NERSC Site Report
One year of Slurm

Douglas Jacobsen
NERSC

SLURM User Group 2016

NERSC Vital Statistics

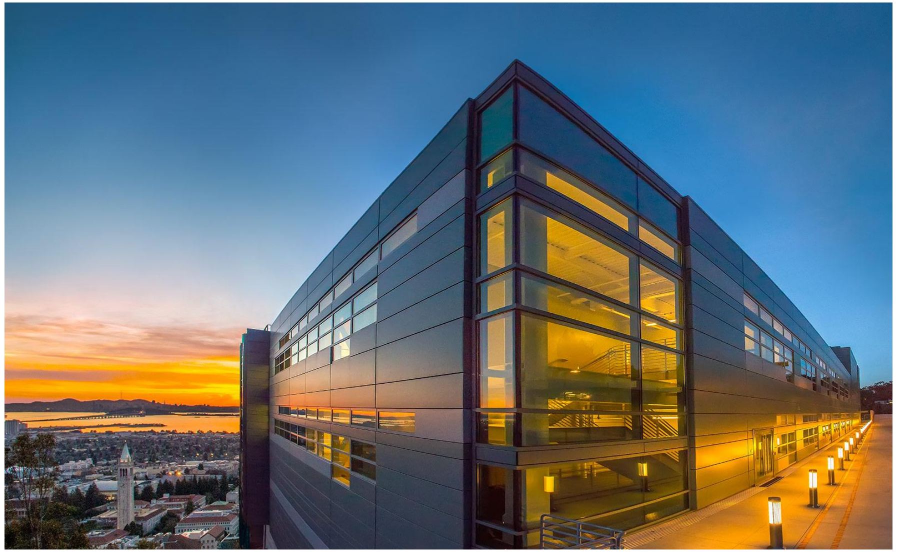
- 860 active projects
- 7,750 active users
- 700+ codes both established and in-development
- migrated production capability systems to Slurm 09/2015 - 01/2016

NERSC is part of the Lawrence Berkeley National Laboratory

- Recently moved from Oakland, CA to Berkeley, CA

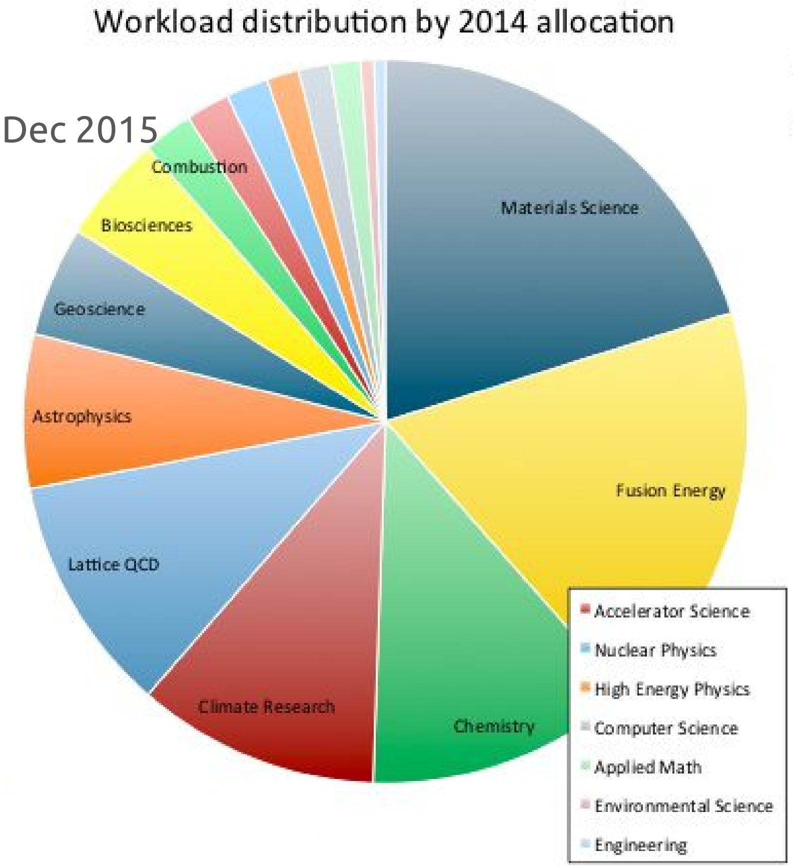
NERSC operates multiple supercomputers for the U.S. Department of Energy

Computer time is allocated by DOE for open science research projects funded by DOE.



