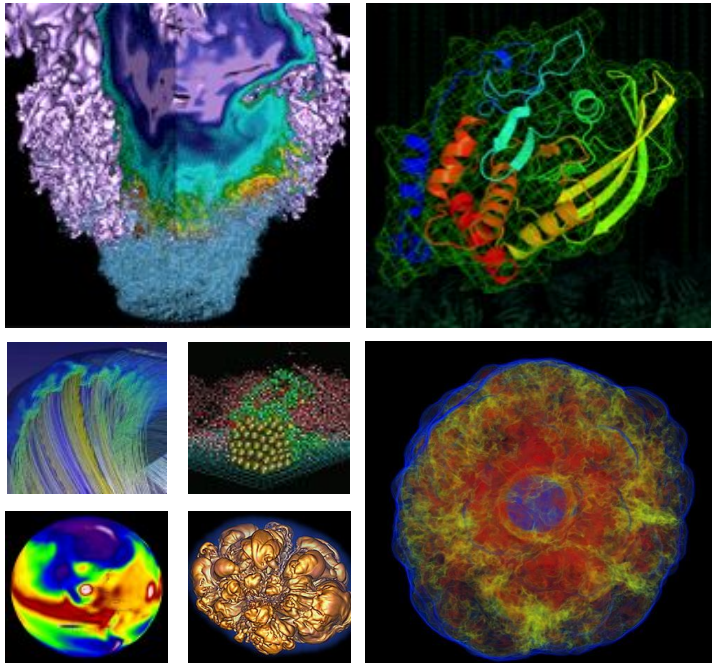
Scheduling Jobs at NERSC



Lisa Gerhardt

May 20, 2020
NUG SIG EX Meeting

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Large-scale simulations
"I need to run my climate model on 9000 nodes"

Shared-node Queue
Transfer Queue
"I only need 2 cores (and I'm not willing to pay for the full node)"
"I need to transfer many PB of data"

Pipeline/Workflow Management Nodes
"I need a service running 24/7 to manage my data analysis pipeline"

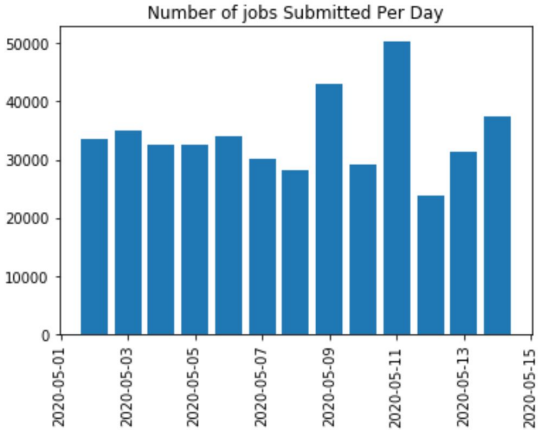
Real-Time Queues for Co-Scheduling w/Experiments
"I need to analyse this microscope data immediately otherwise my experiment will fail"

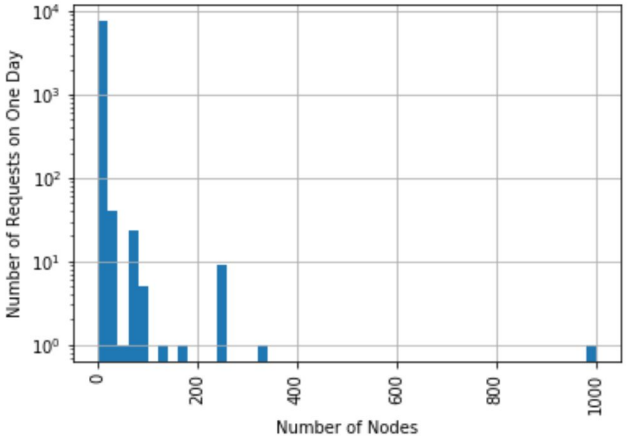
Deadline Computing
"I need to analyze this telescope data before sunrise"

Interactive Queues: 64 Nodes x 4 Hours
"I need to interactively debug my code at scale without waiting in the batch queue"

