Slurm Workload Manager

Tim Wickberg
SchedMD LLC
tim@schedmd.com
What is Slurm?

- Historically, Slurm was an acronym of: Simple Linux Utility for Resource Management
- Dev. started in 2002 @ Lawrence Livermore National Lab
- Originally designed as a resource manager for Linux clusters
- Sophisticated scheduling plugins added in 2008
- About 550,000 lines of C code today
- Supports Linux and limited support for other Unix variants
- Used on many of the world's largest computers
- Active global user community
Who is SchedMD

- Started by the two principle Slurm developers - Moe and Danny, the ‘MD’ in ‘SchedMD’ - in 2011 to be able to provide custom development and direct support outside of LLNL.
- SchedMD continues to develop, support, and issue new releases.
- Most recent stable release is 18.08 (2018 - August), and we’re working toward 19.05 (2019 - May) right now.
Who is SchedMD

- SchedMD provides custom development.
  - All new development is open-source, and made available through the main distribution.
  - Customers always running the same release as the wide community.
- SchedMD provides commercial support.
  - Help with configuration issues, software bugs, and architectural advice.
  - Advanced notification of security-related release.
Who is SchedMD

- SchedMD provides training.
  - Offered alongside commercial support.
  - Onsite, and customized to your system and staff.
  - Our goal is to leave you with a solid understanding of Slurm, a working configuration matching your system and policies, and a direct line to get in touch should further questions arise.
Slurm Design Goals

- Highly scalable
  - Managing 3.1 million core Tianhe-2
  - Tested to much larger systems using emulation
- Open source GPLv2, available on Github: https://github.com/SchedMD/
- System administrator friendly
- Secure
- Fault-tolerant (no single point of failure)
- Portable - Targeting POSIX2008.1 and C99
Slurm Release Schedule

- Previous major release was 17.11 (November 2017)
- Latest major release 18.08 (August 2018)
- Next major release planned 19.05 (May 2019)
18.08 Release Highlights

- Heterogeneous environments
- Burst buffer enhancements
- Fault tolerance
- Cloud computing
- Queue stuffing
- New TRES
- New TRES reporting options
- And more...
19.05 Release Roadmap

- Major development is focused on “cons_tres” plugin.
- Provides for scheduling and allocation based around alternate resource types, such as GPUs.
  - Make them first class citizens, instead of focusing primarily on allocating CPUs to jobs.
  - Extensive architectural work to plan job allocation, extend backfill scheduling systems, and add new syntax to simplify resource requests.
New select/cons_tres Plugin

- “cons_tres” represents “Consumable TRES”
- “TRES” represents “Trackable RESources”
- All functionality provided by “cons_res” plugin is also supported by “cons_tres” (e.g. CR_LLN, CR_PACK_NODES, CR_SOCKET, etc.)
- New “gpu” job options only supported the cons_tres plugin
  - No other select plugin recognizes the new GPU options
New Job Submit Options for GPUs

Same options apply to salloc, sbatch and srun commands

- **--gpus-per-node=** Works like “--gres=gpu:” option today
- **-G/--gpus=** GPU across entire job allocation (GPUs per job)
- **--gpus-per-socket=** GPUs per allocated socket
- **--gpus-per-task=** GPUs per spawned task
- **--cpus-per-gpu=** CPUs required per allocated GPU
- **--gpu-bind=** Task/GPU binding option
- **--gpu-freq=** Specify GPU freq, memory freq, voltage
- **--mem-per-gpu=** Memory per allocated GPU
Configuration Changes

● New GPU parameters available globally and on per-partition basis. The command line options override these default values.
  ○ DefCpusPerGPU= Default CPUs count per allocated GPU
  ○ DefMemPerGPU= Default memory size per allocated GPU
● GPUs state information gathered using NVIDIA library
  ○ GPU specification in gres.conf file no longer required
Examples of Use (1 of 2)

$ sbatch --ntasks=16 --gpus-per-task=2 my.bash

$ sbatch --ntasks=8 --ntasks-per-socket=2 --gpus-per-socket=tesla:4 my.bash

$ sbatch --gpus=16 --gpu-freq=low,verbose --gpu-bind=closest --nodes=2 my.bash

$ sbatch --gpus=gtx1080:8,gtx1060:2 --nodes=1 my.bash
Examples of Use (2 of 2)

Allocation of resources to job steps also supports these GPU options:

$ cat my.bash
#!/bin/bash
srun --gpus=1 --ntasks=1 --nnodes=1 app1 &
srun --gpus=1 --ntasks=1 --nnodes=1 app2 &
srun --gpus=2 --ntasks=1 --nnodes=1 app3 &
srun --gpus=2 --ntasks=1 --nnodes=1 app4 &
wait

$ sbatch --gpus=2 my.bash
Longer term and experimental work

- Previews of a few long-term projects
  - nss_slurm
  - RESTful API
  - Containers

- Features may be in the 19.05 release.
  - But these are active development projects.
  - And are not yet committed to a specific release target.
nss_slurm

- Allow the slurmd process on each compute node to provide uid/gid/username lookup directly.
- Could potentially remove the need for nslcd, sssd, and other user and group info caching services on the compute nodes.
- Simplify cloud-burstsed and containerized workloads by removing the requirement for access to LDAP/NIS.
RESTful API

- Provide a translation layer between a REST service to Slurm’s internal RPC layers.
- Allow for job submission and job management over web services.
- Should provide for better integration with emerging tools such as JupyterLab.
- Simpler target for internal portals and other tools to build off of.
RESTful API

- Exact set of supported APIs is still being determined.
  - We are interested in feedback from the community as to what your anticipated use cases may be.
Containers

- Slurm already supports resource allocation and enforcement through the Linux cgroups infrastructure.
  - Additional job containment possible through tools such as `pam_slurm_adopt`.
- Support for Singularity and/or Shifter integrations today.
  - Designed to solve the software packaging side of containerization.
  - This is, in my opinion, the compelling use case for containers in HPC.
Containers

- Long-term, expect to see Slurm move towards built-in capabilities to manage and launch containers directly.
  - Looking into OCI and other image formats, and closely following developments from other HPC-Containerization working groups.
Containers

- Long-term, we do still see value in the traditional HPC scheduler role, and Slurm in particular.
  - K8S and other technologies assume that compute capacity is available on demand.
  - In HPC systems, demand outstrips resources, and sophisticated scheduling, job prioritization, and user limits still have a role to play in managing access to those resources.
Slurm and SchedMD @ SC18

Join us at booth 1242.
Presentation schedule available at the booth.