LCI workshop
on high performance clustered computing

Linux Clusters Institute: Scheduling and Resource Management

slurm
workload manager

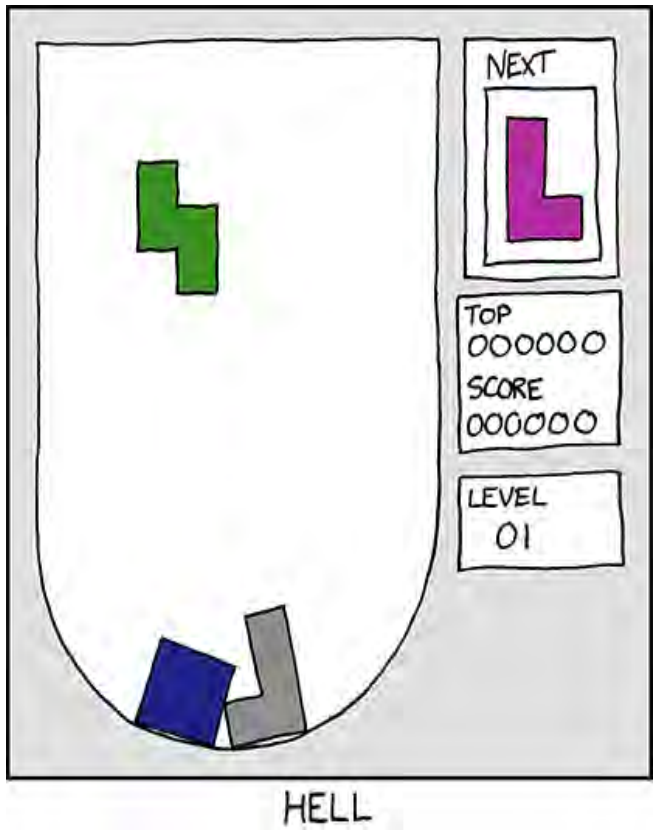
Brian Haymore, Sr IT Arch - HPC, University of Utah

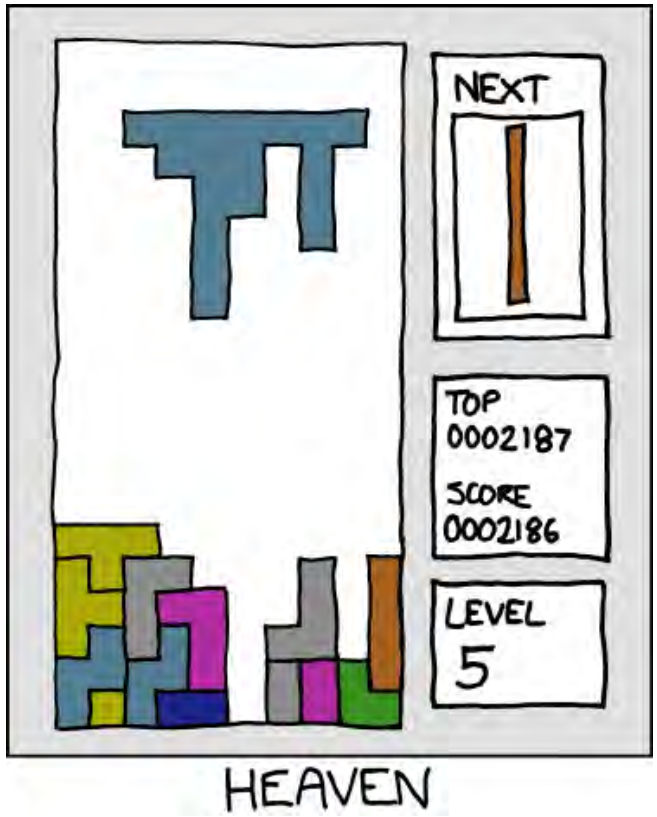
May 2017

THE UNIVERSITY OF UTAH®

This content was originally developed by Ryan Cox (2014) and updated by Brian Haymore (2017) in for LCI.

(image source: xkcd.org)





Slurm Workload Manager

- What is Slurm?
- Installation
- Slurm Configuration
- Daemons
- Configuration Files
- Client Commands
- User and Account Management
- Policies, tuning, and advanced configuration
- Priorities
- Fairshare
- Backfill
- QOS
- Accounting
- Health Check
- Troubleshooting

Concept: What is Slurm?