NERSC Vital Statistics

- edison XC30, 5586 ivybridge nodes
  - Moved edison from Oakland, CA to Berkeley, CA in Dec 2015
  - 24 cores per node, 134,064 cores total
  - 64 GB per node, 2.6GB/core, 350TB total
  - Primarily used for large capability jobs
  - Small - midrange as well



NERSC Vital Statistics

- cori phase 1 XC40, 1,628 haswell nodes
  - DataWarp BurstBuffer
  - realtime jobs for experimental facilities
  - massive quantities of serial jobs
  - regular HPC workload too
  - shifter for Linux Containers

cori XC40

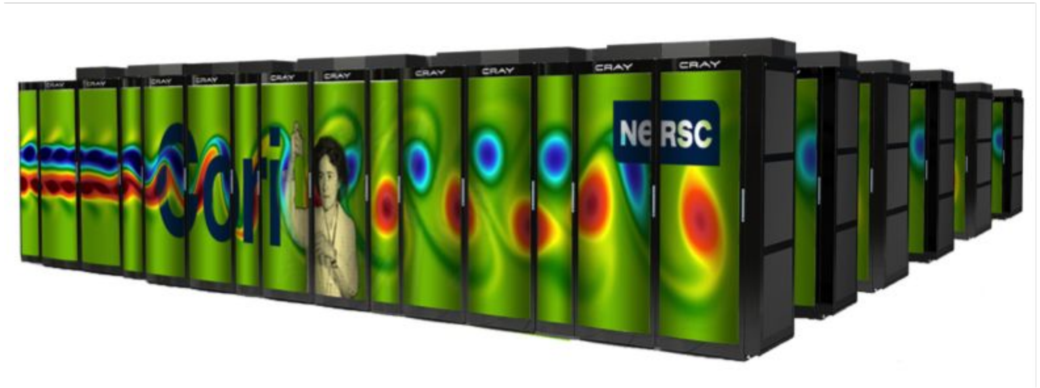
2,000 haswell nodes 64,000 cores

9,308 knl nodes 632,944 cores

keep all phase 1 workload

add massive KNL development workload

greatly expand HPC workload

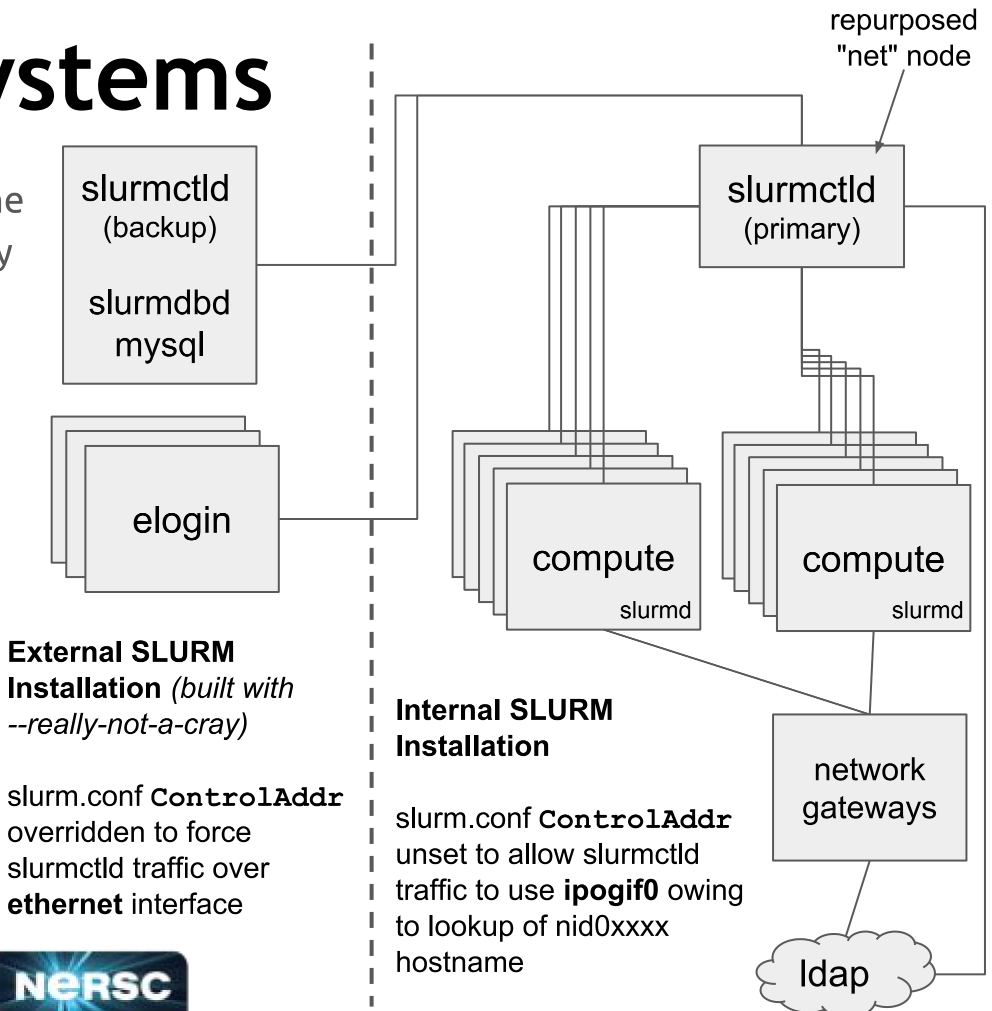


