Submitting and Running Jobs

Scott French
NERSC User Services Group

New User Training
August 13, 2015
Jobs at NERSC

• Most are parallel jobs (10s to 100,000+ cores)
• Production runs execute in batch mode
• Interactive and debug jobs are supported for up to 30 minutes
• Typically run times are a few to 10s of hours.
  – Each machine has different limits.
  – Limits are necessary because of MTBF and the need to accommodate 5,500 users’ jobs

• Also a number of “serial” jobs
  – Typically “pleasantly parallel” simulation or data analysis
Login Nodes and Compute Nodes

• Each machine has 3 types of nodes visible to users
  
  • **Login nodes**
    – Edit files, compile codes, submit batch jobs, etc.
    – Run short, serial utilities and applications

  • **Compute nodes**
    – Execute your application
    – Dedicated resources for your job

  • Shared application launcher or “**MOM**” nodes
    – Execute your batch script commands

• **Note:** This will change when we move to SLURM
Launching Parallel Jobs (Cray system)

Login Node → qsub → MOM Node → aprun → Compute Nodes
Launching Parallel Applications

• An “application launcher” executes your code
  – Starts multiple instances of your executable across the compute nodes you were allocated
  – Manages execution of your application
  – On Edison / Hopper: the launcher is called “aprun”

• Only the application launcher can start your application on compute nodes

• You can’t run the launcher from login nodes (only from a batch script or interactive session)
Submitting Batch Jobs

• To run a batch job on the compute nodes you must write a “batch script” that contains
  – Directives to allow the system to schedule your job
  – An aprun command that launches your parallel executable (this will change to srun under SLURM)

• Submit the job to the queuing system with the qsub command
  – % qsub my_batch_script
Edison - Cray XC30

• 133,824 cores, 5,576 nodes
• “Aries” interconnect
• 2 x 12-core Intel ’Ivy Bridge’ 2.4 GHz processors per node
• 24 processor cores per node, 48 with hyperthreading
• 64 GB of memory per node
• 357 TB of aggregate memory
• 2.7 GB memory / core for applications
• /scratch disk quota of 10 TB
• 7.6 PB of /scratch disk
• Choice of full Linux operating system or optimized Linux OS (Cray Linux)
• Intel, Cray, and GNU compilers
Sample Edison Batch Script - MPI

```
#PBS -q debug
#PBS -l mppwidth=96
#PBS -l walltime=00:10:00
#PBS -N my_job

cd $PBS_O_WORKDIR
aprun -n 96 ./my_executable
```
Job directives: instructions for the batch system

- Submission queue
- How many compute cores to reserve for your job (/ 24 = # nodes)
- How long to reserve those nodes
- Optional: what to name STDOUT files, what account to charge, whether to notify you by email when your job finishes, etc.
Sample Edison Batch Script - MPI

```bash
#PBS -q debug
#PBS -l mppwidth=96
#PBS -l walltime=00:10:00
#PBS -N my_job

cd $PBS_O_WORKDIR
aprun -n 96 ./my_executable
```

Change from home directory to job submission directory

- Script is initially run from your home directory, **which is not advisable** (as we mention in the filesystem intro)
- You will see much better performance if your job reads / writes from one of the high-performance scratch filesystems
Sample Edison Batch Script - MPI

Launches parallel executable on the compute nodes

- Carries over (partial) login environment
- Controls how your executable:
  - maps to processors on the compute nodes (e.g. how many tasks?)
  - accesses the memory on each processor

```bash
#PBS -q debug
#PBS -l mppwidth=96
#PBS -l walltime=00:10:00
#PBS -N my_job

cd $PBS_O_WORKDIR
aprun -n 96 ./my_executable
```
Sample Edison Batch Script - MPI

```bash
#PBS -q debug
#PBS -l mppwidth=96
#PBS -l walltime=00:10:00
#PBS -N my_job

cd $PBS_O_WORKDIR
aprun -n 96 ./my_executable
```

mppwidth is number of compute cores requested for your job
- mppwidth = 24 x # of nodes on Edison (and Hopper)
- must be **greater than or equal to** the number of tasks requested (\(-n\))
Sample Edison Batch Script - MPI

```bash
#PBS -q debug
#PBS -l mppwidth=192
#PBS -l walltime=00:10:00
#PBS -N my_job

cd $PBS_O_WORKDIR
aprun -n 96 -N 12 .:/my_executable

-N = number of tasks per node
 Might do this to get more memory / task
 Note that mppwidth has changed accordingly
```
Sample Edison Batch Script - MPI

```bash
#PBS -q debug
#PBS -l mppwidth=48
#PBS -l walltime=00:10:00
#PBS -N my_job

cd $PBS_O_WORKDIR
aprun -n 96 -j 2 ./my_executable

-j = Turn on hyperthreading
```
#PBS -q regular
#PBS -l mppwidth=96
#PBS -l walltime=00:10:00
#PBS -N my_job

cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=6
aprun -n 16 -d 6 -N 4 -S 2 ./hybrid.x

A more complex example for mixing MPI and OpenMP:
• 16 tasks (–n), 4 on each node (–N), 6 OpenMP threads per task (–d),
  assign 2 tasks to each NUMA node (–S)
Many more examples on www.nersc.gov
Interactive Parallel Jobs

- You can run small parallel jobs interactively for up to 30 minutes (ex. is for Hopper / Edison)

```
login% qsub -I -l mppwidth=48
[wait for job to start]

mom% cd $PBS_O_WORKDIR
mom% aprun -n 48 ./mycode.x
```
Serial Jobs

• Both Hopper and Edison now have a special queue for running serial jobs
  – A single process running on a single core
  – Each serial node can run up to 24 jobs from different users depending on their memory requirements

```bash
#PBS -q serial
#PBS -l walltime=00:10:00
#PBS -l vmem=4GB
#PBS -N my_job

cd $PBS_O_WORKDIR
./myexecutable
```
Monitoring Your Job

• Once your job is submitted, it enters the queue and will start when resources are available.