Simple Linux Utility for Resource Management:

- Anything but simple...
- Resource manager and scheduler
- Originally developed at LLNL (Lawrence Livermore)
- GPL v2
- Commercial support/development available
- Active user community
- Core development by SchedMD
- Other major contributors exist
- Built for scale and fault tolerance
- Plugin-based: lots of plugins to modify Slurm behavior for your needs

BYU's Scheduler History

- BYU has run several scheduling systems through its HPC history.
- Moab/Torque was the primary scheduling system for many years.
- Slurm replaced Moab/Torque as BYU's sole scheduler in January 2013.
- BYU has now contributed Slurm patches, some small and some large:
  - New fair share algorithm: LEVEL_BASED (depreciated now).
  - cgroup out-of-memory notification in job output.
  - Pam-slurm-adopt pam module for process control with cgroups.
  - Script to generate a file equivalent to PBS_NODEFILE.
  - Optionally charge for CPU equivalents instead of just CPUs.

University of Utah's Scheduler History

- Utah has also run over the years several batch and or scheduling systems including DQS, NQS, LoadLeveler, PBS/PBSPro/OpenPBS/Torque, Maui/Moab, and SLURM.
- Maui/OpenPBS/QBank -> Moab/Torque/(Gold/MAM) were primary scheduling and allocation enforcement systems between 1999-2015.
- Initial testing of Slurm started Fall 2014 focusing the support for elastic/cloud (private) resources.
- Completed transition to Slurm from Moab/Torque as Utah’s batch system & scheduler of choice April 2015.

Terminology

- Partition – A set, “queue” or pool of nodes that make up a “Cluster”.
- Cluster –Multiple Slurm “clusters” can be managed by one slurmdbd; one slurmctld per cluster.
- Job step – a suballocation within a job.
  - E.g. Job 1234 is allocated 12 nodes. It launches 4 job steps, each running 3 nodes.
  - Similar to subletting an apartment.
    - Sublet the whole place or just a room or two.
- User – A user (“Bob” has a Slurm “user”: “bob”).
- Account – A group of users and subaccounts.
- QOS – (Quality of Service) - Priority, Preemption, Policies, and limits.
- Association – the combination of user, account, qos, partition, and cluster.
  - A user can be members of multiple accounts with different limits for different partitions and clusters, etc.

Terminology: Daemons

- Daemons
  - slurmctld – controller that handles scheduling, communication with nodes, etc.
  - slurmdbd – (Optional) database for managing users, qos, accounts, and job accounting records.
    - communicates with MySQL or MariaDB database backend.
  - slurmd – runs on each compute node and launches jobs.
    - Maintains node state with slurmctld.
    - Hierarchical communication between slurmd instances (for scalability).
    - Launches jobs.
  - slurmstepd – run by slurmd to launch a job step.
  - munged – runs on all cluster nodes and authenticates RPC calls.
    - https://dun.github.io/munge/ (Also available via EPEL)
    - Must share a common key within the cluster or if slurmdbd is used within the pool of clusters sharing that slurmdbd.
  - slurmctld and slurmdbd can have primary and backup instances for HA.
    - State synchronized through shared file system (StateSaveLocation).

Installation

- Version numbers are Ubuntu-style (e.g. 17.02.2)
  - 17.02 == major version released in February 2017
  - .2 == minor version
- Download official releases from schedmd.com
- git repo available at http://github.com/SchedMD/slurm
  - Active development occurs at github.com; releases are tagged (git tag)
- Two main methods of installation
  - ./configure && make && make install
  - Build RPMs, etc
- Some distros have a package, usually “slurm-llnl”
  - Version may be behind by a major release or three
  - If you want to patch something, this is the hardest approach

Installation: Prep Work & Dependencies

- Create a slurm user and group locally on each node (consistent uid:gid) or via your network service of choice (ldap, nis, …).
  - No password needed, this is a service account.
- Insure package dependencies are satisfied (your exact needs may vary):
  - 'yum install openssl openssl-devel pam-devel numactl numactl-devel hwloc hwloc-devel lua lua-devel readline-devel rrdtool-devel ncurses-devel man2html libibmad libibumad munge munge-libs munge-devel rng-tools -y'
- Download Slurm Source:
  - 'wget https://www.schedmd.com/downloads/latest/slurm-17.02.2.tar.bz2'.

