Slurm at the George Washington University
Tim Wickberg - wickberg@gwu.edu
Slurm User Group Meeting 2013
September 19, 2013
Colonial One Background

- Colonial One - new shared HPC cluster at GW
  - GW has no prior experience managing HPC systems at a University-wide level
- “Pay to play” - groups who contribute resources have priority in the scheduling system
- Not a “condo”, priority on overall system, not on dedicated hardware they bought
  - Impact on scheduling priority (more on this later)
Dell C8220 cluster, 96 node currently
- 32x GPU nodes, each with dual NVIDIA K20 GPUs
- 64x CPU nodes, each with dual 2.6GHz 8-core Intel Xeon CPUs, and 64/128/256GB of RAM

Heterogeneous hardware... not ideal for a new system
- Need to carve out separate partitions, make it obvious how to get requested resources
- Most users only care about CPU vs. GPU
- 5 partitions - 64gb, 128gb, 256gb, defq (all three cpu node types), and gpu
Colonial One - Current System
Bright Cluster Manager 6.0
- Uses Slurm 2.4 by default
- Partitions match node definitions in CM

Switched to manually installed Slurm 2.6
- Needed more control over:
  - accounting - using for priority
  - partitions - difference between logical and software images

And to get new features...
Cool New Feature - Job Arrays

• New in 2.6
  • Didn’t know we needed it until it was available
  • Users immediate took to it

• Genomics, Molecular Biology, Physics…
  • Submit hundreds to thousands of identical jobs with different job seeds.
Before:

- Users run their own `./launch.sh` script, which looks like:
  ```bash
  for i in `seq 1 300`; do
    sbatch ./slurm.sh 100 $i
  done
  ```
- Adds hundreds of jobs to the queue at once
- ‘squeue’ becomes unreadable

Cool New Feature - Job Arrays (2)
Cool New Feature - Job Arrays (3)

- After:
  
  ```bash
  sbatch --array 1-300 ./slurm.sh 100 %a
  ```

- Can be managed with a single job number
- Array values can be embedded in job scripts with `#SBATCH` directives - easier for users to share
- Keeps the queues tidy
Other initial tricks

• Force users to set a time limit
  • `job_submit/require_timelimit` plugin
  • thanks to Dan Weeks, RPI

• Improve backfill scheduling by getting better estimates from the users

• Don’t give the users a default - they won’t change it, hurting system throughput
• Complicated due to funding relationships,
  • But Slurm helps with multifactor priority plugin
• Currently running priority/multifactor, with accounting hierarchy built between different schools and research groups.
• Looking at alternatives and ways to improve - QOS / other priority mechanisms?
Requests...

- Priority tools - we have a lot of demands to demonstrate disparate groups are getting their “fair” share of resources
  - Reporting on current status vs. ideal priority settings
  - Simulation tools to model different priority / QOS adjustments reusing past submission info
Thank You

Any questions?

Tim Wickberg - wickberg@gwu.edu
http://colonialone.gwu.edu