SLURM.
Our Way.

Douglas Jacobsen, James Botts, Helen He
NERSC

CUG 2016

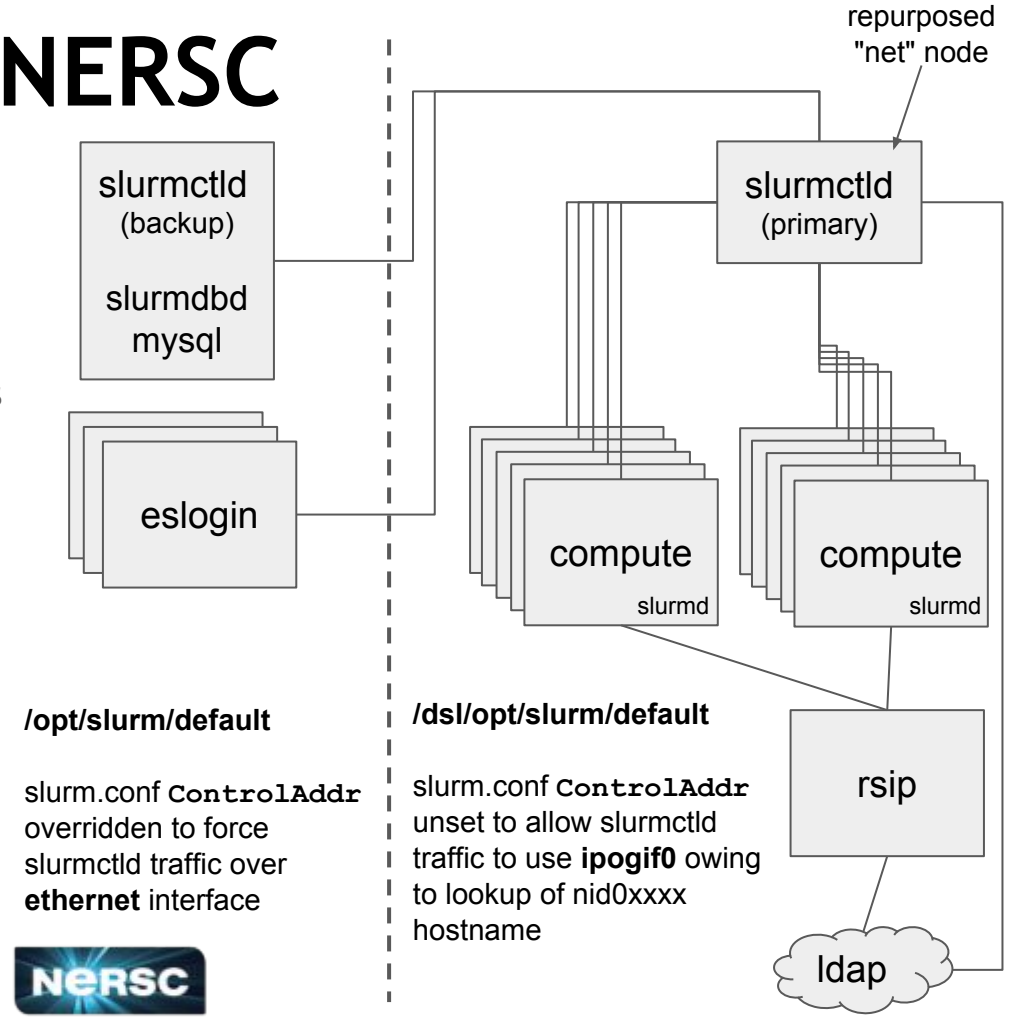
NERSC Vital Statistics

- 860 active projects
  - DOE selects projects and PIs, allocates most of our computer time
- 7750 active users
- 700+ codes both established and in-development
- edison XC30, 5586 ivybridge nodes
  - Primarily used for large capability jobs
  - Small - midrange as well
  - Moved edison from Oakland, CA to Berkeley, CA in Dec 2015
- cori phase 1 XC40, 1628 haswell nodes
  - DataWarp
  - realtime jobs for experimental facilities
  - massive quantities of serial jobs
  - regular workload too
  - shifter

Native SLURM at NERSC

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"



Basic CLE 5.2 Deployment

Challenge: Upgrade native SLURM
Issue: Installed to /dsl/opt/slurm/<version>, with symlink to "default".
→ Changing symlink can have little impact on actual version "pointed to" on compute nodes
Result: Often receive recommendation to reboot supercomputer after upgrading.

