Slurm 19.05 Release

Tim Wickberg
SchedMD

Slurm User Group Meeting 2019

19.05 Release Contributors

Aditi Gaur, Albert Gil, Alejandro Sanchez, Artem Polyakov, Artem Y. Polyakov, Bas Nijholt, Ben Roberts, Boris Karasev, Brian Christiansen, Broderick Gardner, Chad Vizino, Chris Rorvick, Chris Samuel, Daniel Letai, Danny Auble, Didier GAZEN, Dineshkumar RAJAGOPAL, Dominik Bartkiewicz, Doug Jacobsen, Felip Moll, Gavin Howard, Gennaro Oliva, Isaac Hartung, Jacob Jenson, Jakub Yaghob, Janne Blomqvist, Jason Booth, Josko Plazonic, Kilian Cavalotti, Lewis Lakerink, Marcin Stolarek, Marshall Garey, Matt Ezell, Matthias Gerstner, Michael Hinton, Mike Nolta, Moe Jette, Nate Rini, Paddy Doyle, Paolo Margara, Philip Kovacs, Ross Dickson, Tim Wickberg, Trey Dockendorf, Yu Watanabe

New cons_tres Select Plugin

- “cons_tres” represents “Consumable TRES”
- “TRES” represents “Trackable RESources”
  - GPUS are a type of TRES within Slurm.
- GPUs added as a first-class entity alongside CPUs and Memory

cons_tres - New 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

cons_tres - 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

cons_tres

- Can revert to cons_res without loosing the queue
  - Although jobs using new cons_tres options cannot run
  - Both share a common state format to make this possible
    - Unlike cons_tres ⇔ serial which will drop the queue
- Long-term, cons_tres will replace cons_res
  - Both are supported for 19.05 and 20.02 releases
  - Expect to see cons_res removed before 20.11 release

Cloud/PowerSave Improvements

- Cloud/PowerSave Improvements:
  - Better responsiveness to resuming and suspending nodes
  - Powering down nodes put in "Powering Down / %" state until after **SuspendTimeout**.
  - Powering down nodes not eligible to be allocated until after **SuspendTimeout**
- Allocate nodes that are booting.
  - Previously, nodes that were being booted were off limits for allocation
  - Caused more nodes to be booted than needed in a cloud environment

Preemption

- Added PreemptExemptTime parameter
  - Job not eligible for preemption for configured time.
    - Better than GraceTime
  - Can set global parameter in slurm.conf
  - Set on QOS
    - Use partition QOS to use on partition

Job Prioritization Mechanisms

- FairTree scheduling has been made the default.
- Added new NO_NORMAL_[ALL|ASSOC|PART|QOS|TRES] options to PriorityFlags to disable factor normalization if desired.
  - May make it simpler to build complex priority models, especially if using other options like bf_max_prio_resv

New Job Prioritization Mechanisms

- Added new Association job priority factor
  - Set through sacctmgr on an association
  - Recursively applies to lower levels of the association hierarchy if not explicitly set
  - Easier way to set strict priority offsets based on user or account membership.

New Job Prioritization Mechanisms

- Added new Site Factor
  - Designed to be set through site-specific plugins, and allow you to build your own priority scheme
  - Can be set either through the existing job_submit plugin API statically for each job
  - Or through a new site_factor plugin API, which can set it initially upon job submission, and update it periodically until the job launches.
  - No normalization, raw integer values will be added to the other factors

cli_filter interface

- Developed with NERSC
- Designed to allow for user-command-side option manipulation
  - Permits for slower and more complex routing logic that would cause performance issues within the slurmctld if run as a job_submit plugin
  - Note: unlike job_submit, you cannot rely on cli_filter as a security mechanism, as users can potentially bypass anything run command side

cli_filter interface

- Provides a consistent string-based interface to all allocation options
- Allows a cli_filter plugin to change/reset any and all options
- Can be used to run checks that should not be done as part of job_submit
  - E.g., check filesystem quotas

cli_filter

- Zero plugins ship in 19.05, but the API is supported today
- NERSC may submit some of their plugins for general inclusion ahead of the 20.02 release
  - Talk to them if you want to use them today

cli_filter interface

- Required complete overhaul of salloc, sbatch, and srun argument parsing code
  - All merged into src/common/slurm_opt.c
- Associated cleanup is why --cpu_bind went away briefly
  - Deprecated in favor of --cpu-bind since 17.11 release
  - OpenMPI has been hard-coding it in their srun wrapper scripts, so it'll stay for the foreseeable future

Revamped X11 Forwarding

- The Slurm Internal X11 forwarding has been completely revamped.
  - No longer uses libssh2, now uses MUNGE credentials to authenticate forwarding requests between the nodes and the originating salloc/srun command
  - Still enabled with PrologFlags=X11, no additional changes required
  - Provides higher throughput, and works in more varied environments where SSH key management did not align with our previous design

Revamped X11 Forwarding

- Implemented as a general-purpose MUNGE authenticated network forwarding layer
  - Could be used for other network forwarding tricks if desired
  - File a ticket if you have a use case for this

New nss_slurm capability

- Slurm can now serve as an NSS provider through a new nss_slurm.so.2 library.
- Extends the existing “LaunchParameters=send_gids” concept to provide uid/gid resolution for the job’s owner within any processes spawned by that job.
- Avoids LDAP/NIS performance issues, especially on large-scale job launches, or after node reboots have cleared the NSS caches.

New nss_slurm capability

- UID/User Name/GECOS/Home Dir/Shell, and a list of all GIDs/Group Names the user belongs to encoded as part of the task launch credential.
- This info will be provided to processes within a given job step for any getpwuid/getpwnam/getgrgid/getgrnam syscalls.
- Enabled with LaunchParameters=enable_nss_slurm, installing libnss_slurm.so.2 in the node image, and adding slurm to the passwd and group providers in nsswitch.conf.

New nss_slurm capability

- Designed so that it can stack alongside other NSS providers.
- Or may, depending on your site's configuration, be able to replace sssd/nslcd/ldap on the node entirely.

nss_slurm demo

Cray Specific Changes

- Heterogeneous Jobs are now supported.
  - Previously restricted to non-Cray MPI implementations.
- Support for Cray/ALPS has been removed.
  - All systems must run in “Native Cray” (a.k.a. “Native Slurm”) mode.

Cray Specific Changes

- Support for “Cray NHC” has been removed.
  - Sites should use a Cray-provided Epilog script instead to provide similar health-check behavior on the node directly.
  - Allows for jobs with failed nodes to return the majority of the nodes to service immediately, rather than blocking access to all nodes until an admin has intervened and released the ALPS reservation.
  - This Epilog script has been previously adopted by most larger Cray Aries + Slurm installations, and is now the only supported approach in 19.05.

Cray Plugin Renaming

- In preparation of the new Cray Shasta stack, all “cray” plugins have been renamed “cray_aries”.
- Except for “burst_buffer/cray”, which is now “burst_buffer/datawarp”.
- No functional changes, but your slurm.conf configuration files will need to be updated as part of an upgrade.

Questions?