

Building and running codes on CORAL SIERRA systems (SIERRA, RZANSEL and LASSEN)

Presented by John Gyllenhaal

Updated December 3rd, 2018



Building and running codes currently significantly different on SIERRA than on CORAL EA systems

- **All** bsub scripts run on a few shared launch nodes!
 - SIERRA has 5 login nodes and 5 shared launch nodes
 - RZANSEL has just 1 login node and 1 shared launch node
 - LASSEN has 3 nodes to divide up, 2 shared launch nodes
 - On CORAL EA, script and interactive sessions run on dedicated backend node
 - Many CORAL EA bsub scripts will hammer shared launch nodes unless modified
- Must explicitly run `make`, `spack`, etc. on SIERRA backend nodes
 - Provide helper `'lrun -1'` to launch heavy weight commands on backend nodes
 - For example: `lrun -1 spack install zlib`
 - Light activity (e.g. tiny compiles) ok on shared nodes but easy to impact others
- SIERRA bsub in new 'easy' mode, google returns bad bsub info
 - Node allocated, so use `-nnodes <nodes>` not `-n <slots>`
- LLNL's helper scripts (`lalloc`, `lrun`) help straightforward tasks
- LLNL's FLUX scheduler enables non-straightforward running schemes

Current but hopefully temporary known limitations on SIERRA Systems

- You must have passphrase-less ssh keys set up or get cryptic errors
 - See: <https://lc.llnl.gov/confluence/display/SIERRA/Quickstart+Guide>
 - Most common problem - Everyone needs to set them up the first time
 - Replacement technology implemented but still being debugged at scale
- Can launch parallel jobs from backend nodes with some limitations
 - Several race conditions and bugs scheduled to be fixed ~Dec 17th
 - Control-C breaks allocation and simultaneous jsruns can trigger failures
 - 'lalloc' puts you on backend node, oldlalloc puts you on shared launch node
 - Working towards getting batch scripts optionally on backend nodes also
 - Batch scripts currently need to offload all work to backend nodes!
- Difficult to get good default jsrun configuration, see quickstart guide
 - Many jsrun options currently required to get desired behavior (or use lrun)
 - Plan to develop lpack for regression tests once Dec 17th drop installed
- A different environment than CORAL EA systems
 - Painful to have two different environments
 - IBM approved CORAL EA systems to same software (jsrun w/ launch nodes)
 - Hope to have bandwidth in early 2019 after Sierra stable/swung to closed side

Key commands everyone should know

See <https://lc.llnl.gov/confluence/display/SIERRA/Quickstart+Guide>

- **lsfjobs**: LLNL-specific script to print queue info
- **bsub**: submit a batch script to LSF to allocate nodes and run app
- **lalloc**: LLNL-specific bsub wrapper to get interactive allocation on backend
- **oldlalloc**: Wrapper to get interactive allocation on shared launch node
- **jsrun**: launch parallel job on backend nodes of LSF allocation
- **lrun**: LLNL-specific jsrun wrapper to launch and bind jobs on backend nodes
- **mpibind**: LLNL-specific bind utility (now jsrun compatible as of 7/23/18)
- **check_sierra_nodes**: LLNL-specific script to test nodes in allocation
- **bkill**: Kill queued or running LSF jobs
- LLNL-specific utilities (lrun, etc.) live in /usr/tcetmp/bin

Commands power users might find useful

See <https://lc.llnl.gov/confluence/display/SIERRA/Running+Jobs>

- **js_task_info**: MPI utility app that prints binding info for each MPI rank
- **bstop**: suspend a pending job, so it will not be scheduled to run
- **bresume**: re-enable a suspended job, so it can be scheduled to run
- **bjobs**: display your jobs in the scheduling queues, one job per line
- **bjobs -l <jobID>**: display detailed info about any running job
- **bhist -l <jobID>**: display info about a finished job
- **bmod**: modify a job's requirements (e.g. add dependency)
- **bpeek**: display the stdout and stderr output of a running job (only yours)
- **bqueues**: display open/closed state of available queues (during DATs)
- **bugroup**: display user group membership (such as guests and lcstaff)

lsfjobs output example

```
sierra4360{gyllen}23: lsfjobs
```

```
JOBID  PROCS  PFILE  NODES  USER  STATE  PRIO  QUEUE  GROUP  REMAINING  LIMIT
143409 20480   40    512  gyllen  RUN    -    pbatch  guests  00:59:51  01:00:00
```

```
QUEUE  TOTAL  DOWN  BUSY  FREE  DEFAULTTIME  MAXTIME  NODE_GROUP(s)
pbatch  2142  157   918  1067          30:00  1:00:00:00  batch_hosts
pdebug   18    0     0    18          30:00  1:00:00    debug_hosts
```

