Submitting and Running Jobs

Helen He
NERSC User Engagement Group

New User Training
March 21, 2016
Jobs at NERSC

• Most are parallel jobs (10s to 100,000+ cores)
• Also a number of “serial” jobs
  – Typically “pleasantly parallel” simulation or data analysis
• Production runs execute in batch mode
• Our batch scheduler is SLURM (native)
• 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 6,000 users’ jobs
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
Cori Phase 1 - Cray XC40

- 52,160 cores, 1,630 nodes
- “Aries” interconnect
- 2 x 16-core Intel ’Haswell' 2.3 GHz processors per node
- 32 processor cores per node, 64 with hyperthreading
- 128 GB of memory per node
- 203 TB of aggregate memory
- 4 GB memory / core for applications
- /scratch disk quota of 20 TB
- 30 PB of /scratch disk
- Choice of full Linux operating system or optimized Linux OS (Cray Linux)
- Intel, Cray, and GNU compilers
Cori Phase 1 Compute Nodes

- Cori Phase 1: NERSC Cray XC40, 1,630 nodes, 52,160 cores.
  - Each node has 2 Intel Xeon 16-core Haswell processors.
  - 2 NUMA domains per node, 16 cores per NUMA domain.
    2 hardware threads per core.
- Memory bandwidth is non-homogeneous among NUMA domains.

To obtain processor info:
Get on a compute node:
% salloc –N 1
Then:
% cat /proc/cpuinfo
or
% hwloc-ls
Login Nodes and Compute Nodes

Each machine has 2 types of nodes visible to users

• **Login nodes (external)**
  – Edit files, compile codes, submit batch jobs, etc.
  – Run short, serial utilities and applications

• **Compute nodes**
  – Execute your application
  – Dedicated resources for your job
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 `srun` command that launches your parallel executable

• Submit the job to the queuing system with the `sbatch` command
  – `% sbatch my_batch_script`
Launching Parallel Jobs with SLURM

**Head compute node:**
- Runs commands in batch script
- Issues job launcher “srun” to start parallel jobs on all compute nodes (including itself)

**Other Compute Nodes allocated to the job**

**Login node:**
- Submit batch jobs via sbatch or salloc
- Please do not issue “srun” from login nodes
- Do not run big executables on login nodes
#!/bin/bash -l
#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -n 1280
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 ./mycode.exe
Sample Cori Batch Script - MPI

```bash
#!/bin/bash -l
#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -n 1280
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 .:/mycode.exe
```

- Need to specify which shell to use for batch script
- Use “-l” as login shell is optional.
- Environment is automatically imported
Sample Cori Batch Script - MPI

#!/bin/bash -l

#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -n 1280
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 ./mycode.exe

Job directives: instructions for the batch system
• Submission partition (default is “debug”)
• How many compute nodes to reserve for your job
• How long to reserve those nodes
• More optional SBATCH keywords
Sample Cori Batch Script - MPI

```bash
#!/bin/bash -l
#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -n 1280
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 ./mycode.exe
```

SBATCH optional keywords:
- how many instances of applications to launch (# of MPI tasks)
- which QOS to use via "#SBATCH --qos=..." (default is normal)
- what to name STDOUT files
- what account to charge
- whether to notify you by email when your job finishes
- ...
#!/bin/bash -l
#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -n 1280
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 ./mycode.exe

- By default, hyperthreading is on. SLURM sees 2 threads are available for each of the 32 physical CPUs on the node.
- No need to set this if your application programming model is pure MPI.
- If your code is hybrid MPI/OpenMP, set this value to 1 to run in pure MPI mode.
Sample Cori Batch Script - MPI

#!/bin/bash -l
#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -n 1280
#SBATCH -J myjob
export OMP_NUM_THREADS=1
srun -n 1280 ./mycode.exe

“srun” command launches parallel executables on the compute nodes
• srun flags overwrite SBATCH keywords
• No need to repeat flags in srun command if already defined in SBATCH keywords. (e.g. “srun ./my_executable” will also do in above example)
Sample Cori Batch Script - MPI

```bash
#!/bin/bash -l
#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -n 1280
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 ./mycode.exe
```

- There are 64 logical CPUs on each node
- With 40 nodes, using hyperthreading, up to 40*64=2,560 MPI tasks can be launched: “srun -n 2560 ./my_executable” is OK
Hybrid MPI/OpenMP

```bash
#!/bin/bash -l
#SBATCH -p regular
#SBATCH -N 40
#SBATCH -t 1:00:00
export OMP_NUM_THREADS=8
srun -n 160 -c 8 ./mycode.exe
```