Challenge: NERSC patches SLURM often and is not interested in rebooting
Issue: /dsl DVS mount attribute cache prevents proper dereference of "default" symlink
Solution: mount /dsl/opt/slurm a second time with short (15s) attrcache
Result: NERSC can live upgrade without rebooting

Also moved slurm sysconfdir to /opt/slurm/etc, where etc is a symlink to conf.<*rev*> to workaround a rare dvs issue

Original Method:

```
/opt/slurm/15.08.xx_instTag_20150912xxxx
/opt/slurm/default -> /etc/alternatives/slurm
/etc/alternatives/slurm -> /opt/slurm/15.08.
xx_...
```

Production Method:

```
/opt/slurm/15.08.xx_instTag_20150912xxxx
/opt/slurm/default -> 15.08.xx_instTag_20150912xxxx
```

AND

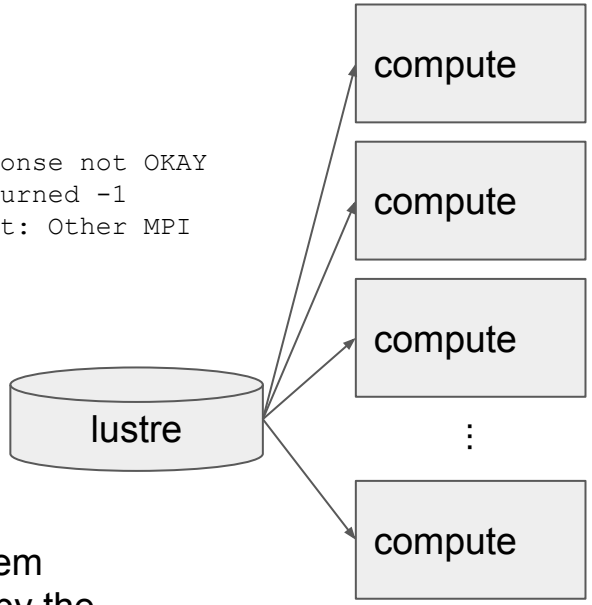
Compute node /etc/fstab:

```
/opt/slurm /dsl/opt/slurm dvs \
    path=/dsl/opt/slurm,nodename=<dslNidList>, \
    <opts>,attrcache_timeout=15
```

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: New 15.08 srun feature merging sbcast and srun

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

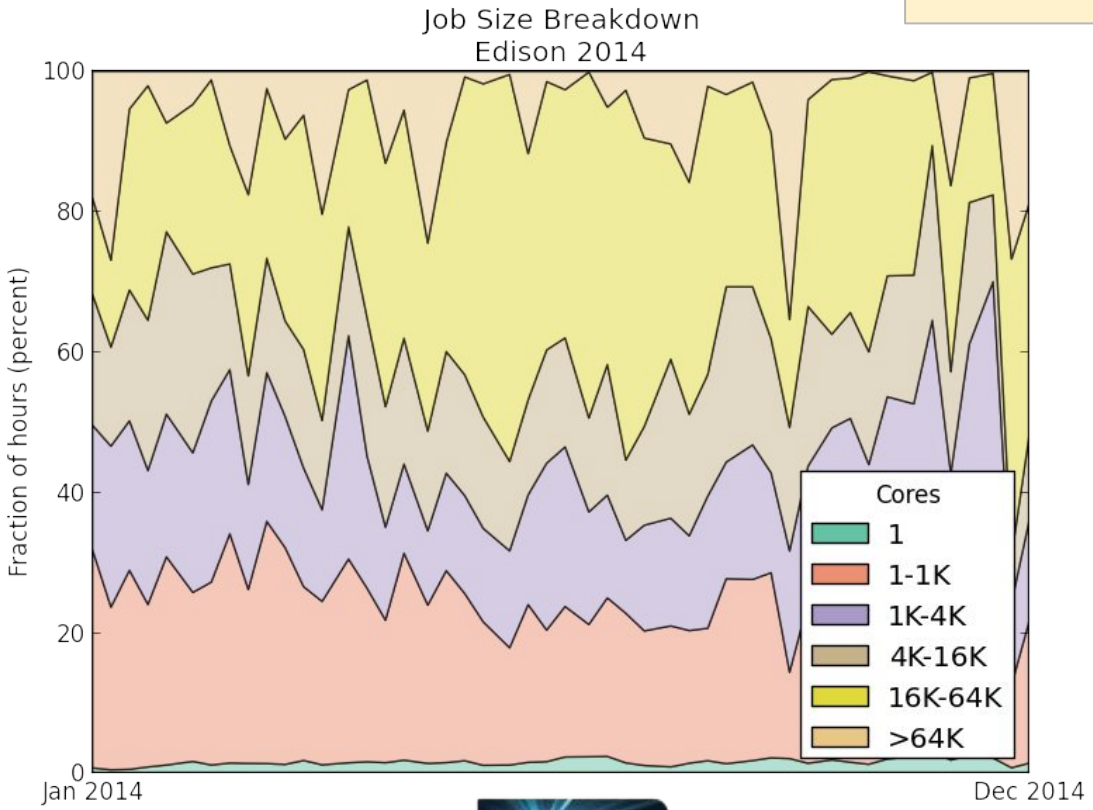
slurm 16.05 adds `--compress` option to deliver executable in similar time as aprun