Installation: MUNGE Uid 'N' Gid Emporium

- 'yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm'
- 'yum install -y munge munge-libs munge-devel'
- A shared munge key must be created and distributed to all cluster nodes.
  - 'dd if=/dev/random bs=1 count=1024 >/etc/munge/munge.key'
  - 'chmod 400 /etc/munge/munge.key'
  - Distribute in your preferred fashion this key to all cluster nodes.
  - Insure permissions and ownership are correct on each node.
- Init Startup: 'chkconfig munge on' on each node.
- SystemD Startup: 'systemctl enable munge' on each node.

Installation: Non-RPM Build

- ‘tar -jxf ~/slurm-17.02.2.tar.bz2 -C /usr/local/src/’
- ‘chown -R slurm.slurm /usr/local/src/slurm-17.02.2’
- ‘cd /usr/local/src/slurm-17.02.2’
- ‘./configure --prefix=/usr/local --sysconfdir=/usr/local/etc --enable-debug --with-hdf5 --with-hwloc’
- ‘make’ (consider adding a ‘-j16’ or other value to speed up the build?)
- ‘make install’
  - Local Install:
    - Note: some files such as ${sysconfdir}/slurm.conf must be kept consistent between all nodes.
  - Network Install:
    - ${prefix} and ${sysconfdir} should be pointed to common network paths.

Installation: Non-RPM Build

- Init Startup:
  - ‘cp /usr/local/src/slurm-17.02.2/etc/init.d.slurm /etc/init.d/slurm && chkconfig --add slurm’ on slurmd and slurmctld nodes.
- SystemD Startup:
  - slurmctld node: ‘cp /usr/local/src/slurm-17.02.2/etc/slurmctld.service /etc/systemd/system/ && systemctl enable slurmctld’
  - slurmd nodes: ‘cp /usr/local/src/slurm-17.02.2/etc/slurmd.service /etc/systemd/system/ && systemctl enable slurmd’

Installation: Build RPMs

- 'yum install rpm-build'
- 'rpmbuild -ta ~/slurm-17.02.2.tar.bz2'
- Review output and find packages in: '/root/rpmbuild/RPMS/x86_64'.
- Per your clusters management method install RPMs onto cluster nodes.
  - NOTE: some files such as /usr/local/etc/slurm.conf must be kept consistent between nodes.
- Init Startup:
  - 'chkconfig slurm on' on slurmd and slurmctld nodes.
- SystemD Startup:
  - slurmctld node: 'systemctl enable slurmctld'
  - slurmd nodes: 'systemctl enable slurmd'

Configuration: slurm.conf

- Too many parameters in slurm.conf to cover everything in this session:
  - Man Page: https://slurm.schedmd.com/slurm.conf.html
  - Online Config Tools
    - Easy Version: https://slurm.schedmd.com/configurator.easy.html
    - Full Version: https://slurm.schedmd.com/configurator.html
  - Sample: '/usr/local/src/slurm-17.02.2/etc/slurm.conf.example'
- Core components of slurm.conf:
  - Cluster name, controlmachine, ports, security, timers, etc...
  - State, {epi,pro}log, logging, and accounting locations, etc...
  - Policies, backfill, process tracking, resource selection, task launch/control, etc...
  - Accounting type(s) and storage location(s).
  - ComputeMachines definitions:
    - name, addr, cpu, socket, corespersocket, threadspercore, realmemory, weight, etc…
    - gres (Generic Resources): gpu, mic(xeon phi), other...
  - Partitions: name, membership, limits, ACLs, preempt, prio, sharing, etc...

Concept: Host Range Syntax

- Usable with commands and config files.
- Host range syntax is more compact, allows smaller RPC calls, easier to read config files, etc.
- Node lists have a range syntax with [] using “,” and “-”.
- n[1-10,40-50] and n[5-20] are valid.
  - Also valid: n[01-10,40-50] and n[05-20].
- Up to two ranges are allowed: n[1-100]-[1-16].
  - I haven't tried this out recently so it may have increased; manpage still says two.
- Comma separated lists are allowed:
  - a-[1-5]-[1-2],b-3-[1-16],b-[4-5]-[1-2,7,9]
- One negative is that it can be harder to search log files for a specific node if it’s buried within a range.

