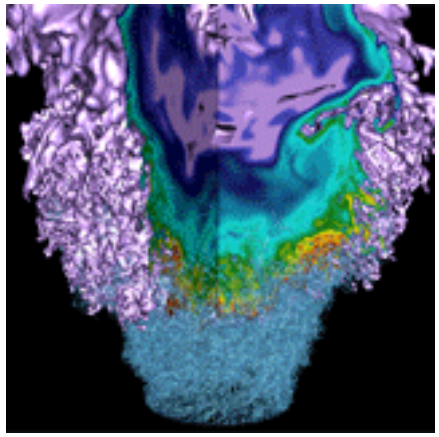
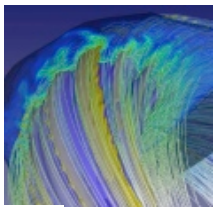
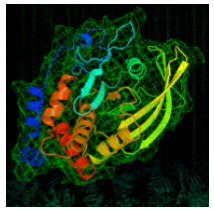
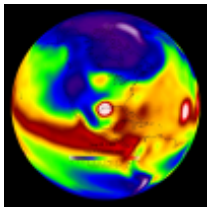
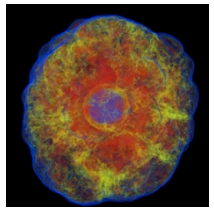
SLURM Resource Manager is Coming to NERSC

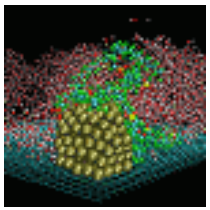












Helen He

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.

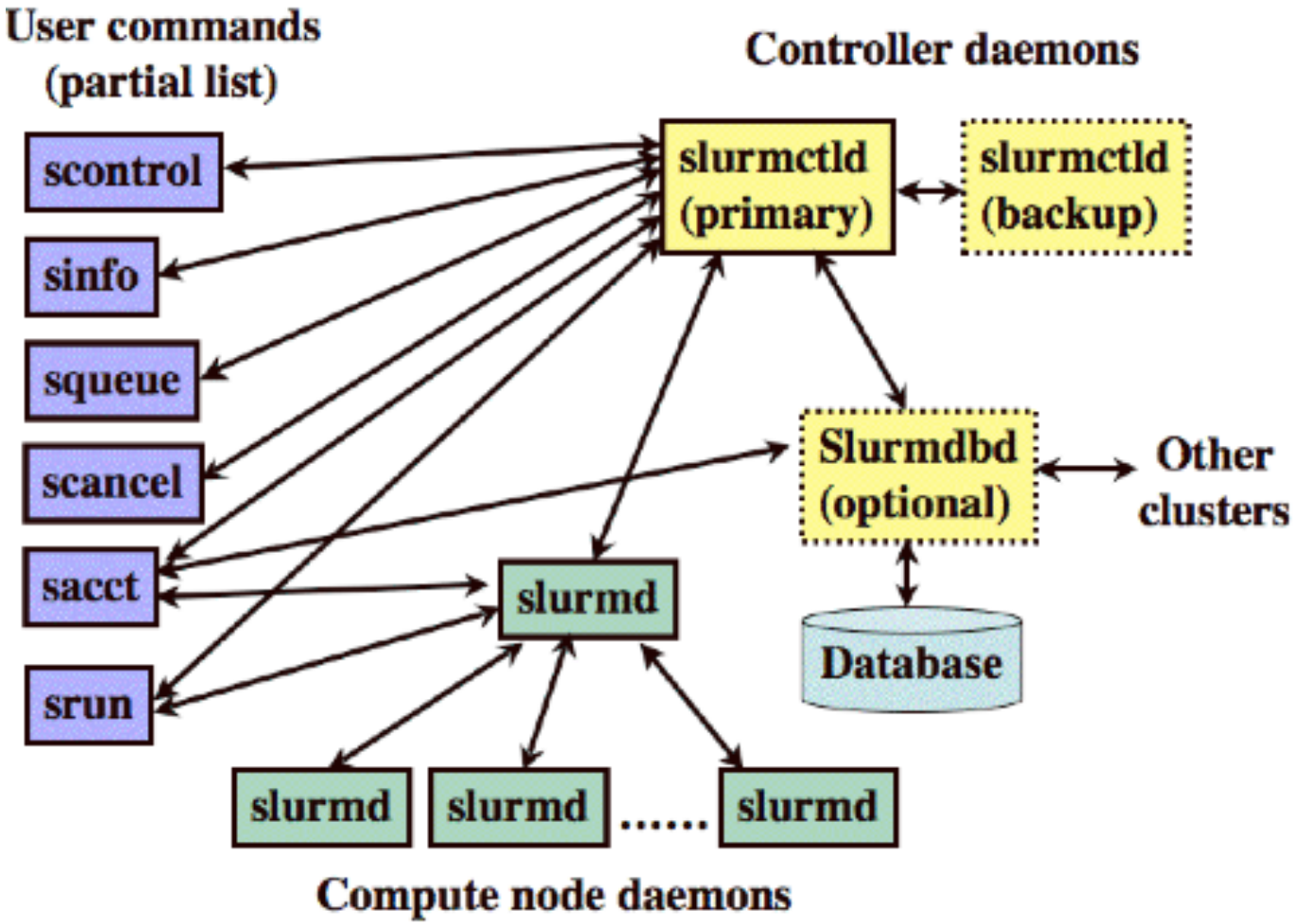
NERSC's Plan to Adopt SLURM

- 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



Running with SLURM

- Use “sbatch” (as “qsub” in Torque) to submit batch script or “salloc” (as “qsub -I” in Torque) to request interactive batch session.
- Use “srun” to launch parallel jobs (as with “aprun” 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

```
#!/bin/bash -l

#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

