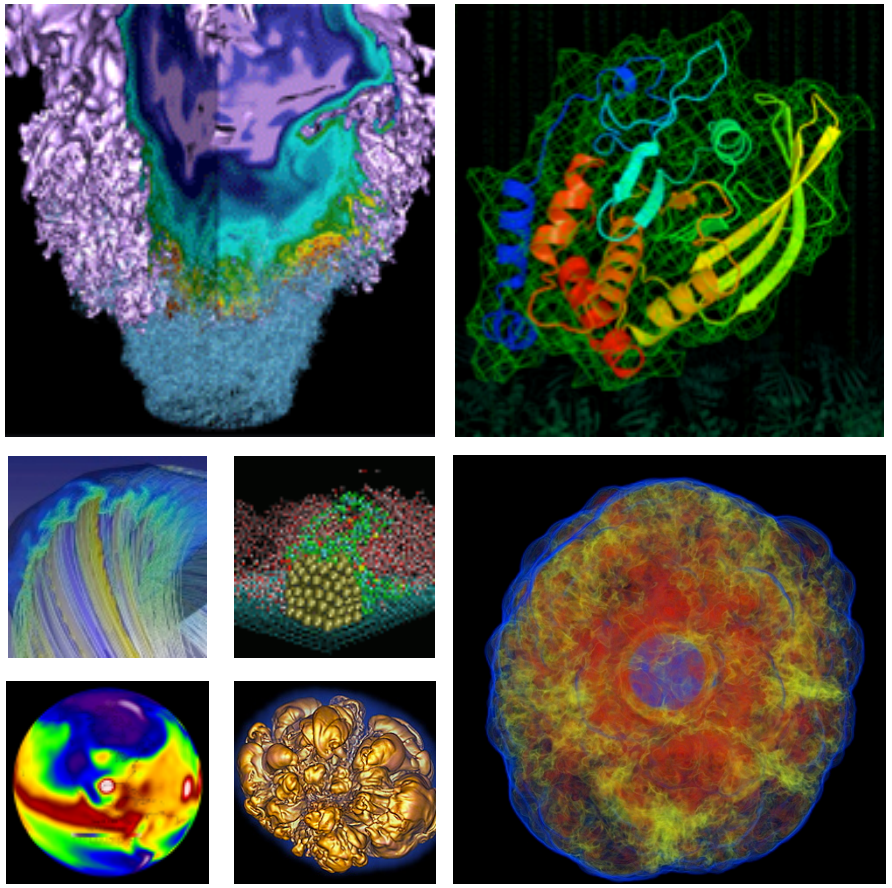
Running Jobs at Wang Hall



Daniel Udwary

NERSC Data Science Engagement Group

February 3, 2016

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Outline

- Genepool move logistics
- Differences between Crays and Genepool
- Cori and Edison architecture and configurations
- Intro to SLURM

Why am I running this training session?

- **During the Mendel move (next week!), we will have a period of reduced Genepool compute availability**
- **We want to encourage more JGI compute work on NERSC's flagship supercomputers, when it makes sense**
  - Last year, used less than half of CPU-hour allocation
- **NERSC wants to know what it can do to better enable bioinformatics work on those machines, and identify where future problems might lie**
- **Genepool may move to SLURM in the future**