SLURM on Cray Systems

"Natively" running SLURM replaces some of the ALPS resource manager functionalities on Cray Systems

Why native?

1. Enables direct support for serial jobs
2. Simplifies operation by easing prolog/epilog access to compute nodes
3. Simplifies user experience
   a. No shared batch-script nodes
   b. Similar to other cluster systems
4. Enables new features and functionality on existing systems
5. Creates a "platform for innovation"



External SLURM Installation *(built with --really-not-a-cray)*

slurm.conf `ControlAddr` overridden to force slurmctld traffic over **ethernet** interface

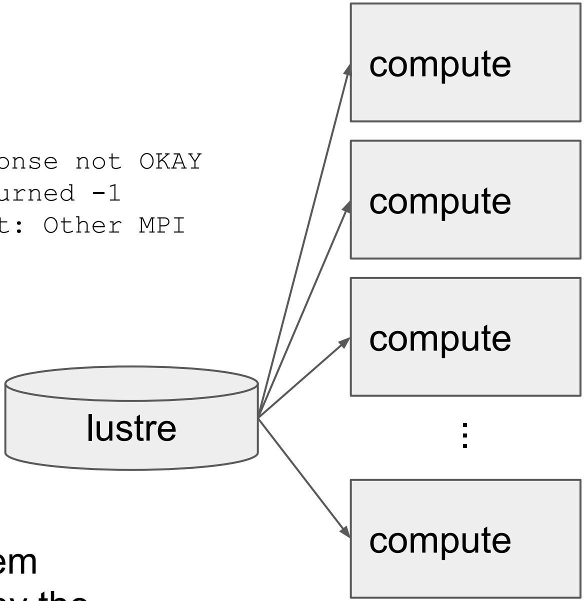
Internal SLURM Installation

slurm.conf `ControlAddr` unset to allow slurmctld traffic to use **ipogif0** owing to lookup of nid0xxxx hostname