- `srun` does most of optimal process and thread binding automatically. Only flags such as “-n” “-c”, along with OMP_NUM_THREADS are needed for most applications.
- Hyperthreading is enabled by default. Jobs requesting more than 32 cores (MPI tasks * OpenMP threads) per node will use hyperthreads automatically.
Interactive Parallel Jobs

• You can run small parallel jobs interactively for up to 30 minutes

  login% salloc -N 2 -p debug -t 15:00
  [wait for job to start]
  compute% srun -n 64 ./mycode.exe
Serial Jobs on Cori

• The “shared” partition on Cori allows multiple executables from different users to share a node
• Each serial job run on a single core of a “shared” node
• Up to 32 jobs from different users depending on their memory requirements

```bash
#SBATCH -p shared
#SBATCH -t 1:00:00
#SBATCH --mem=4GB
#SBATCH -J my_job
./mycode.x
```

• Small parallel job that use less than a full node can also run in the “shared” partition

• Do not specify #SBATCH -N”
• Default “#SBATCH -n” is 1
• Default memory is 1,952 MB
• Use -n or --mem to request more slots for larger memory
• Do not use “srun” for serial executable (reduces overhead)
### Edison Queue Policy (as of March 2016)

Specify these partitions with 
`#SBATCH -q partition_name`

Specify these QOS with 
`#SBATCH --qos=premium`

These limits are per user per partition/QOS limits

<table>
<thead>
<tr>
<th>Partition</th>
<th>Nodes</th>
<th>Physical Cores</th>
<th>Max Wallclock</th>
<th>QOS(^1)</th>
<th>Run Limit</th>
<th>Submit Limit</th>
<th>Relative Priority</th>
<th>Charge Factor(^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>1-512</td>
<td>1-12,288</td>
<td>30 mins</td>
<td>-</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1-682</td>
<td>1-16,368</td>
<td>36 hrs</td>
<td>normal</td>
<td>24</td>
<td>100</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>premium</td>
<td>8</td>
<td>20</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>low</td>
<td>24</td>
<td>100</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scavenger</td>
<td>8</td>
<td>100</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>regular</td>
<td>683-5462</td>
<td>16,369-130,181</td>
<td>36 hrs</td>
<td>normal</td>
<td>8</td>
<td>100</td>
<td>2</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>premium</td>
<td>2</td>
<td>20</td>
<td>1</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>low</td>
<td>8</td>
<td>100</td>
<td>5</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scavenger</td>
<td>8</td>
<td>100</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>xfer(^3)</td>
<td></td>
<td></td>
<td>24 hrs</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Jobs with insufficient allocations to run are directed to “scavenger”

---

\(^1\) QOS types: normal, premium, low, scavenger

\(^2\) Charge factors: 2, 4, 6

\(^3\) xfer: transfer queue
# Cori Queue Policy (as of March 2016)

<table>
<thead>
<tr>
<th>Partition</th>
<th>Nodes</th>
<th>Physical Cores</th>
<th>Max Walltime per Job</th>
<th>QOS</th>
<th>Max Number of Running Jobs</th>
<th>Max Total Num Nodes per User for Running Jobs</th>
<th>Number of Jobs per User Submit Limit</th>
<th>Relative Priority</th>
<th>Charge Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>1-112</td>
<td>1-3,072</td>
<td>30 min</td>
<td>normal</td>
<td>1</td>
<td>112</td>
<td>5</td>
<td>3</td>
<td>2.5</td>
</tr>
<tr>
<td>regular</td>
<td>1-2</td>
<td>1-64</td>
<td>48 hrs</td>
<td>normal</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td>4</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>premium</td>
<td>10</td>
<td>100</td>
<td>40</td>
<td>2</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>low</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td>5</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scavenger</td>
<td>10</td>
<td>100</td>
<td>40</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>3-512</td>
<td>65-16,384</td>
<td>36 hrs</td>
<td></td>
<td>normal</td>
<td>10</td>
<td>512</td>
<td>50</td>
<td>4</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>premium</td>
<td>2</td>
<td>512</td>
<td>10</td>
<td>2</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>low</td>
<td>10</td>
<td>512</td>
<td>50</td>
<td>5</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scavenger</td>
<td>2</td>
<td>512</td>
<td>10</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>513-1,420</td>
<td>16,385-45,440</td>
<td>12 hrs</td>
<td></td>
<td>normal</td>
<td>1</td>
<td>1,420</td>
<td>4</td>
<td>4</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>premium</td>
<td>1</td>
<td>1,420</td>
<td>2</td>
<td>2</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>low</td>
<td>1</td>
<td>1,420</td>
<td>4</td>
<td>5</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scavenger</td>
<td>1</td>
<td>1,420</td>
<td>2</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>shared</td>
<td>1</td>
<td>1-16</td>
<td>48 hrs</td>
<td>normal</td>
<td>500</td>
<td>--</td>
<td>2,500</td>
<td>4</td>
<td>2.5/32</td>
</tr>
<tr>
<td>realtime</td>
<td>custom</td>
<td>custom</td>
<td>custom</td>
<td>custom</td>
<td>custom</td>
<td>--</td>
<td>1</td>
<td>1 (special permission)</td>
<td>--</td>
</tr>
<tr>
<td>xfer</td>
<td>1</td>
<td>1</td>
<td>12 hrs</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>1</td>
<td>--</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Large user limits**
- **For serial workload**
- **For realtime workflow**
Which System to Run My Jobs

