SLURM Resource Manager is Coming to NERSC

Helen He
NUG Meeting, 11/06/2015
What is SLURM

• In simple word, SLURM is a workload manager, or a batch scheduler.

• SLURM stands for Simple Linux Utility for Resource Management.

• SLURM unites the cluster resource management (such as Torque) and job scheduling (such as Moab) into one system. Avoids inter-tool complexity.

• As of June 2015, SLURM is used in 6 of the top 10 computers, including the #1 system, Tianhe-2, with over 3M cores.
NERSC’s Plan to Adopt SLURM

- SLURM has been in production on Babbage (Intel Xeon Phi KNC test bed) as of August 14.
- SLURM is the batch scheduler for Cori Phase 1.
- Hopper stays with Torque/Moab until retire.
- Edison stays with Torque/Moab before moving to CRT. Edison will come back online at CRT with SLURM.
- We use the “native” SLURM (as compared to the “hybrid” SLURM).
Advantages of Using SLURM

- Fully open source.
- SLURM is extensible (plugin architecture)
- Low latency scheduling. Highly scalable.
- Integrated “serial” or “shared” queue
- Integrated Burst Buffer support
- Good memory management
- Built-in accounting and database support
- “Native” SLURM runs without Cray ALPS (Application Level Placement Scheduler)
  - Batch script runs on the head compute node directly
  - Easier to use. Less chance for contention compared to shared MOM node.
Native SLURM Architecture

User commands (partial list):
- scontrol
- sinfo
- squeue
- scancel
- sacct
- srun

Controller daemons:
- slurmd (primary)
- slurmd (backup)

Slurmd (optional)

Other clusters

Compute node daemons:
- slurmd
- slurmd
- slurmd
Running with SLURM

- Use “sbatch” (as “qsub” in Torque) to submit batch script or “salloc” (as “qsub -l” in Torque) to request interactive batch session.
- Use “srun” to launch parallel jobs (as with “aprune” with Torque/Moab or with hybrid SLURM)
- Most SLURM command options have two formats (long and short)
- Need to specify which shell to use for batch script.
- Environment is automatically imported (as “#PBS -V” in Torque)
- Lands on the submit directory (“cd $PBS_O_WORKDIR” not needed as in Torque)
- Batch script runs on the head compute node
- No need to repeat flags in the srun command if already defined in SBATCH keywords.
- srun flags overwrite SBATCH keywords
- 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. Advanced users can experiment more options such as –num_tasks_per_socket, –cpu_bind, --mem, etc.
- Hyperthreading is enabled by default. Jobs requesting more than 32 cores (MPI tasks * OpenMP threads) per node will use hyperthreads automatically.
Sample SLURM Batch Script

### Long command options

```bash
#!/bin/bash
#SBATCH
#SBATCH --partition=regular
#SBATCH --job-name=test
#SBATCH --account=mpccc
#SBATCH --nodes=2
#SBATCH --time=00:30:00

srun -n 16 ./mpi-hello
export OMP_NUM_THREADS=8
srun -n 8 -c 8 ./xthi
```

### Short command options

```bash
#!/bin/bash
#SBATCH
#SBATCH -p regular
#SBATCH -J test
#SBATCH -A mpccc
#SBATCH -N 2
#SBATCH -t 00:30:00

srun -n 16 ./mpi-hello
export OMP_NUM_THREADS=8
srun -n 8 -c 8 ./xthi
```

To submit a batch job:

% sbatch mytest.sl

Submitted batch job 15400
salloc and srun Example (interactive batch)

```
yunhe/> salloc -N 2 -p debug -t 30:00
salloc: Granted job allocation 16180
salloc: Waiting for resource configuration
salloc: Nodes nid00[408-409] are ready for job

yunhe@nid00408> export OMP_NUM_THREADS=8

yunhe@nid00408> srun --n 8 --c 8 ./xthi | sort -k4n -k6n
```