Scaling Up

Challenge: Small and mid-scale jobs work great!
When MPI ranks exceed ~50,000 sometimes users get:

```
Sun Jan 24 04:51:29 2016: [unset]:_pmi_alps_get_apid:alps response not OKAY
Sun Jan 24 04:51:29 2016: [unset]:_pmi_init:_pmi_alps_init returned -1
[Sun Jan 24 04:51:30 2016] [c3-0c2s9n3] Fatal error in MPI_Init: Other MPI
error, error stack:
MPIR_Init_thread(547):
MPID_Init(203).......: channel initialization failed
MPID_Init(584).......: PMI2 init failed: 1
<repeat ad nauseum for every rank>
```



Workaround: Increase PMI timeout from 60s to something bigger (app env): `PMI_MMAP_SYNC_WAIT_TIME=300`

Problem: srun directly execs the application from the hosting filesystem location. FS cannot deliver the application at scale. aprun would copy the executable to in-memory filesystem by default.

Solution: 15.08 srun feature merging sbcast and srun

```
srun --bcast=/tmp/a.out ./mpi/a.out
```

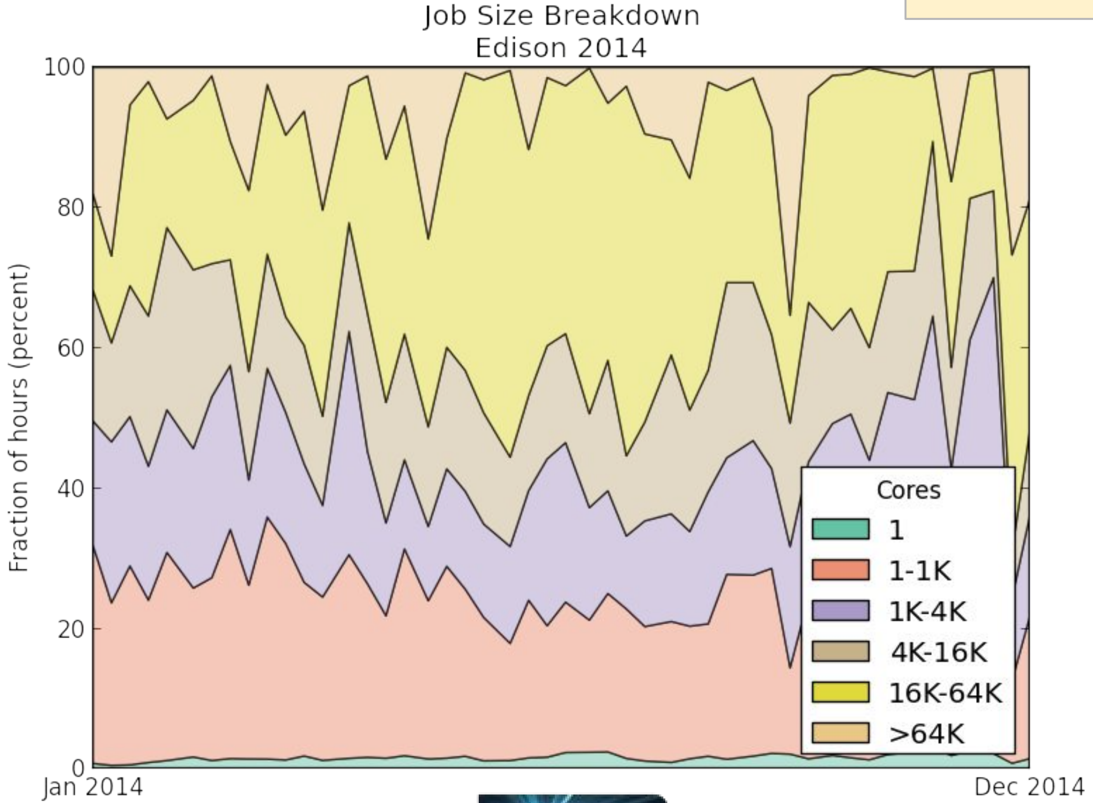
slurm 16.05 adds `--compress` option to deliver executable in similar time as aprun

Other scaling topics:

- srun ports for stdout/err
- rsip port exhaustion
- slurm.conf TreeWidth
- Backfill tuning

Scheduling

"NERSC users run applications at every scale to conduct their research."