Configuration: gres.conf

- Generic resource (GRES) scheduling is supported through a flexible plugin mechanism. Support is currently provided for Graphics Processing Units (GPUs) and Intel® Xeon Phi (MIC) processors.
- slurm.conf config:
  - 'GresTypes' a comma delimited list of generic resources to be managed.
    - 'GresTypes=gpu'
  - 'Gres' also must be defined as part of the relevant 'NodeName' lines.
    - 'NodeName=tux[0-7] Gres=gpu:p100:2 …'
- gres.conf config:
  - 'NodeName=tux[0-7] Name=gpu Type=p100 File=/dev/nvidia0 CPUs=0-13,28-41'
  - 'NodeName=tux[0-7] Name=gpu Type=p100 File=/dev/nvidia1 CPUs=14-27,42-55'
  - For most it is important that you match the Gres(GPU) device to the appropriate CPU cores/socket to maximize performance. Especially when cgroups are in use.

Info: Documentation & Resources

- Slurm is too configurable to cover everything here.
- Online Manpages and Documentation:
  - https://slurm.schedmd.com/documentation.html
  - https://slurm.schedmd.com/man_index.html
- Mailing Lists:
  - https://slurm.schedmd.com/mail.html
- Annual user group meeting:
  - https://slurm.schedmd.com/meetings.html
- New features are added frequently.

Commands: Overview

- squeue - view the queue.
- sbatch - submit a batch job.
- salloc - launch an interactive job.
- srun - two uses:
  - outside of a job - run a command through the scheduler on compute node(s) and print the output to stdout.
  - inside of a job - launch a job step (i.e. suballocation) and print to the job's stdout.
- sacct - view job accounting information.
- sacctmgr - manage users and accounts including limits.
- sstat - view job step information.
- sreport - view reports about usage.
- sinfo - information on partitions and nodes.
- sview - simple gui interface.
- sprio - display idle job prio details.
- scancel - signal or cancel jobs.
- sdiag - display slurmctld performance metrics.

Commands: sinfo

- Sinfo - View information about nodes and partitions.
  - Most parameters can be combined together to form very specific queries.
  - By default groups nodes of similar config and state and displays them as a 'range'.
    - 'sinfo -N ...' - Will instead list nodes one per line.
  - Query nodes by state: alloc, comp, down, drain, err, fail, idle, maint, mix, etc…
    - 'sinfo -t ${state}'
  - 'sinfo -p ${partition}' - List nodes of a particular partition.
  - 'sinfo -T' - Display reservation names, state, start/end time, duration, nodelist.
  - 'sinfo -R' - Display reasons nodes are in state down, drain, fail, or failing state.
  - Customize output format: Example - Compact state per type per partition.
    - 'sinfo -o "%20P %5D %14F %8z %10m %10d %11l %16f %N %20G"'

```
PARTITION       NODES NODES(A/I/O/T) S:C:T    MEMORY    TMP_DISK   TIMELIMIT   AVAIL_FEATURES   NODELIST GRES
kingspeak*         32   32/0/0/32    2:8:2    64000     340000     3-00:00:00  chpc,c16         kp[001-032] (null)
kingspeak*         16   16/0/0/16    2:10:2   64000+    820000+    3-00:00:00  chpc,c20         kp[110-111,158-167,196-199]
(null)
ucgd-kp            28   27/1/0/28    2:10:2   128000    690000     14-00:00:00 ucgd,c20         kp[168-195] (null)
ucgd-kp            28   21/7/0/28    2:12:2   128000    690000     14-00:00:00 ucgd,c24         kp[200-227] (null)
soc-gpu-kp          4    3/1/0/4     2:14:2   256000    1800000    14-00:00:00 soc,tesla,p100,c kp[359-362] gpu:p100:2
```