Hello from rank 0, thread 0, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 1, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 2, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 3, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 4, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 5, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 6, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 7, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 1, thread 0, on nid00408. (core affinity = 16-23,48-55)
Hello from rank 1, thread 1, on nid00408. (core affinity = 16-23,48-55)
...
Hello from rank 1, thread 6, on nid00408. (core affinity = 16-23,48-55)
Hello from rank 1, thread 7, on nid00408. (core affinity = 16-23,48-55)
...
Hello from rank 4, thread 0, on nid00409. (core affinity = 0-7,32-39)
Hello from rank 4, thread 1, on nid00409. (core affinity = 0-7,32-39)
...
Hello from rank 4, thread 6, on nid00409. (core affinity = 0-7,32-39)
Hello from rank 4, thread 7, on nid00409. (core affinity = 0-7,32-39)
...
# SLURM vs. Torque/Moab Keywords

<table>
<thead>
<tr>
<th>Description</th>
<th>#PBS</th>
<th>#SBATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>-q [queue]</td>
<td>--partition=[queue] or -p [queue]</td>
</tr>
<tr>
<td>Number of tasks</td>
<td>-n [count]</td>
<td>--ntasks=[count] or -n [count]</td>
</tr>
<tr>
<td>Node count</td>
<td>-l mppwidth=[count]*24</td>
<td>--nodes=[count] or -N [count]</td>
</tr>
<tr>
<td>Tasks per node</td>
<td>-l mppnnppn=[count]</td>
<td>--ntasks-per-node=[count]</td>
</tr>
<tr>
<td>CPUs per task</td>
<td>(-d option in aprun)</td>
<td>--cpus-per-task=[count] or -c [count]</td>
</tr>
<tr>
<td>Wall clock limit</td>
<td>-l walltime=[hh:mm:ss]</td>
<td>--time=[days-hh:mm:ss] or -t [hh:mm:ss]</td>
</tr>
<tr>
<td>Standard output</td>
<td>-o [file]</td>
<td>--output=[file] or -o [file]</td>
</tr>
<tr>
<td>Standard error</td>
<td>-e [file]</td>
<td>--error=[file] or -e [file]</td>
</tr>
<tr>
<td>Combine stdout/stderr</td>
<td>-j oe (both to stdout)</td>
<td>default behavior if no --error or -e</td>
</tr>
<tr>
<td>Event notification</td>
<td>-m abe</td>
<td>--mail-type=[events]</td>
</tr>
<tr>
<td>Email address</td>
<td>-M [address]</td>
<td>--mail-user=[address]</td>
</tr>
<tr>
<td>Job name</td>
<td>-N [name]</td>
<td>--job-name=[name] or -J [name]</td>
</tr>
<tr>
<td>Job restart</td>
<td>-r[y/n] (NERSC default is n)</td>
<td>requeue or –no-requeue (NERSC default)</td>
</tr>
<tr>
<td>Job arrays</td>
<td>-t [array_spec]</td>
<td>--array=array_spec or -a[array_spec]</td>
</tr>
</tbody>
</table>
# SLURM vs. Torque/Moab Environment Variables

<table>
<thead>
<tr>
<th>Description</th>
<th>Torque/Moab</th>
<th>SLURM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job id</td>
<td>$PBS_JOBID</td>
<td>$SLURM_JOB_ID</td>
</tr>
<tr>
<td>Job name</td>
<td>$PBS_NODENAME</td>
<td>$SLURM_JOB_NAME</td>
</tr>
<tr>
<td>Submit directory</td>
<td>$PBS_O_WORKDIR</td>
<td>$SLURM_SUBMIT_DIR</td>
</tr>
<tr>
<td>Node list</td>
<td>$PBS_NODEFILE</td>
<td>$SLURM_NODELIST</td>
</tr>
<tr>
<td>Host submitted from</td>
<td>$PBS_O_HOST</td>
<td>$SLURM_SUBMIT_HOST</td>
</tr>
<tr>
<td>Nodes allocated</td>
<td>$PBS_NUM_NODES</td>
<td>$SLURM_JOB_NUM_NODES</td>
</tr>
<tr>
<td>Number cores/nodes</td>
<td>$PBS_NUM_PPN</td>
<td>$SLURM_CPUS_ON_NODE</td>
</tr>
</tbody>
</table>
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
• **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
• **sqs**: NERSC custom queue display (*subject to change*)
### squeue and sinfo Examples