```
#!/bin/bash -l

#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

Description	#PBS	#SBATCH
Queue	-q [queue]	--partition=[queue] or -p [queue]
Number of tasks	-n [count]	--ntasks=[count] or -n [count]
Node count	-l mppwidth=[count]*24	--nodes=[count] or -N [count]
Tasks per node	-l mppnppn=[count]	--ntasks-per-node=[count]
CPUs per task	(-d option in aprun)	--cpus-per-task=[count] or -c [count]
Wall clock limit	-l walltime=[hh:mm:ss]	--time=[days-hh:mm:ss] or -t [hh:mm:ss]
Standard output	-o [file]	--output=[file] or -o [file]
Standard error	-e [file]	--error=[file] or -e [file]
Combine stdout/stderr	-j oe (both to stdout)	default behavior if no --error or -e
Event notification	-m abe	--mail-type=[events]
Email address	-M [address]	--mail-user=[address]
Job name	-N [name]	--job-name=[name] or -J [name]
Account to charge	-A [account]	--account=[account] or -A [account]
Job restart	-r[y/n] (NERSC default is n)	--requeue or –no-requeue (NERSC default)
Job dependency	-W depend=[state:jobid]	--depend=[state:jobid] or -d [state:jobid]
Job arrays	-t [array_spec]	--array=array_spec or -a[array_spec]

SLURM vs. Torque/Moab Environment Variables

Description	Torque/Moab	SLURM
Job id	$PBS_JOBID	$SLURM_JOB_ID
Job name	$PBS_NODENAME	$SLURM_JOB_NAME
Submit directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR
Node list	$PBS_NODEFILE	$SLURM_NODELIST
Host submitted from	$PBS_O_HOST	$SLURM_SUBMIT_HOST
Nodes allocated	$PBS_NUM_NODES	$SLURM_JOB_NUM_NODES
Number cores/nodes	$PBS_NUM_PPN	$SLURM_CPUS_ON_NODE

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

```
JOBID       USER  ACCOUNT       NAME  PARTITION     QOS NODES    TIME_LIMIT        TIME    ST
16179    keskital   planck     reproc      debug     low   128         30:00       10:51     R
14645    psteinbr    m2078 l328f21b64    regular  normal    32       2:30:00     1:47:53     R
...
15974      heikki    m1820 ipnonsa_1e     shared  normal     1       6:00:00       22:48     R
15975      heikki    m1820 ipnonsa_1e     shared  normal     1       6:00:00       22:29     R
...
14734    psteinbr    m2078 l328f21b64    regular  normal    32       2:30:00        0:00    PD
14735    psteinbr    m2078 l328f21b64    regular  normal    32       2:30:00        0:00    PD
...
15944       zuntz      des spteg_disc    regular  normal    32       6:00:00        0:00    PD
15945       zuntz      des spteg_bulg    regular  normal    32       6:00:00        0:00    PD
15952       zuntz      des spteg_bulg    regular  normal    32       6:00:00        0:00    PD
14651    psteinbr    m2078 l328f21b64    regular  normal     1       2:30:00     1:47:44    CG
```

yunhe> sinfo

```
PARTITION AVAIL JOB_SIZE  TIMELIMIT   CPUS  S:C:T   NODES STATE      NODELIST
debug*    up    1-infini      30:00     64 2:16:2    1478 allocated  nid[00024-00063,…]
debug*    up    1-infini      30:00     64 2:16:2     150 idle       nid[00408-00409,…]
regular   up    1-infini   12:00:00     64 2:16:2    1478 allocated  nid[00024-00063,…]
regular   up    1-infini   12:00:00     64 2:16:2     150 idle       nid[00408-00409,…]
realtime  down  1-infini    6:00:00     64 2:16:2    1478 allocated  nid[00024-00063,…]
realtime  down  1-infini    6:00:00     64 2:16:2     150 idle       nid[00408-00409,…]
shared    up    1          12:00:00     64 2:16:2      40 allocated  nid[00188-00191,…]
```

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
   SubmitTime=2015-11-06T05:48:11 EligibleTime=2015-11-06T05:48:11
   StartTime=2015-11-06T05:48:37 EndTime=2015-11-06T06:18:37
   …
   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

JobID	Elapsed
13230	00:17:41
13230.0	00:18:08
13230.1	00:00:15
13230.2	00:08:17
13234	00:05:43
13234.0	00:06:13
13234.1	00:04:44
13237	00:30:23
13237.0	00:30:50
13237.1	00:00:14
13237.2	00:00:35
13237.3	00:00:03

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

Partiton	Nodes	Physical Cores	Max Walltime	Relative Priority	Charge Factor
debug	1-64	1-2,048	30 min	2	Free
	65-128	2,049-4,096	30 min	3	Free
regular	1-2	1-64	12 hrs	4	Free
	3-256	65-8,192	6 hrs	4	Free
	257-512	8,193-16,384	6 hrs	4	Free
	513-1,024	16,385-32,768	4 hrs	4	Free
	1,025-1,428	32,769-45,696	2 hrs	4	Free
	1,429-1,628	45,697-52,096	2 hrs	4	Free
shared	1	1-32	12 hrs	4	Free
realtime	1-32	1-1,024	6 hrs	1	Free

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**
  - https://www.nersc.gov/users/computational-systems/cori/running-jobs/for-edison-users/torque-moab-to-slurm-transition-guide/
- **Running jobs on Babbage using SLURM:**
  - https://www.nersc.gov/users/computational-systems/testbeds/babbage/running-jobs-under-slurm-on-babbage/
- **Running iobs on Edison's test system (Alva) with native SLURM**
  - https://www.nersc.gov/users/computational-systems/edison/alva-test-and-development-system-for-edison/#toc-anchor-7