Commands: squeue

- Squeue - View information about current jobs in the scheduling queue.
  - Most parameters can be combined together to form very specific queries.
  - Query jobs in queue by state: pending, running, completing, preempted, etc…
    - ‘squeue -t ${state}’
  - ‘squeue --start’ - Report the expected start time and resources to be allocated.
  - ‘squeue -w ${hostlist}’ - Query jobs associated with nodes in ${hostlist}.
  - ‘squeue -R ${resname}’ - Query jobs associated with reservation ${resname}.
  - ‘squeue -u ${user} -p ${partition} -A ${account} -q ${qos}’ - Job credential query.
  - Customize output format:
    - ‘squeue -o "%8i %12j %4t %9u %15q %12a %10g %12P %8Q %5D %11l %11L %R" --sort=-p,i’

```
JOBID   NAME     ST  USER     QOS        ACCOUNT    GROUP      PARTITION  PRIORITY  NODES  TIME_LIMIT   TIME_LEFT   NODELIST(REASON)
2648929 run.slr  R   u1069216 kingspeak  ardakani   ardakani   kingspeak  109666    1      3-00:00:00   21:17:14    kp166
2653976 jobname.err R  u0884302 mah-kp    mah-kp     mah        mah-kp     104095    1      7-04:00:00   6-09:57:59  kp294
2653781 jobname.err R  u0549046 kingspeak mah        mah        kingspeak  103691    1      3-00:00:00   2-03:43:12  kp163
2655186 jobname.err R  u0661781 wjohnson-kp wjohnson-kp wjohnson wjohnson-kp 101015   1      3-00:00:00   2-19:03:39  kp106
2653640 wrf.slurm PD   u0952282 kingspeak  zipser     zipser     kingspeak  100528    16     2:00:00      2:00:00     (Resources)
2653646 wrf.slurm PD   u0952282 kingspeak  zipser     zipser     kingspeak  100528    16     2:00:00      2:00:00     (Priority)
```

Commands: squeue (Example)

- What if I want to see all running jobs:
  - On nodes ‘ash[001-032]’...
  - Submitted by all users in account ‘smithp-ash’...
  - Using QOS ‘smithp-ash-res’...
  - With a certain job name, ‘BSFCase25Rest’...
  - In reservation ‘DAT-u0659711_HPC-4716’...
  - But only show the job ID and the list of nodes the jobs are assigned..
  - Then sort it by time remaining then descending by job ID?
  - There's a command for that!
    - ‘squeue -t running -w ash[001-032] -A smithp-ash -q smithp-ash-res -n BSFCase25Rest -R DAT-u0659711_HPC-4716 -o "%.10i %N" -S +L,-i’

```
JOBID NODELIST
1788443 ash[001-245]
```

Commands: sbatch (salloc & srun)

- sbatch parses #SBATCH in a job script and accepts parameters on CLI
  - salloc and srun accept most of the same options
  - Short and long versions exist for most options
- -N 2 # node count
- -n 8 # task count
  - default behavior is to try loading up fewer nodes as much as possible rather than spreading tasks
- -t 2-04:30:00 # time limit in d-h:m:s, d-h, h:m:s, h:m, or m
- -p p1 # partition name(s): can list multiple partitions
- --qos=standby # QOS to use
- --mem=24G # memory per node
- --mem-per-cpu=2G # memory per CPU
- -a 1-1000 # job array

Commands: sbatch (salloc & srun)

- LOTS of options: read the manpages
- Easy way to learn/teach the syntax: BYU's Job Script Generator
  - Web based tool that can create Slurm or PBS job submission scripts.
  - Demo: https://byuhpc.github.io/BYUJobScriptGenerator/
  - Code: https://github.com/BYUHPC/BYUJobScriptGenerator/
  - LGPL v3, Javascript, available on Github
  - Slurm and PBS syntax
  - May need modification by your site