**yunhe> squeue**

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>ACCOUNT</th>
<th>NAME</th>
<th>PARTITION</th>
<th>QOS</th>
<th>NODES</th>
<th>TIME_LIMIT</th>
<th>TIME</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>16179</td>
<td>keskital</td>
<td>planck</td>
<td>reproc</td>
<td>debug</td>
<td>low</td>
<td>128</td>
<td>30:00</td>
<td>10:51</td>
<td>R</td>
</tr>
<tr>
<td>14645</td>
<td>psteinbr</td>
<td>m2078</td>
<td>1328f21b64</td>
<td>regular</td>
<td>normal</td>
<td>32</td>
<td>2:30:00</td>
<td>1:47:53</td>
<td>R</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15974</td>
<td>heikki</td>
<td>m1820</td>
<td>ipnonsa_1e</td>
<td>shared</td>
<td>normal</td>
<td>1</td>
<td>6:00:00</td>
<td>22:48</td>
<td>R</td>
</tr>
<tr>
<td>15975</td>
<td>heikki</td>
<td>m1820</td>
<td>ipnonsa_1e</td>
<td>shared</td>
<td>normal</td>
<td>1</td>
<td>6:00:00</td>
<td>22:29</td>
<td>R</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14734</td>
<td>psteinbr</td>
<td>m2078</td>
<td>1328f21b64</td>
<td>regular</td>
<td>normal</td>
<td>32</td>
<td>2:30:00</td>
<td>0:00</td>
<td>PD</td>
</tr>
<tr>
<td>14735</td>
<td>psteinbr</td>
<td>m2078</td>
<td>1328f21b64</td>
<td>regular</td>
<td>normal</td>
<td>32</td>
<td>2:30:00</td>
<td>0:00</td>
<td>PD</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15944</td>
<td>zuntz</td>
<td>des</td>
<td>spteg_disc</td>
<td>regular</td>
<td>normal</td>
<td>32</td>
<td>6:00:00</td>
<td>0:00</td>
<td>PD</td>
</tr>
<tr>
<td>15945</td>
<td>zuntz</td>
<td>des</td>
<td>spteg_bulg</td>
<td>regular</td>
<td>normal</td>
<td>32</td>
<td>6:00:00</td>
<td>0:00</td>
<td>PD</td>
</tr>
<tr>
<td>15952</td>
<td>zuntz</td>
<td>des</td>
<td>spteg_bulg</td>
<td>regular</td>
<td>normal</td>
<td>32</td>
<td>6:00:00</td>
<td>0:00</td>
<td>PD</td>
</tr>
<tr>
<td>14651</td>
<td>psteinbr</td>
<td>m2078</td>
<td>1328f21b64</td>
<td>regular</td>
<td>normal</td>
<td>1</td>
<td>2:30:00</td>
<td>1:47:44</td>
<td>CG</td>
</tr>
</tbody>
</table>

**yunhe> sinfo**

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>JOB_SIZE</th>
<th>TIMELIMIT</th>
<th>CPUS</th>
<th>S:C:T</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug*</td>
<td>up</td>
<td>1-infini</td>
<td>30:00</td>
<td>64</td>
<td>2:16:2</td>
<td>1478</td>
<td>allocated</td>
<td>nid[00024-00063,...]</td>
</tr>
<tr>
<td>debug*</td>
<td>up</td>
<td>1-infini</td>
<td>30:00</td>
<td>64</td>
<td>2:16:2</td>
<td>150</td>
<td>idle</td>
<td>nid[00408-00409,...]</td>
</tr>
<tr>
<td>regular</td>
<td>up</td>
<td>1-infini</td>
<td>12:00:00</td>
<td>64</td>
<td>2:16:2</td>
<td>1478</td>
<td>allocated</td>
<td>nid[00024-00063,...]</td>
</tr>
<tr>
<td>regular</td>
<td>up</td>
<td>1-infini</td>
<td>12:00:00</td>
<td>64</td>
<td>2:16:2</td>
<td>150</td>
<td>idle</td>
<td>nid[00408-00409,...]</td>
</tr>
<tr>
<td>realtime</td>
<td>down</td>
<td>1-infini</td>
<td>6:00:00</td>
<td>64</td>
<td>2:16:2</td>
<td>1478</td>
<td>allocated</td>
<td>nid[00024-00063,...]</td>
</tr>
<tr>
<td>realtime</td>
<td>down</td>
<td>1-infini</td>
<td>6:00:00</td>
<td>64</td>
<td>2:16:2</td>
<td>150</td>
<td>idle</td>
<td>nid[00408-00409,...]</td>
</tr>
<tr>
<td>shared</td>
<td>up</td>
<td>1</td>
<td>12:00:00</td>
<td>64</td>
<td>2:16:2</td>
<td>40</td>
<td>allocated</td>
<td>nid[00188-00191,...]</td>
</tr>
</tbody>
</table>