• Your job’s place in the queue is a mix of time and priority, so line jumping is allowed, but it may cost more.

• You can monitor it with:
  – `qstat -a`
  – `qstat -u username`
  – `showq`
  – `qs`

• On the web:
  - [https://my.nersc.gov](https://my.nersc.gov)
### Job Limits

There are per user, per machine job limits. Here are the limits on Edison as of August, 2015.

Specify these queues with

```bash
#PBS -q queue_name
```

Not these!

<table>
<thead>
<tr>
<th>Submit Queue</th>
<th>Execution Queue</th>
<th>Nodes</th>
<th>Physical Cores</th>
<th>Max Wallclock (hours)</th>
<th>Relative Priority</th>
<th>Run Limit</th>
<th>Eligible Limit</th>
<th>Charge Factor*</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>debug</td>
<td>1-512</td>
<td>1-12,288</td>
<td>30 mins</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ccm_int¹</td>
<td>ccm_int</td>
<td>1-512</td>
<td>1-12,288</td>
<td>30 mins</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>regular</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reg_small</td>
<td>1-682</td>
<td>1-16,368</td>
<td>48 hrs</td>
<td>3</td>
<td>24</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>reg_med</td>
<td>683-2048</td>
<td>16,369-49,152</td>
<td>36 hrs</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>reg_big</td>
<td>2049-4096</td>
<td>49,153-98,304</td>
<td>36 hrs</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>reg_xbig</td>
<td>4097-5462</td>
<td>98,305-131,088</td>
<td>12 hrs</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1.2</td>
</tr>
<tr>
<td>ccm_queue</td>
<td>ccm_queue</td>
<td>1-682</td>
<td>1-16,368</td>
<td>96 hrs</td>
<td>3</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>premium</td>
<td>premium</td>
<td>1-2048</td>
<td>1-49,152</td>
<td>36</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>low</td>
<td>low</td>
<td>1-682</td>
<td>1-16,368</td>
<td>24</td>
<td>4</td>
<td>16</td>
<td>6</td>
<td>1.0</td>
</tr>
<tr>
<td>killable²</td>
<td>killable</td>
<td>1-682</td>
<td>1-16,368</td>
<td>48 hrs</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>serial³</td>
<td>serial</td>
<td>1</td>
<td>1</td>
<td>48 hrs</td>
<td>-</td>
<td>50</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>xfer</td>
<td>xfer</td>
<td>-</td>
<td>-</td>
<td>12</td>
<td>-</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

¹ ccm_int is the default queue.
² killable is available for all users.
³ serial is available for serial jobs.
Tips for jobs

• Submit shorter jobs, they are easier to schedule
  – Checkpoint if possible to break up long jobs
  – Short jobs can take advantage of ‘backfill’ opportunities
  – Run short jobs just before maintenance

• Very important: make sure the wall clock time you request is accurate
  – As noted above, shorter jobs are easier to schedule
  – Many users unnecessarily enter the largest wall clock time possible as a default
How Your Jobs Are Charged

• Your repository is charged for each node your job was allocated for the entire duration of your job.
  – The minimum allocatable unit is a node (except for the serial queues). Hopper and Edison have 24 cores / node, so your minimum charge is 24*walltime.

MPP hours = (# nodes) * (# cores / node) * (walltime) * (QCF) * (MCF)

  – Example: 96 Edison cores for 1 hour in regular queue
    MPP hours = (4) * (24) * (1 hour) * (1) * (2) = 192 MPP hours
    – Serial jobs are charged with: (walltime) * (MCF)

• If you have access to multiple repos, pick which one to charge in your batch script
  
  #PBS -A repo_name
• Each machine has a “machine charge factor” (MCF) that multiplies the “raw hours” used
  – Edison MCF = 2.0
  – Hopper MCF = 1.0
  – Carver MCF = 1.5

• Each queue has a “queue charge factor” (QCF) and corresponding relative scheduling priorities
  – Premium QCF = 2.0
  – Low QCF = 0.5
  – Regular (and everything else) QCF = 1.0 (Hopper: 0.8)

• On Edison:
  – Jobs requesting more than 682 nodes (reg_med, reg_big, reg_xbig queues) get a 40% discount (QCF = 0.6)
More Information

NERSC Web pages

Hopper:
http://www.nersc.gov/users/computational-systems/hopper/running-jobs/

Edison:
http://www.nersc.gov/users/computational-systems/edison/running-jobs/

Carver (retiring September, 2015):
http://www.nersc.gov/users/computational-systems/carver/running-jobs/

Contact NERSC Consulting:

- Toll-free 800-666-3772
- 510-486-8611, #3
- Email consult@nersc.gov.
Thank You