Info: BYU Script Generator Example

Parameters	
Limit this job to one node:	☐
Number of processor cores **across all nodes**:	8
Number of GPUs:	2
Memory per processor core:	4 GB
Walltime:	12 hours 00 mins 00 secs
Job is a **test** job:	☐
Job is preemptable:	☑
Job is requeueable:	☑
I am in a file sharing group and my group members need to read/modify my output files:	☑
Group name (case sensitive):	MYGROUPNAME
Need licenses?	☑
Licenses:	Name lic1 Count 12 Name overpriced Count 4 Name Count
Job name:	HelloWorld
Receive email for job events:	☑ begin ☑ end ☐ abort
Email address:	me@example.com
Features:	☐ **amd [?]** Nodes avail: 0/2 Cores avail: 0/32 ☐ **avx [?]** Nodes avail: 35/320 Cores avail: 817/5120 ☑ **ib [?]** Nodes avail: 71/356 Cores avail: 1185/5488 ☐ **intel [?]** Nodes avail: 114/894 Cores avail: 2266/12076 ☐ **m2050 [?]** Nodes avail: 1/1 Cores avail: 12/12 ☐ **s1070 [?]** Nodes avail: 1/2 Cores avail: 8/16 ☐ **sse4.1 [?]** Nodes avail: 114/894 Cores avail: 2266/12076 ☑ **sse4.2 [?]** Nodes avail: 113/892 Cores avail: 2258/12060
Partitions:	☐ **p1 [?]** Nodes avail: 35/320 Cores avail: 817/5120 ☑ **p2 [?]** Nodes avail: 78/572 Cores avail: 1441/6940 ☐ **p3 [?]** Nodes avail: 2/32 Cores avail: 39/512