There are many other options. See man page.
scontrol Example

yunhe>  scontrol show partition debug
PartitionName=debug
  AllowGroups=ALL  AllowAccounts=ALL  AllowQos=ALL
  AllocNodes=ALL  Default=YES  QoS=part_debug
  DefaultTime=00:10:00  DisableRootJobs=YES  ExclusiveUser=NO  GraceTime=0  Hidden=NO
  MaxNodes=UNLIMITED  MaxTime=00:30:00  MinNodes=1  LLN=NO  MaxCPUsPerNode=UNLIMITED
  Nodes=nid0[0024-0063,0080-0083,….2128-2175,2192-2239,2256-2303]
  Priority=2000  RootOnly=NO  ReqResv=NO  Shared=EXCLUSIVE  PreemptMode=REQUEUE
  State=UP  TotalCPUs=104192  TotalNodes=1628  SelectTypeParameters=N/A
  DefMemPerNode=UNLIMITED  MaxMemPerNode=124928

yunhe>  scontrol show job 16178
JobId=16178  JobName=mpi-hello.sl
  UserId=yunhe(18456)  GroupId=yunhe(1018456)
  Priority=834180  Nice=0  Account=mpccc  QOS=normal_regular_4
  RunTime=00:00:00  TimeLimit=00:30:00  TimeMin=N/A
  NodeList=nid00[408-409]
  BatchHost=nid00408
  NumNodes=2  NumCPUs=128  CPUs/Task=1  ReqB:S:C:T=0:0:*:*  
  TRES=cpu=128,mem=249856,node=2
  Socks/Node=*  NtasksPerN:B:S:C=2:0:*:*  CoreSpec=*  
  Command=/global/u1/y/yunhe/shared/mpi-hello.sl
  WorkDir=/global/u1/y/yunhe/shared
  StdErr=/global/u1/y/yunhe/shared/slurm-16178.out
  StdOut=/global/u1/y/yunhe/shared/slurm-16178.out
sacct Example

```
yunhe> sacct -u yunhe --start=11/3/15T9:00 --end=1/3/15T15:00 -o JobID,elapsed

<table>
<thead>
<tr>
<th>JobID</th>
<th>Elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>13230</td>
<td>00:17:41</td>
</tr>
<tr>
<td>13230.0</td>
<td>00:18:08</td>
</tr>
<tr>
<td>13230.1</td>
<td>00:00:15</td>
</tr>
<tr>
<td>13230.2</td>
<td>00:08:17</td>
</tr>
<tr>
<td>13234</td>
<td>00:05:43</td>
</tr>
<tr>
<td>13234.0</td>
<td>00:06:13</td>
</tr>
<tr>
<td>13234.1</td>
<td>00:04:44</td>
</tr>
<tr>
<td>13237</td>
<td>00:30:23</td>
</tr>
<tr>
<td>13237.0</td>
<td>00:30:50</td>
</tr>
<tr>
<td>13237.1</td>
<td>00:00:14</td>
</tr>
<tr>
<td>13237.2</td>
<td>00:00:35</td>
</tr>
<tr>
<td>13237.3</td>
<td>00:00:03</td>
</tr>
</tbody>
</table>
```

There are many other options. See man page.
SLURM Features on Cori

- The Cluster Compatibility Mode (CCM) capability is implemented as an SBATCH command line option (--ccm), which enables SSH to compute nodes. No separate CCM partition is needed. CCM jobs can run in any partition now.