Source: Brian Austin, NERSC

Scheduling

cori

- "shared" partition
  - Up to 32 jobs per node
  - HINT: set --gres=craynetwork:0 in job_submit.lua for shared jobs
  - allow users to submit 10,000 jobs with up to 1,000 concurrently running
- "realtime" partition
  - Jobs must start within 2 minutes
  - Per-project limits implemented using QOS
  - Top priority jobs + exclusive access to small number of nodes (92% utilized)
- burstbuffer QOS gives constant priority boost to burst buffer jobs

edison

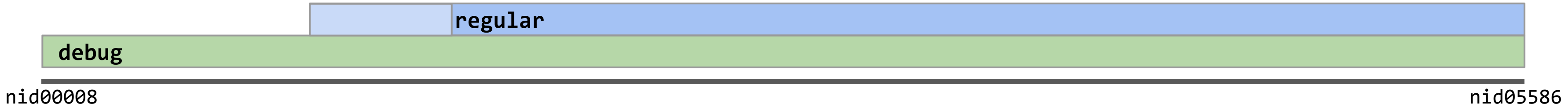
- big job metric - need to always be running at least one "large" job (>682 nodes)
  - Give priority boost + discount

cori+edison

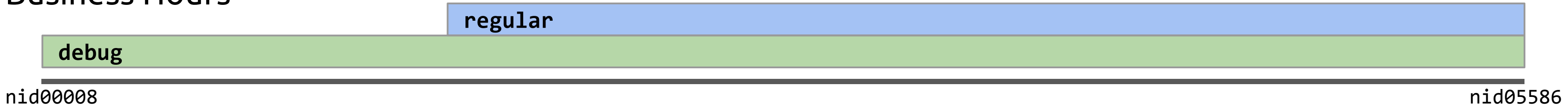
- debug partition
  - delivers debug-exclusive nodes
  - more exclusive nodes during business hours
- regular partition
  - Highly utilized workhorse
- low and premium QOS
  - accessible in most partitions
- scavenger QOS
  - Once a user account balance drops below zero, all jobs automatically put into scavenger. Eligible for all partitions except realtime

Scheduling - How Debug Works

Nights and Weekends



Business Hours



Debug jobs:

- are **smaller** than "regular" jobs
- are **shorter** than "regular" jobs
- have access to **all nodes** in the system
- have advantageous **priority**

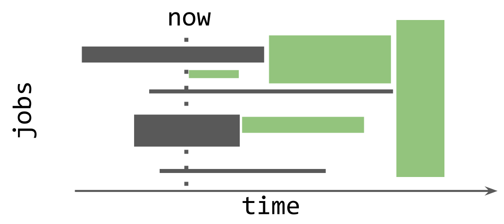
these concepts are extended for cori's realtime and shared partitions

Day/Night:

- cron-run script manipulates **regular** partition configuration (scontrol update partition=regular…)
- during night mode adds a reservation to prevent long running jobs from starting on contended nodes