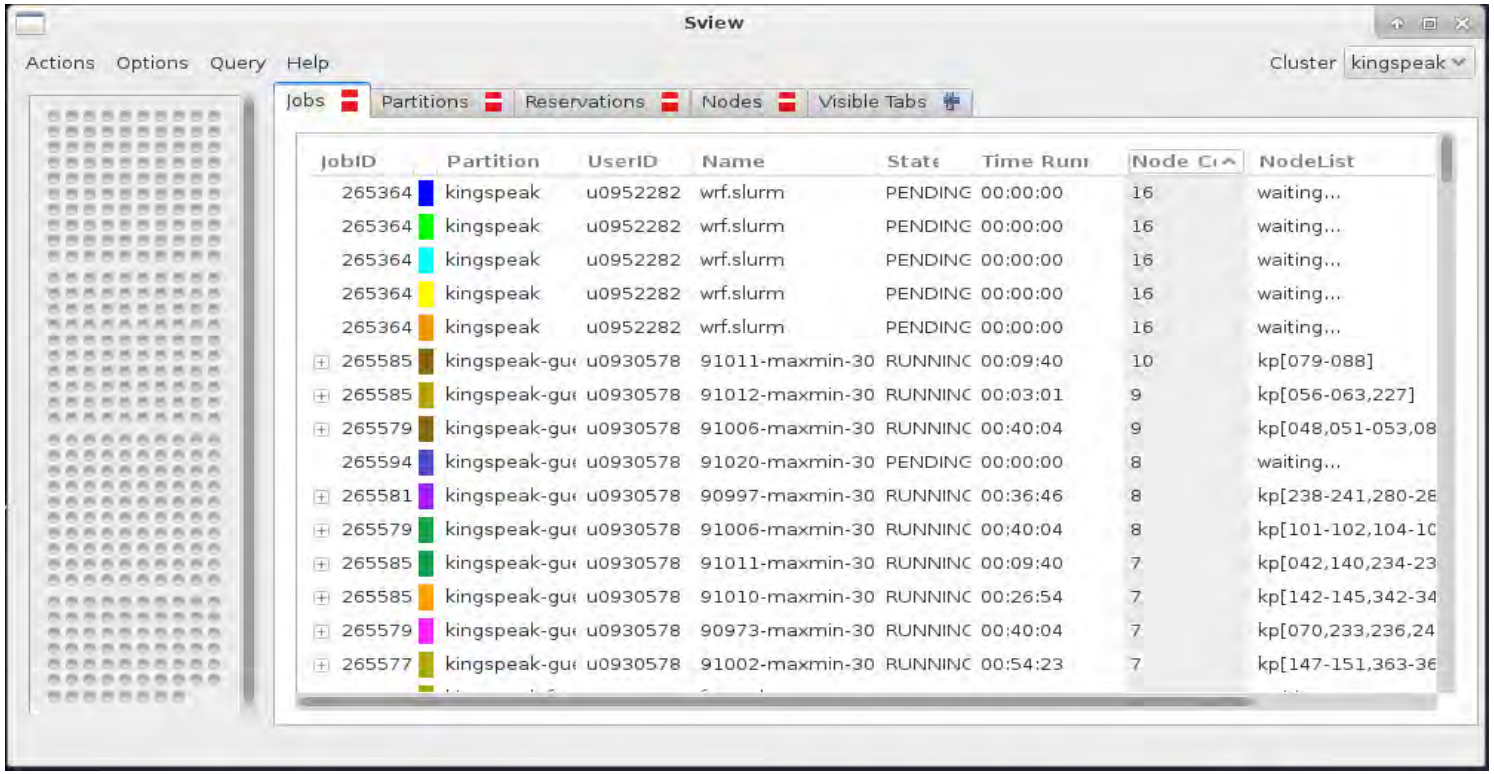
Info: BYU Script Generator Example

Script format: Slurm

```
#!/bin/bash

#Submit this script with: sbatch thefilename

#SBATCH --time=12:00:00   # walltime
#SBATCH --ntasks=8   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --gres=gpu:2
#SBATCH -C 'ib&sse4.2'   # features syntax (use quotes): -C 'a&b&c&d'
#SBATCH -p p2   # partition(s)
#SBATCH --mem-per-cpu=4G   # memory per CPU core
#SBATCH -J "HelloWorld"   # job name
#SBATCH --mail-user=me@example.com   # email address
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --qos=standby
#SBATCH --requeue   #requeue when preempted and on node failure
#SBATCH --licenses=lic1:12,overpriced:4   #format: lic1_name:lic1_count,lic2_name:lic2_count
#SBATCH --gid=MYGROUPNAME


# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
```

Commands: sview

- Graphical frontend providing ‘sinfo’, ‘squeue’, and other information.



Concept: Job Arrays

- Mechanism for submitting and managing collections of similar jobs.
  - Common size, time, limits, etc…
  - $SLURM_ARRAY_TASK_ID stores the job's index number
  - An individual job looks like 1234_7 where ${SLURM_JOB_ID}_${SLURM_ARRAY_TASK_ID}
- Range syntax support:
  - ‘sbatch --array=0-31’ - Step through range.
  - ‘sbatch --array=1,3,5,7’ - Step through specific list.
  - ‘sbatch --array=1-7:2’ - Step through range by 2’s.
  - ‘sbatch --array=0-15%4’ - Step through range, but limit concurrent running tasks.
- Interaction with array:
  - ‘scancel 1234’ - Cancel whole array.
  - ‘scancel 1234_7’ - Cancel a single job in the array.
  - ‘scancel 1234_[1-3]’ - Cancel a range within the array.

Concept: Resource Enforcement

- Slurm can enforce resource requests through the OS.
  - Linux CGroups offer the best and most complete enforcement.
    - Must insure we do not allow remote commands to “escape” cgroup control.
      - https://github.com/SchedMD/slurm/tree/master/contribs/pam_slurm_adopt
- CPU
  - task/cgroup – uses cpuset cgroup (best)
  - task/affinity – pins a task using sched_setaffinity (good but a user can escape it)
- Memory
  - memory cgroup (best)
  - polling (polling-based: race conditions, better than nothing; users can escape it)
- GRES - GPU, MIC(Xeon PHI)
  - ENV variables set by slurm.
    - CUDA_VISIBLE_DEVICES
  - Prolog/Epilog scripts.

Concept: QOS

- A QOS can be used to:
  - Modify job priorities based on QOS priority.
  - Configure preemption.
  - Allow access to dedicated resources.
  - Override or impose limits.
  - Change “charge rate” (a.k.a. UsageThreshold).
- A QOS can have limits: per QOS and per user per QOS.
- List existing QOS:
  - ‘sacctmgr -p list qos‘
- Modify existing QOS:
  - ‘sacctmgr modify qos long set MaxWall=14-0’

Concept: QOS Preemption

- Preemption is easy to configure:
  - Trigger based on QOS or Partition priority, or from Job Start Time (youngest first).
  - QOS based preemption example:
    - slurm.conf:
      - Define default behavior, can be overridden at partition/qos.
      - 'PreemptMode=CANCEL'
      - 'PreemptType=priority/qos'
    - 'sacctmgr modify qos high set priority=100000 preemptmode=cancel preempt=normal,low'
    - 'sacctmgr modify qos normal set priority=1000 preemptmode=cancel preempt=low'
    - 'sacctmgr modify qos normal set priority=1 preemptmode=cancel'
- GraceTime (optional):
  - Guarantees a minimum runtime for preempted jobs.
  - Can be set at either/both partition or qos.
    - 'GraceTime=00:15:00'