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

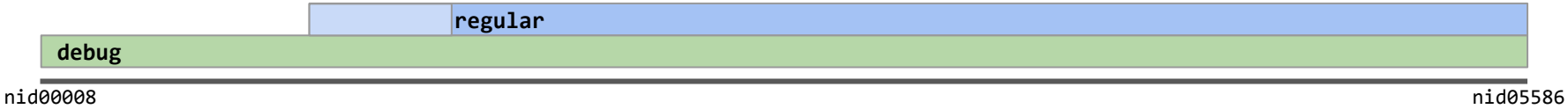
- big job metric - need to always be running at least one "large" job (>682 nodes)
  - Give priority boost + discount

cori+edison

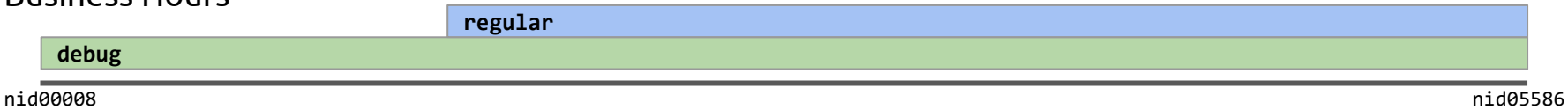
- debug partition
  - delivers debug-exclusive nodes
  - more exclusive nodes during business hours
- regular partition
  - Highly utilized workhorse
- low and premium QOS
  - accessible in most partitions
- scavenger QOS
  - Once a user account balance drops below zero, all jobs automatically put into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

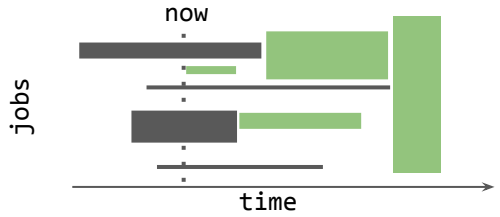
these concepts are extended for cori's realtime and shared partitions

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



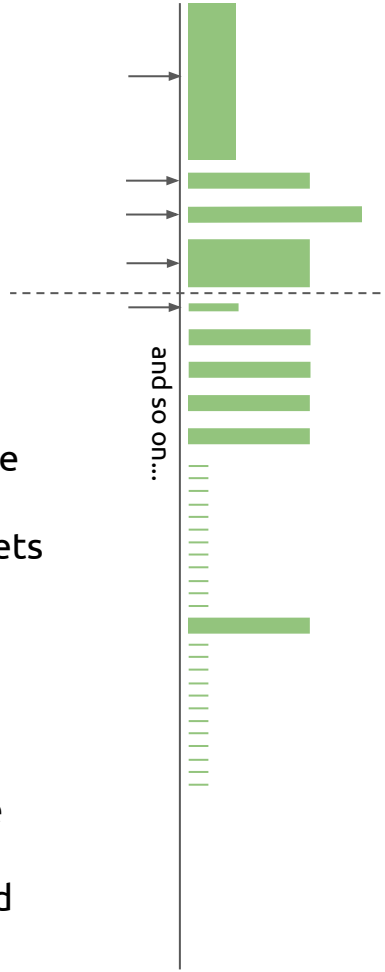
New Backfill Algorithm!