- Since sbatch or salloc lands on a compute node, applications such as “matlab” can be launched directly without CCM support.

- The native “shared” partition on Cori is not implemented as the Edison “serial” queue via CCM. It allows multiple users to share one node. Users can request more than 1 slot (either by tasks or memory) and be charged accordingly.

- Potential to expand or shrink job size at run time.

- Another batch server will be implemented to run “xfer”, “bigmem”, “workflow” jobs on one or more login nodes. Users can submit to either batch server in a single batch script.

- We will experiment batch job priority with combination of job size, queue wait time and fare share initially.
# Current Batch Configuration

<table>
<thead>
<tr>
<th>Partition</th>
<th>Nodes</th>
<th>Physical Cores</th>
<th>Max Walltime</th>
<th>Relative Priority</th>
<th>Charge Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>1-64</td>
<td>1-2,048</td>
<td>30 min</td>
<td>2</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>65-128</td>
<td>2,049-4,096</td>
<td>30 min</td>
<td>3</td>
<td>Free</td>
</tr>
<tr>
<td>regular</td>
<td>1-2</td>
<td>1-64</td>
<td>12 hrs</td>
<td>4</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>3-256</td>
<td>65-8,192</td>
<td>6 hrs</td>
<td>4</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>257-512</td>
<td>8,193-16,384</td>
<td>6 hrs</td>
<td>4</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>513-1,024</td>
<td>16,385-32,768</td>
<td>4 hrs</td>
<td>4</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>1,025-1,428</td>
<td>32,769-45,696</td>
<td>2 hrs</td>
<td>4</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>1,429-1,628</td>
<td>45,697-52,096</td>
<td>2 hrs</td>
<td>4</td>
<td>Free</td>
</tr>
<tr>
<td>shared</td>
<td>1</td>
<td>1-32</td>
<td>12 hrs</td>
<td>4</td>
<td>Free</td>
</tr>
<tr>
<td>realtime</td>
<td>1-32</td>
<td>1-1,024</td>
<td>6 hrs</td>
<td>1</td>
<td>Free</td>
</tr>
</tbody>
</table>
Current Batch Policies

- Users can request modification to job priority by adding "#SBATCH --qos=premium" (disabled during free charging time) or "#SBATCH --qos=low" or "$SBATCH --qos=scavenger" in the batch script.
- Overall job priorities are a combination of partition, QOS, queue wait time, job size, wall time request, and fair share.
- There are no specific run limits or idle limits per job type/size category per user (except 1 debug job per user). Instead, we are experimenting with a RunMins limit (total remaining CPU minutes of all running jobs) on the per user/repo base.
- Debug jobs requesting more than 64 nodes have lower priority to discourage using the entire system for small number of jobs. No more than 128 nodes can be used for all debug jobs in the system at any time.
- Relative priorities for different sizes of the “regular” jobs will be adjusted in the future.
- Jobs from different users can share a node to run parallel or serial jobs in the "shared" partition. The maximum number of nodes a single job can use is 1. There are a total of 40 nodes in the system can be used for "shared" jobs.
- The "realtime" partition usage is by special permission only. There are a total of 32 nodes in the system can be used for "realtime" jobs. "realtime" jobs can choose to share a node by adding "#SBATCH --shared" in the batch script.
Summary

- SLURM provides equivalent or similar functionality with Torque/Moab.
- `srun` provides equivalent or similar process and thread affinity with `aprun`.
- Please let us know if you have an advanced or complicated workflow, and anticipate potential porting issues. We can work with you to migrate your scripts.
- Batch configurations are still subject to tunings and modifications before the system is in full production.
Documentations

• SchedMD web page:
  – http://www.schedmd.com/
• Running Jobs on Cori
  – https://www.nersc.gov/users/computational-systems/cori/running-jobs/
• Man pages for slurm, sbatch, salloc, squeue, sinfo, sacct, scontrol, scancel, etc.
• Torque/Moab vs. SLURM Comparisons
• Running jobs on Babbage using SLURM:
• Running jobs on Edison’s test system (Alva) with native SLURM