NERSC has moved to a new building

- **All systems must move from Oakland to Berkeley**

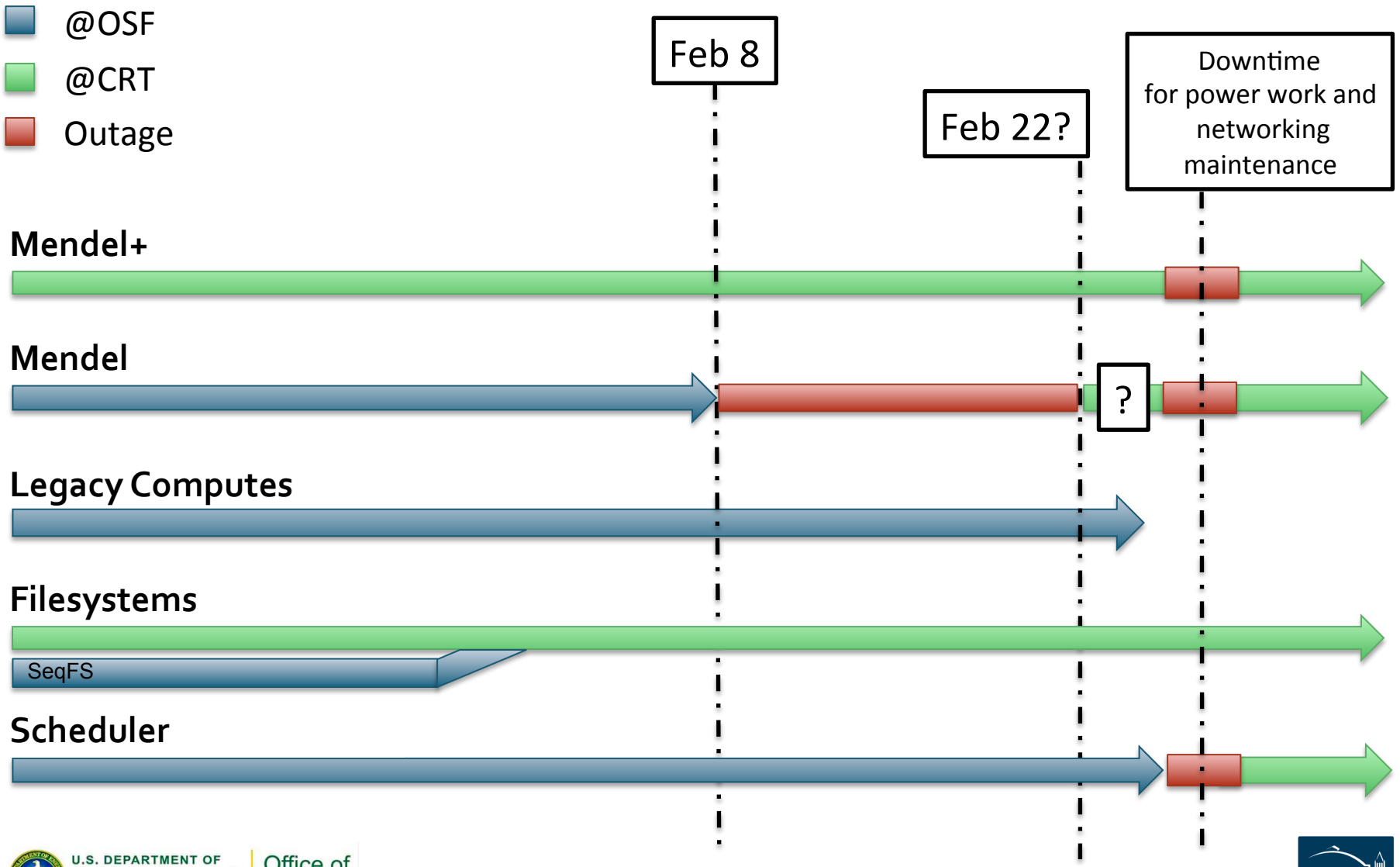
Resources at Wang Hall (aka CRT)

- **New Mendel+ nodes**
- **New login nodes (genepool13 and genepool14)**
- **All filesystems (almost…)**
- **Cori**
- **Edison**

Still at OSF:

- Old Mendel nodes – moving starting Feb 8
- Legacy Genepool nodes – to be shutdown ~Feb 22
- Tape archive – No plan to move (yet)

Move Schedule – Current Plan



Key Differences Between Cori/Edison and Genepool

Cori and Edison

- Generally large, multi-node jobs
- Jobs are charged
- Wait time until job start measured in days
- Users generally compile and install their own software – few modules
- SLURM

Genepool

- Many small, single node (or even single-CPU) jobs
- No job charging
- Wait time measured in hours, if not minutes
- Awesome JGI consultants manage bioinformatics software as modules
- UGE

Basics of NERSC Cray architecture

- **Cori Phase I**
  - Cray XC
  - 1630 nodes
  - 128 GB memory per node
  - 32 cores per node
    - (2x16 core 2.3 GHz Haswell)
- **Edison**
  - Cray XC30
  - 5576 nodes
  - 64 GB memory per node
  - 24 cores per node
    - (2x12 core 2.4 GHz Ivy Bridge)
- **Cori Phase II**
  - >9300 nodes
  - Knights Landing CPUs

Edison Queue Structure

https://www.nersc.gov/users/computational-systems/edison/running-jobs/queues-and-policies/