`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

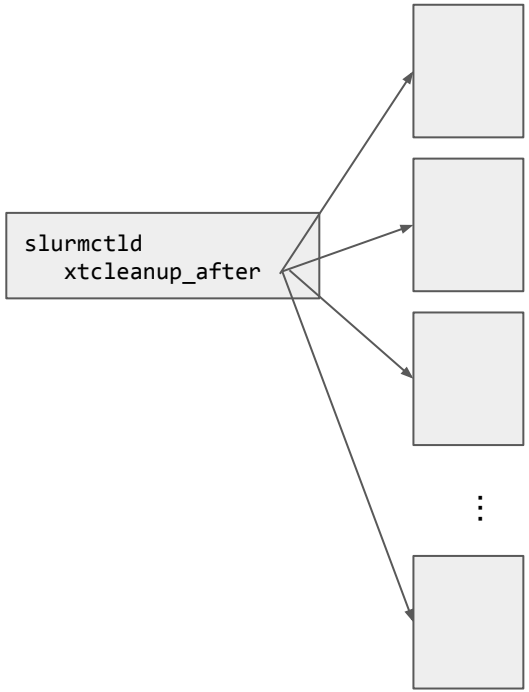
Every backfill opportunity is realized



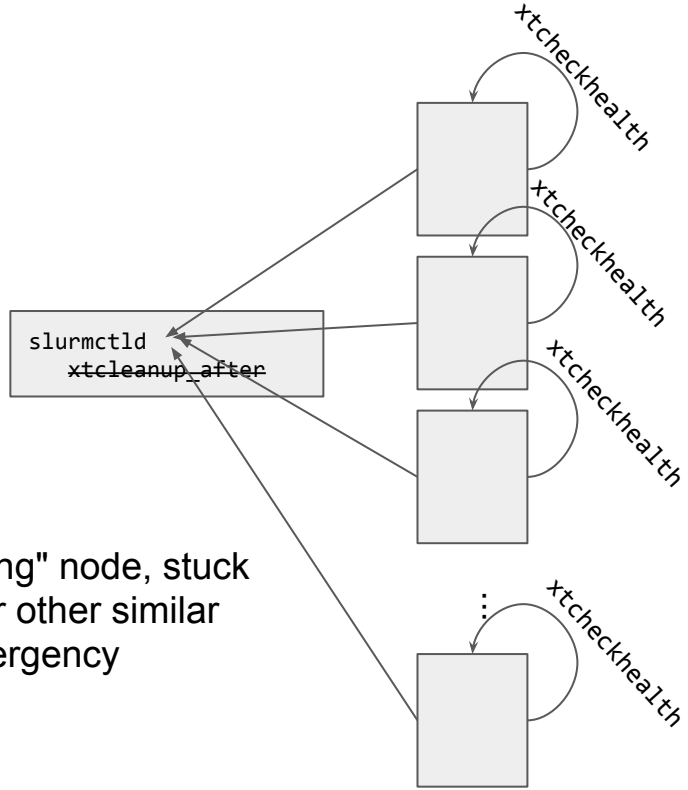
Job Prioritization

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

Primary Difficulty Faced



needs to become



Issue is that a "completing" node, stuck on unkillable process (or other similar issue), becomes an emergency

NHC doesn't run until entire allocation has ended. In cases slow-to-complete node, this holds large allocations idle.

If NHC is run from per-node epilog, each node can complete independently, returning them to service faster.

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

my speculations about Rhine/Redwood

details of realtime implementation

blowing up slurm without getting burned

burstbuffer / DataWarp integration

NERSC slurm plugins: vtune, blcr, shifter, 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
- Our typical experience is that bugs are repaired same-or-next day
- Native SLURM is a new technology that has rough edges with great opportunity!
- Increasing resolution of binding affinities

- Integrating Cori Phase 2 (+9300 KNL)
  - 11,000 node system
  - New processor requiring new NUMA binding capabilities, node reboot capabilities,
- Deploying SLURM on Rhine/Redwood
  - Continuous delivery of configurations
  - Live rebuild/redeploy (less frequent)
- Scaling topologically aware scheduling

Acknowledgements

NERSC

- Tina Declerck
- Ian Nascimento
- Stephen Leak

Cray

- Brian Gilmer

SchedMD

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