- Use `bjobs -l jobid` to get details on running or pending jobs

Example of compiling and running MPI OpenMP 4.5 GPU code on SIERRA

- Example: Allocate 1 interactive backend node, build and run app

```
sierra4360{gyl1en}2: oldl1alloc 1 <- emulates bsub script, on shared node  
+ exec bsub -nnodes 1 -Is -XF -w 60 -G guests -core_isolation 2 /bin/tcsh  
Job <143406> is submitted to default queue <pbatch>.
```

```
<<ssh X11 forwarding job>>
```

```
<<Waiting for dispatch ...>>
```

```
<<Starting on sierra4368>> <- Not appearing indicates critical error
```

```
sierra4368{gyl1en}2: cd ~/debug/hasgpu  
/g/g0/gyl1en/debug/hasgpu
```

```
sierra4368{gyl1en}3: lrun -1 make  
mpicc-gpu -O mpihasgpu.c -o mpihasgpu
```

```
sierra4368{gyl1en}4: lrun -n 4 ./mpihsgpu
```

```
Rank  1 Host sierra1221  Able to use GPU 1  CPUs 40  
Rank  0 Host sierra1221  Able to use GPU 0  CPUs  0  
Rank  2 Host sierra1221  Able to use GPU 2  CPUs 88  
Rank  3 Host sierra1221  Able to use GPU 3  CPUs 128
```

```
sierra4368{gyl1en}5: exit
```

Usage: lalloc nodes [bsub_options]

- Uses reasonable defaults to create interactive bsub command
 - Defaults to 60 minutes in default queue with -core_isolation 2
 - Now redirects you to first backend node node (your private node)
 - Currently always prints actual bsub line used

```
sierra4359{gyllen}7: lalloc 2 -w 30 -q pdebug
+ exec bsub -nnodes 2 -w 30 -q pdebug -Is -XF -G guests -core_isolation 2 /bin/tcsh
Job <143499> is submitted to queue <pdebug>
<<ssh X11 forwarding job>>
<<waiting for dispatch ...>>
<<Starting on sierra4369>>
<<Redirecting to compute node sierra2972, setting up as private launch node>>
```

- Specify normal bsub options to override defaults
 - -W 30 (30 minute max runtime)
 - -q pdebug (pick specific queue, pdebug from lsjob output)
 - -core_isolation 0 (disables core isolation 2 cores per socket, 4 total default)
 - -G lcstaff (pick different group/bank to run under, uses default bank by default)
 - Can also set env var LSB_DEFAULT_USERGROUP to pick default group/bank

Usage: `lrun -T<ntasks_per_node> |-n<ntasks> |-1 [-N<nnodes>] [--nolbind] [<jsrun_options>] <app> [app-args]`

- Runs jsrun with reasonable default binding behavior
 - Use 'lrun -1' to run make, spack, etc on one backend node in bsub script
 - Use -lrun -n tasks to run over all allocated nodes, max 1 socket per task
 - mpibind prevents more than 20 OpenMP threads per task, one socket, by design
 - Evolves with each rapidly-evolving jsrun release, sometimes uses resource file
 - Set MPIBIND to 'j' to see jsrun command line created

```
sierra4367{gyl}2: setenv MPIBIND j
```

```
sierra4367{gyl}3: lrun -1 make  
+ exec jsrun --np 1 -c 40 -g 4 -r 1 -d plane:1 --bind none --exit_on_error 0 make  
mpicc-gpu -O mpihasgpu.c -o mpihasgpu
```

```
sierra4367{gyl}9: lrun -n 4 js_task_info |& sort  
+ exec jsrun --np 4 -c 40 -g 4 -r 1 -d plane:4 --bind none --exit_on_error 1  
/usr/tcetmp/packages/mpibind/bin/mpibind js_task_info
```

```
Task 0 ( 0/4, 0/4 ) is bound to cpu[s] 0,4,8,12,16,20,24,28,32,36 on host  
sierra1262 with OMP_NUM_THREADS=10 and with  
OMP_PLACES={0},{4},{8},{12},{16},{20},{24},{28},{32},{36} and  
CUDA_VISIBLE_DEVICES=0  
<snip>
```

Running batch jobs with bsub

See <https://lc.llnl.gov/confluence/display/SIERRA/Running+Jobs>

- bsub currently only accepts #BSUB lines on stdin
 - Recommended invocation: `bsub < script`
 - Command line option override #BSUB options
 - `#!` sets shell language (defaults to bash)
 - IBM working on reading #BSUB options from file on command line
- Pick bsub dependency option that runs after crashed jobs
 - Use `-w 'ended(job_name)'` or `-w 'ended(job_id)'`
 - Usually submitted in bsub script so `LSB_JOBID` available for use
 - bsub will reject if dependency not running or recently completed (can be tricky)
- Use `lrun/jsrun` inside script to launch everything on backend nodes
- `lrun` by default adds `--exit_on_error 1`, add if using `jsrun` directly
 - Otherwise one node's tasks segfaulting or exiting out may hang job

Example of script 'cat << EOF' trick useful for submitting bsub jobs from scripts

```
sierra4359{gyllen}52: cat do_simple_bsub  
#!/bin/sh
```

```
cat << EOF | bsub -nnodes 32 -w 360  
#!/bin/bash    <- optionally set shell language, bash default  
#BSUB -core_isolation 2 -G guests -J "MYJOB1"  
cd ~/debug/hasgpu  
lrun -T 4 ./mpihsgpu arg1 arg2  
EOF
```

```
sierra4359{gyllen}53: ./do_simple_bsub  
Job <143505> is submitted to default queue <pbatch>.
```



Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.