	64	64	0	0	0

GPGPU (continue)

```
#BSUB -q gpu
```

```
#BSUB -n 4
```

```
#BSUB -gpu num=2
```

```
#BSUB -R v100
```

```
#BSUB -R span[hosts=1]
```

```
module purge
```

```
module load anaconda3 ( or 2)
```

```
module load cuda
```

```
source activate tfGPU
```

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

```
# submit to gpu queue
```

```
# number of CPU cores on the node
```

```
# 2 GPUs on V100 node
```

```
# all CPU cores and GPU cards on the same node
```

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 -R span[ptile=2]
#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)

`/path/to/bin/program_to_run < input_jason > output_jason`

`/path/to/bin/program_to_run < input_tom > output_tom`

`...`

`/path/to/bin/program_to_run < input_jane > output_jane`

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.

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 3 GB 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 256 -R span[hosts=1]

Tips for efficient usage of the queuing system

- Request reasonable resource
 - **Prior knowledge needed.** (Try short test runs before production to get a reasonable estimate)
 - User limit:
 - Max running jobs per user: 4,000
 - Max pending jobs per user: 20,000
 - Max num. of GPUs per user: 45
 - Global Memory limit: 30TB
 - Heavy users: depending on the resource requested
 - Monitor resource usage of a running job: “***bjobs -l JobID***”
 - ...
 - MEMORY USAGE:
MAX MEM: 68.1 Gbytes; AVG MEM: 37.4 Gbytes; **MEM Efficiency: 79.83%**
 - CPU USAGE:
CPU PEAK: 19.89 ; **CPU Efficiency: 99.43%**

Tips for efficient usage of the queuing system

- Find appropriate queue and nodes
 - use -q interactive: for debug (both CPU and GPU with internet access)
 - use -q express if walltime < 12h
- Memory request is **per core** in MB, *not per job*.
- You can open an interactive session on a regular compute node, too.
`bsub -q premium -n ... -W ... -P ... -ls /bin/bash`
- Job not start after a long pending time
 - Whether the resource requested is non-exist:
`-R rusage[mem = 100GB] -n 256 -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>.
```
- /sc/arion/{projects,work,scratch} not backed up,
- Efficient use of limited resources.
- Job temporary dir configured to /local/JOBS instead of /tmp.

Final Friendly Reminder

All publications utilizing Minerva resources must include one of the following acknowledgements depending on your funding status:

- **If your NIH-funded GPU-related projects use the AIMS B200 GPUs, you must include the following in all your publications:** *“This work was supported in part through the Minerva computational and data resources and staff expertise provided by Scientific Computing and Data at the Icahn School of Medicine at Mount Sinai and supported by the Clinical and Translational Science Awards (CTSA) grant UL1TR004419 from the National Center for Advancing Translational Sciences. Research reported in this publication was also supported by the Office of Research Infrastructure of the National Institutes of Health under award number S10OD038231. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.”*
- **If you DO NOT have NIH-funded GPU-related projects on Minerva, you MUST include the following in all your publications:** *“This work was supported in part through the Minerva computational and data resources and staff expertise provided by Scientific Computing and Data at the Icahn School of Medicine at Mount Sinai and supported by the Clinical and Translational Science Awards (CTSA) grant UL1TR004419 from the National Center for Advancing Translational Sciences.”*

Last but not Least

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

hpchelp@hpc.mssm.edu

Follow us by visiting

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

Acknowledgements

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