NERSC

Scheduling is tough



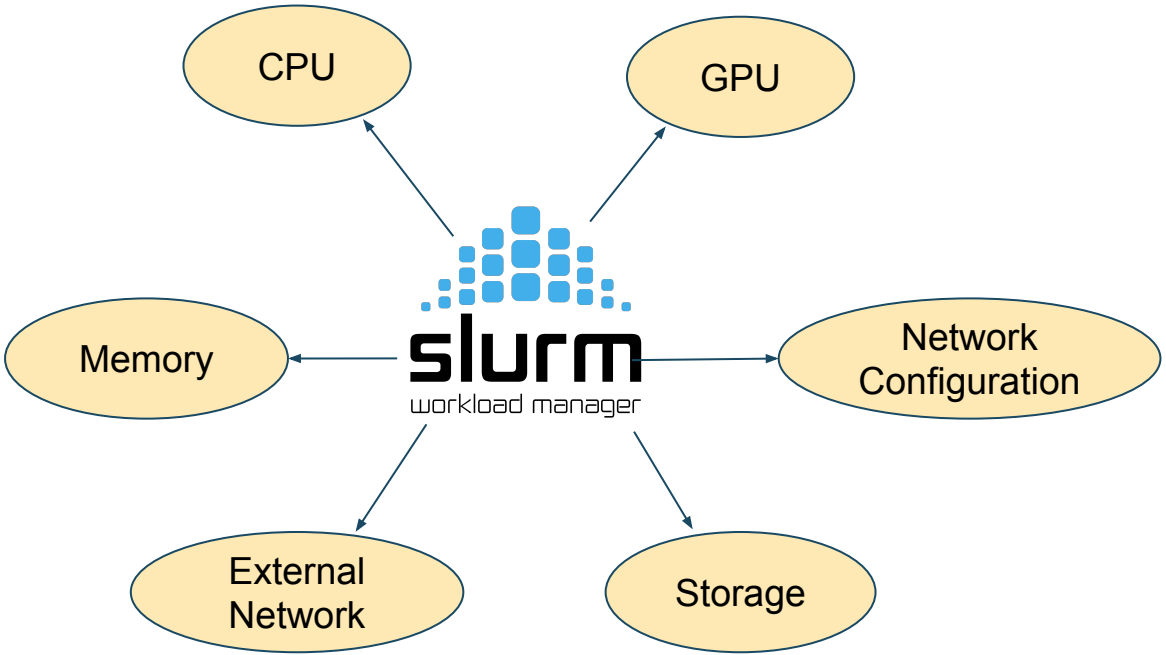


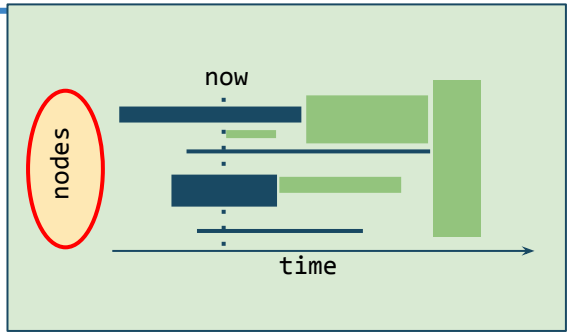
Running Jobs = Job Queue + Algorithm + Policy

Jobs	Algorithm	Policy
• Resource Requests/Constraints • Job Script • Time limit • Metadata	• Immediate Scheduler (FIFO) • Conservative Backfill • Aggressive Backfill	• Initial Priority • How priority changes over time • Quantity/type of resources allowed • System constraints

Heterogeneous Scheduling

The system isn't simply comprised of "nodes", slurm has to balance utilization of many different resources.



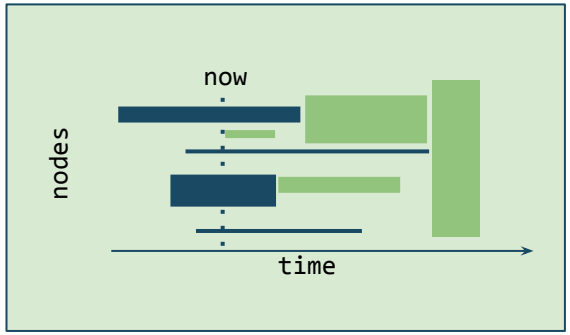


Systems group and SchedMD have developed tools for scheduling GPUs already deployed on Cori's GPU testbed.