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

So, use Edison for large parallel jobs using >682 nodes

Cori queue structure

- **https://www.nersc.gov/users/computational-systems/cori/running-jobs/queues-and-policies/**

Partition	Nodes	Physical Cores	Max Walltime per Job	QOS	Max Number of Running Jobs	Max Total Num Nodes per User for Running Jobs	Number of Jobs per User Submit Limit	Relative Priority	Charge Factor
debug	1-112	1-3,072	30 min	normal	1	112	5	3	1.0
regular	1-2	1-64	48 hrs	normal	50	100	200	4	1.0
				premium	10	100	40	2	2.0
				low	50	100	200	5	0.5
				scavenger	10	100	40	6	0
	3-512	65-16,384	36 hrs	normal	10	512	50	4	1.0
				premium	2	512	10	2	2.0
				low	10	512	50	5	0.5
				scavenger	2	512	10	6	0
	513-1,420	16,385-45,440	12 hrs	normal	1	1,420	4	4	1.0
				premium	1	1,420	2	2	2.0
				low	1	1,420	4	5	0.5
				scavenger	1	1,420	2	6	0.0
shared	1	1-16	48 hrs	normal	500	2,500	4	--	1.0
realtime	custom	custom	custom	custom	custom	--	1	1 (special permission)	--
xfer	1	1	12 hrs	--	--	--	1	--	--

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.
- Cori installed with SLURM, and Edison switched last Nov, after its' move

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.

SLURM User Commands

• **sbatch**	**qsub**	submit a batch script
• **salloc**	**qlogin**	request an interactive session
• **scancel**	**qdel**	delete a batch job
• **scontrol hold**	**qhold**	hold a job
• **scontrol release**	**qrls**	release a job
• **sacct**	**qacct**	display job accounting data
• **sqs**	**qs**	NERSC custom queue display

Running with SLURM

- Use “sbatch” (as “qsub” in UGE) to submit batch script or “salloc” (as “qlogin” in UGE) to request interactive batch session.
- Need to specify which shell to use for batch script.
- Environment is automatically imported (as “qsub -V” in UGE)
- Lands on the submit directory
- Batch script runs on the head compute node
- No need to repeat flags in the srun command if already defined in SBATCH keywords.
- Hyperthreading is enabled by default. Jobs requesting more than 32 cores (MPI tasks * OpenMP threads) per node will use hyperthreads automatically.

Running with SLURM continued

- Use “srun” to launch parallel jobs (as with “aprun” with Torque/Moab)
- 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.