Scheduling - Backfill

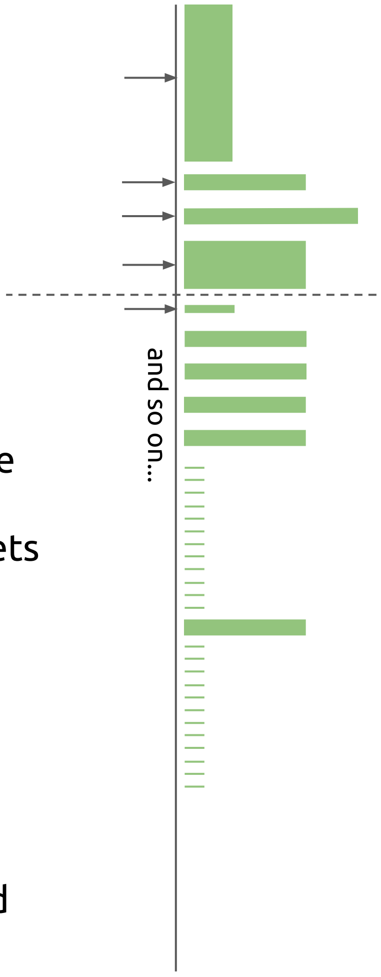
- NERSC typically has hundreds of running jobs (thousands on cori)
- Queue frequently 10x larger (2,000 - 10,000 eligible jobs)
- Much parameter optimization required to get things "working"
    - bf_interval
    - bf_max_job_partition
    - bf_max_job_user
    - ...
- We still weren't getting our target utilization (>95%)
- Still were having long waits with many backfill targets in the queue



New Backfill Algorithm!
`bf_min_prio_reserve`

1. choose particular priority value as threshold
2. Everything above threshold gets resource reservations
3. Everything below is evaluated with simple "start now" check (NEW for SLURM)

Utilization jumped on average more than 7% per day
Every backfill opportunity is realized



Job Prioritization

1. QOS
2. Aging (scaled to 1 point per minute)
3. Fairshare (up to 1440 points)

Priorities

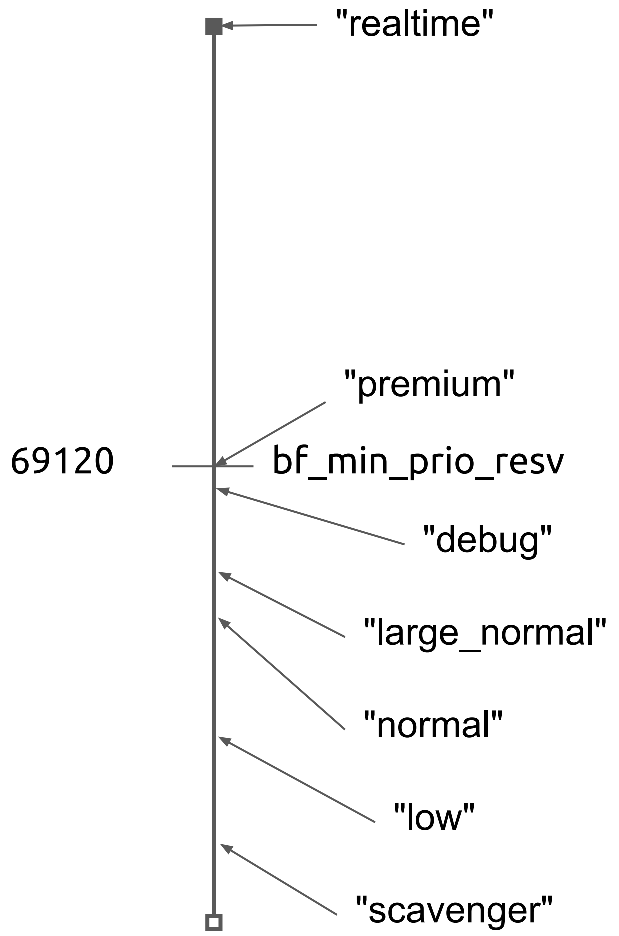
To maximize utilization, backfill resource reservations need to be maintained and re-evaluated in the *same order* every time. This implies that a monotonically increasing evolution of priorities is ideal.