NERSC

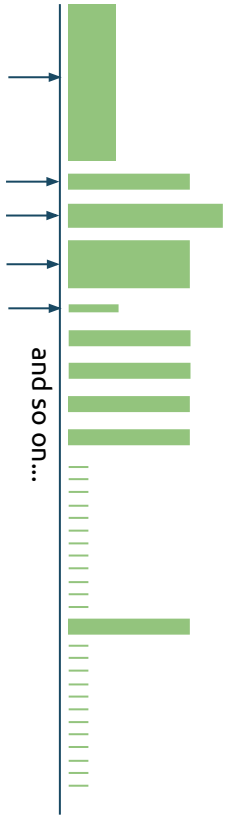
A Tale of Two Schedulers

- Slurm has two schedulers
  - Immediate scheduler looks at only the top 400 jobs
  - Backfill looks at all the pending jobs ordered by priority and QOS and tries to pack them in as efficiently as possible
- Priority of jobs is based on time since submission (with some bonuses depending on the QOS)



Slurm Backfill Enhancements:

- Immediately improved utilization by 7%
- Further improvements have helped keep utilization percent in the mid-high 90s



Backfill

- To pack the system efficiently, the backfill scheduler must be able to look through all the pending jobs in the scheduling window
  - In an ideal world, backfill would loop over all the jobs every 30 seconds
  - In practice, the shortest we've seen it take is 3 minutes
  - Configured cutoff for calculation is 15 minutes
- Utilization drops if the backfill scheduler can't get through the list in this time
  - Usually means slurm is under stress from another source, like too many sruns or squeues, or many nodes finishing at once
  - In the past, could also happen if there are too many jobs to calculate, which led to us imposing limits on what jobs are eligible for backfill

Priority and Backfill Details

- Jobs enter the system at 64800 priority for shared and regular QOS
  - Debug is at 69060 but it has its own dedicated pool of nodes
  - Premium is at 69120
- Jobs age in priority at a rate of 1 / minute
- Only two jobs per user age at a time (more on this in a moment)
- To get through the backlog in time, the backfill algorithm will only reserve (i.e. schedule) jobs with a priority higher than 69121
  - Only looks at the first 100 jobs per association (user + account)
  - It will only run lower priority jobs if they can start immediately and without delaying already scheduled higher priority jobs
- Backfill rebuilds from scratch each time. If a higher priority job appears it can interject itself and reorder the whole schedule
  - This is why predicting starting times is hard

Rule of Two

- Each job puts a fixed load on the backfill scheduler regardless of size
  - Limiting ageing to two jobs per user is an important tool for keeping the load manageable
- We understand this poses a challenge to many users analyzing data from experimental facilities who may use significant amounts of compute time, but bundle work in smaller elements
- This is why we we recommend using different workflow tools as they can ‘trick’ SLURM into thinking there is a single job to schedule but enables multiple jobs to run inside the reservation
  - See Bill’s presentation in two weeks on workflow options
- Longer term we are working on how to better schedule higher throughput workflows without bringing down the scheduler

Scheduling Improvements

- Systems group and SchedMD have been working on improving the performance of the backfill scheduler
  - Funded development to handle storm of LDAP queries
  - Cli_filter: checks user's scripts before the submit filter, reduces load on the scheduler
- Performance has improved to the point where the 3-day gap can be dramatically reduced
  - Will allow jobs to be much more rapidly considered by the backfill scheduler
  - Should be implemented soon
- Future ideas for increasing throughput
  - Would need funded development from SchedMD and may take a few years to be available
  - Only considering the top X node hours for ageing (so 100 1-node jobs from a user is the same a 1 100-node job from another user)
  - Bundling jobs from the same user with the same shape into a "workflow block" that slurm schedules all at once

Questions for You

- If you have a large ensemble of jobs you need to run
  - How much needs to finish to be able to move onto the next stage?
    - 100%? 80%? 50%?
  - What if you could get a small fraction much sooner than the rest?
  - Can these jobs be bundled together?
  - Are these jobs all the same (i.e. same wall time, memory usage, number of cores)?
  - Do they have dependencies?
- If you're not running a large ensemble, what does your workflow look like?
- Are they other features of our queue policy that you would change?

Thank You