http://slurm.schedmd.com/rosetta.pdf

28-Apr-2013

User Commands	PBS/Torque	Slurm	LSF	SGE	LoadLeveler
Job submission	qsub [script_file]	sbatch [script_file]	bsub [script_file]	qsub [script_file]	llsubmit [script_file]
Job deletion	qdel [job_id]	scancel [job_id]	bkill [job_id]	qdel [job_id]	llcancel [job_id]
Job status (by job)	qstat [job_id]	squeue [job_id]	bjobs [job_id]	qstat -u * [-j job_id]	llq -u [username]
Job status (by user)	qstat -u [user_name]	squeue -u [user_name]	bjobs -u [user_name]	qstat [-u user_name]	llq -u [user_name]
Job hold	qhold [job_id]	scontrol hold [job_id]	bstop [job_id]	qhold [job_id]	llhold -r [job_id]
Job release	qrls [job_id]	scontrol release [job_id]	bresume [job_id]	qrls [job_id]	llhold -r [job_id]
Queue list	qstat -Q	squeue	bqueues	qconf -sql	llclass
Node list	pbsnodes -l	sinfo -N OR scontrol show nodes	bhosts	qhost	llstatus -L machine
Cluster status	qstat -a	sinfo	bqueues	qhost -q	llstatus -L cluster
GUI	xpbsmon	sview	xlsf OR xlsbatch	qmon	xload
Environment	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**
Job ID	$PBS_JOBID	$SLURM_JOBID	$LSB_JOBID	$JOB_ID	$LOAD_STEP_ID
Submit Directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR	$LSB_SUBCWD	$SGE_O_WORKDIR	$LOADL_STEP_INITDIR
Submit Host	$PBS_O_HOST	$SLURM_SUBMIT_HOST	$LSB_SUB_HOST	$SGE_O_HOST	
Node List	$PBS_NODEFILE	$SLURM_JOB_NODELIST	$LSB_HOSTS/LSB_MCPU_HOST	$PE_HOSTFILE	$LOADL_PROCESSOR_LIST
Job Array Index	$PBS_ARRAYID	$SLURM_ARRAY_TASK_ID	$LSB_JOBINDEX	$SGE_TASK_ID	
Job Specification	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**
Script directive	#PBS	#SBATCH	#BSUB	#$	#@
Queue	-q [queue]	-p [queue]	-q [queue]	-q [queue]	class=[queue]
Node Count	-l nodes=[count]	-N [min[-max]]	-n [count]	N/A	node=[count]
CPU Count	-l ppn=[count] OR -l mppwidth=[PE_count]	-n [count]	-n [count]	-pe [PE] [count]	
Wall Clock Limit	-l walltime=[hh:mm:ss]	-t [min] OR -t [days-hh:mm:ss]	-W [hh:mm:ss]	-l h_rt=[seconds]	wall_clock_limit=[hh:mm:ss]
Standard Output FIle	-o [file_name]	-o [file_name]	-o [file_name]	-o [file_name]	output=[file_name]
Standard Error File	-e [file_name]	e [file_name]	-e [file_name]	-e [file_name]	error=[File_name]
Combine stdout/err	-j oe (both to stdout) OR -j eo (both to stderr)	(use -o without -e)	(use -o without -e)	-j yes	
Copy Environment	-V	--export=[ALL | NONE | variables]		-V	environment=COPY_ALL
Event Notification	-m abe	--mail-type=[events]	-B or -N	-m abe	notification=start|error|complete|never|always
Email Address	-M [address]	--mail-user=[address]	-u [address]	-M [address]	notify_user=[address]
Job Name	-N [name]	--job-name=[name]	-J [name]	-N [name]	job_name=[name]
Job Restart	-r [y|n]	--requeue OR --no-requeue (NOTE: configurable default)	-r	-r [yes|no]	restart=[yes|no]
Working Directory	N/A	--workdir=[dir_name]	(submission directory)	-wd [directory]	initialdir=[directory]
Resource Sharing	-l naccesspolicy=singlejob	--exclusive OR--shared	-x	-l exclusive	node_usage=not_shared
Memory Size	-l mem=[MB]	--mem=[mem][M|G|T] OR --mem-per-cpu=[mem][M|G|T]	-M [MB]	-l mem_free=[memory][K|M|G]	requirements=(Memory >= [MB])
Account to charge	-W group_list=[account]	--account=[account]	-P [account]	-A [account]	
Tasks Per Node	-l mppnppn [PEs_per_node]	--tasks-per-node=[count]		(Fixed allocation_rule in PE)	tasks_per_node=[count]
CPUs Per Task		--cpus-per-task=[count]			
Job Dependency	-d [job_id]	--depend=[state:job_id]	-w [done | exit | finish]	-hold_jid [job_id | job_name]	
Job Project		--wckey=[name]	-P [name]	-P [name]	
Job host preference		--nodelist=[nodes] AND/OR --exclude=[nodes]	-m [nodes]	-q [queue]@[node] OR -q [queue]@@[hostgroup]	
Quality Of Service	-l qos=[name]	--qos=[name]			
Job Arrays	-t [array_spec]	--array=[array_spec] (Slurm version 2.6+)	J "name[array_spec]"	-t [array_spec]	
Generic Resources	-l other=[resource_spec]	--gres=[resource_spec]		-l [resource]=[value]	
Licenses		--licenses=[license_spec]	-R "rusage[license_spec]"	-l [license]=[count]	
Begin Time	-A "YYYY-MM-DD HH:MM:SS"	--begin=YYYY-MM-DD[THH:MM[:SS]]	-b[[year:][month:]daty:]hour:minute	-a [YYMMDDhhmm]	

SLURM Task arrays

Task arrays work similarly to UGE

- **sbatch --array=1-100**
  - Would start a 100 task job array

- Job arrays will have two additional environment variables set:
  - $SLURM_ARRAY_JOB_ID will be set to the first job ID of the array.
  - $SLURM_ARRAY_TASK_ID will be set to the job array index value.

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

SLURMmary

- SLURM provides equivalent or similar functionality with Torque/Moab and UGE.
- 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


NERSC