Strategy: Normalize priority values into an understandable unit, this will help scale priority contributions within the multi-priority plugin

NERSC normalizes priority to time, where job age is changes job priority over time. We use fairshare as a *very small* contribution for local sorting of job priorities.

Priorities

1. Choose scale of job priority variability and "units" of priority
    a. PriorityMaxAge: 128-00:00:00
    b. PriorityWeightAge: 184320
        i. 128 days * 1440 minutes/day
        ii. MaxAge / Weight = 1 priority pt/minute
2. Choose QOS (or PartitionJobFactor) Priorities to "position" jobs on the scale
    a. PriorityWeightQOS = PriorityWeightAge + bf_min_prio_resv value
    b. Add "scalingfactor" qos with priority=PriorityWeightQOS to ensure all priorities are on constant scale (**NO qos with higher priority than *scalingfactor)***
3. Can add other priority factors *very carefully*, e.g., FairShare, but perhaps at < 5% of bf_min_prio_resv
    a. PriorityWeightFairshare=1440 (one "day" of priority)



Shifter - Linux Containers for HPC

- Docker-like functionality enabled on HPC systems
  - Direct import of any docker image
- End-to-end Slurm integration
  - Integrated image selection and configuration
  - Integrated job start
- Security model compatible with traditional HPC needs
  - User is user
  - No privileges accessible to container processes
- Fully allows resource manager to manage resources
  - Does not manipulate cgroups
- Transparent support for site-optimized MPI with generic container images

```
#!/bin/bash
#SBATCH --image=dmjacobsen/mpitest
#SBATCH --volume=/scratch/dmj/i:/input
#SBATCH --volume=/scratch/dmj/o:/output
#SBATCH -N 250

srun shifter /app
```

Exciting slurm topics I'm not covering today

user training and tutorials

accounting/integrating slurmdbd with NERSC databases

user experience and documentation

draining dvs service nodes with prolog

slurm deployment strategies

details of realtime implementation

blowing up slurm without getting burned

burstbuffer / DataWarp integration

NERSC slurm plugins: vtune, completion

ccm

reservations

job_submit.lua

monitoring

knl

Conclusions and Future Directions

- We have consistently delivered highly usable systems with Slurm since it was put on the systems
- NERSC has filed 142 issues to date - typical experience is that bugs are repaired **same-or-next day**
- Slurm as-a-resource manager on Cray is a new technology that has rough edges with great opportunity!

- Integrating Cori Phase 2 (+9300 KNL)
  - 11,300 node system
  - New processor requiring new NUMA binding capabilities, node reboot capabilities,
  - Expect large increase in job flux with major increase in scale combined with new architecture
- Increasing visibility into slurm operations
- Working to improve priority management to optimize utilization and fairness

Acknowledgements

NERSC

- Tina Declerck
- David Paul
- Ian Nascimento
- Stephen Leak

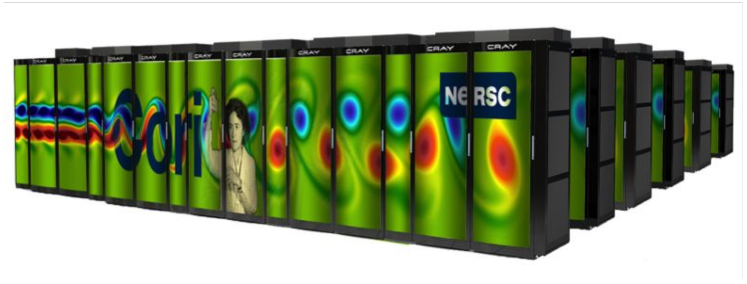
Cray

- Brian Gilmer

SchedMD

- Moe Jette
- Danny Auble
- Tim Wickberg
- Brian Christiansen

NERSC is Hiring!



We want exceptional individuals with

- Strong systems programming skills
- Deep understanding of systems architecture
- Interest in new and innovative technology

You will

- Work on some of largest systems anywhere in the world
- Make an impact on how thousands of researchers use HPC systems