- Queue configuration and policies are still under tuning for max throughput and system utilization.
- The Cori Phase 1 (also known as the "Cori Data Partition") system is designed to accelerate data-intensive applications.
  - 1-2 node jobs in “regular” partition for high throughput jobs: larger user limits, longer wall time limits
  - “shared” partition for serial workload: very large user limits
  - “realtime” partition for realtime workflow (special arrangement)
- Users are encouraged to run large size massive parallel jobs on Edison. Jobs use 683+ nodes on Edison get 40% charging discount.
Monitoring Your Job

• Once your job is submitted, it enters the queue and will start when resources are available
• Overall job priorities are a combination of partition, QOS, queue wait time, job size, wall time request, and fair share.
• You can monitor it with:
  – sqs
  – squeue
On the web:
https://my.nersc.gov
https://www.nersc.gov/users/live-status/ : “Queue Look”
https://www.nersc.gov/users/job-logs-and-analytics/completed-jobs/
SLURM User Commands

- **sbatch**: submit a batch script
- **salloc**: request nodes for an interactive batch session
- **srun**: launch parallel jobs
- **scancel**: delete a batch job
- **sqs**: NERSC custom queue display with job priority ranking info
- **squeue**: display info about jobs in the queue
- **sinfo**: view SLURM configuration about nodes and partitions
- **scontrol**: view and modify SLURM configuration and job state
- **sacct**: display accounting data for jobs and job steps
- **https://www.nersc.gov/users/computational-systems/cori/running-jobs/monitoring-jobs/**
Tips for Getting Better Throughput

• Line jumping is allowed, but it may cost more
• 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
• Queue wait time statistics
  – https://www.nersc.gov/users/queues/queue-wait-times/
Advanced Workflow Management

• Bundle jobs (multiple “srun”s in one script, sequential or simultaneously)

• Use Job Arrays for submitting and managing collections of similar jobs
  – Better managing jobs, not necessary faster turnaround
  – Each array task is considered a single job for scheduling

• Use job dependency features to chain jobs that have dependency
Charge Factors & Discounts

• Each machine has a “machine charge factor” (MCF) that multiplies the “raw hours” used
  – Edison MCF = 2.0
  – Cori MCF = 2.5

• Each QOS has a “QOS charge factor” (QCF)
  – premium QCF = 2.0
  – normal QCF = 1.0 (default)
  – low QCF = 0.5
  – scavenger QCF = 0

• On Edison:
  – Jobs requesting 683 or more nodes get a 40% discount
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 “shared” partition on Cori). Edison have 24 cores/node and Cori has 32 cores/node.

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

  – Example: 4 Cori nodes for 1 hour with “premium” QOS
    MPP hours = (4) * (32) * (1 hour) * (2) * (2.5) = 640 MPP hours
  – “shared” jobs are charged with physical CPUs used instead of entire node.

• If you have access to multiple repos, pick which one to charge in your batch script

  #SBATCH –A repo_name
More Information

NERSC Web pages:

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

• Cori
  http://www.nersc.gov/users/computational-systems/cori/running-jobs/

Contact NERSC Consulting:
  – Toll-free 800-666-3772
  – 510-486-8611, option #3
  – Email consult@nersc.gov
Thank You