Concept: Job Prioritization

- Priority/multifactor plugin uses different components and related weights.
- Priority = sum(component_weight_int * component_value_float)
  - Weights are integers and the values themselves are floats (double?).
  - A jobs priority is an integer that ranges between 0 and 4294967295.
- Priority Factors :
  - Age: Length of time a job has been in queue and eligible to run.
    - Can be capped with 'PriorityMaxAge'.
  - Fairshare: A dynamic value influenced by recent usage and assigned shares.
  - Job Size: The number of nodes or cpus a job requests. Can favor larger or smaller.
    - Can also be further adjusted with 'SMALL_RELATIVE_TO_TIME' to help small short jobs.
  - Partition: Flat value that is then integrated into a jobs overall priority.
  - QOS: Flat value that is then integrated into a jobs overall priority.
  - TRES: Each TRES type can have it's own priority factor and be included in a jobs overall priority score.

Concept: Job Prioritization - Example

- Let's say the weights are:
  - PriorityWeightAge=0
  - PriorityWeightFairshare=10000 (ten thousand)
  - PriorityWeightJobSize=0
  - PriorityWeightPartition=0
  - PriorityWeightQOS=10000 (ten thousand)
- QOS Priorities are:
  - high=5
  - med=2
  - low=0
- userbob (fairshare=0.23) submits a job in qos “med” (qos_priority=2):
  - priority = (PriorityWeightFairshare * .23) + (PriorityWeightQOS * 2 / MAX(qos_priority)))
  - priority = (10000 * .23) + (10000 * (2/5)) = 6300

Concept: Backfill

- Slurm considers pending jobs in priority order, determining when and where each will start, taking into consideration the possibility of job preemption, generic resource (GRES) requirements, memory requirements, etc. If the job under consideration can start immediately without impacting the expected start time of any higher priority job, then it does so.
- Slurm's backfill scheduling plugin is loaded by default and does not require a lot of hand holding in most cases.
- Backfill scheduling is a time consuming operation. Depending on the characteristics of your sites queue you may find need to tune things.
- A list of 'SchedulerParameters' configuration parameters related to backfill scheduling can be found in the man page for slurm.conf.

Concept: Fair Tree Fairshare Algorithm

- Fair Tree prioritizes users such that if accounts A and B are siblings and A has a higher fairshare factor than B, all children of A will have higher fairshare factors than all children of B.
  - Benefits:
    - Higher prio account users have higher fairshare factor than users from lower prio accounts.
    - Users are sorted and ranked to prevent errors due to precision loss.
    - Account coordinators cannot accidentally harm the priority of their users relative to users in other accounts.
  - Level Fairshare Calculation: LF = S / U
    - Under-served associations will have a value greater than 1.0. Over-served associations will have a value between 0.0 and 1.0.
    - LF: The association's Level Fairshare.
    - S: Shares Norm, the associations assigned shared normalized to the shares assigned to itself and it's siblings: S = Sraw[self] / Sraw[self+siblibgs]
    - U: Effective Usage, U is the associations usage normalized to the account's usage: U = Uraw[self] / Uraw[self+siblings]

Concept: Reservations

- Slurm has the ability to reserve resources for jobs by select users and/or accounts. Reservations can also be used to for system maintenance.
  - User Reservation Example:
    - 'scontrol create reservation reservationname=DAT_u0104663 starttime=2017-07-04T08:00:00 duration=14-00:00:00 user=u0104663 nodecnt=128'
  - Jobs that are to use a reservation must add this flag: '--reservation=${RESNAME}'
    - 'sbatch --reservation=DAT_u0104663 -N128 my.slurm.script'
  - How can I allow jobs using a reservation to exceed the usual wall clock limit?
    - Create a QOS for reservations and set 'Flags=PartitionTimeLimit,RequiresReservation', also set 'MaxWall=${MaxResJobTimeLimit}'
      - Flag 'RequiresReservation' will prevent this QOS from being used when there is no reservation.
      - Flag 'PartitionTimeLimit' will allow jobs of this QOS to override the partitions usual time limit.
  - Reservations for system maintenance:
    - 'scontrol create reservation reservationname=SprintDowntime_May1st starttime=2017-05-01T06:00:00 duration=infinite nodes=all flags=maint,ignore_jobs users=root'

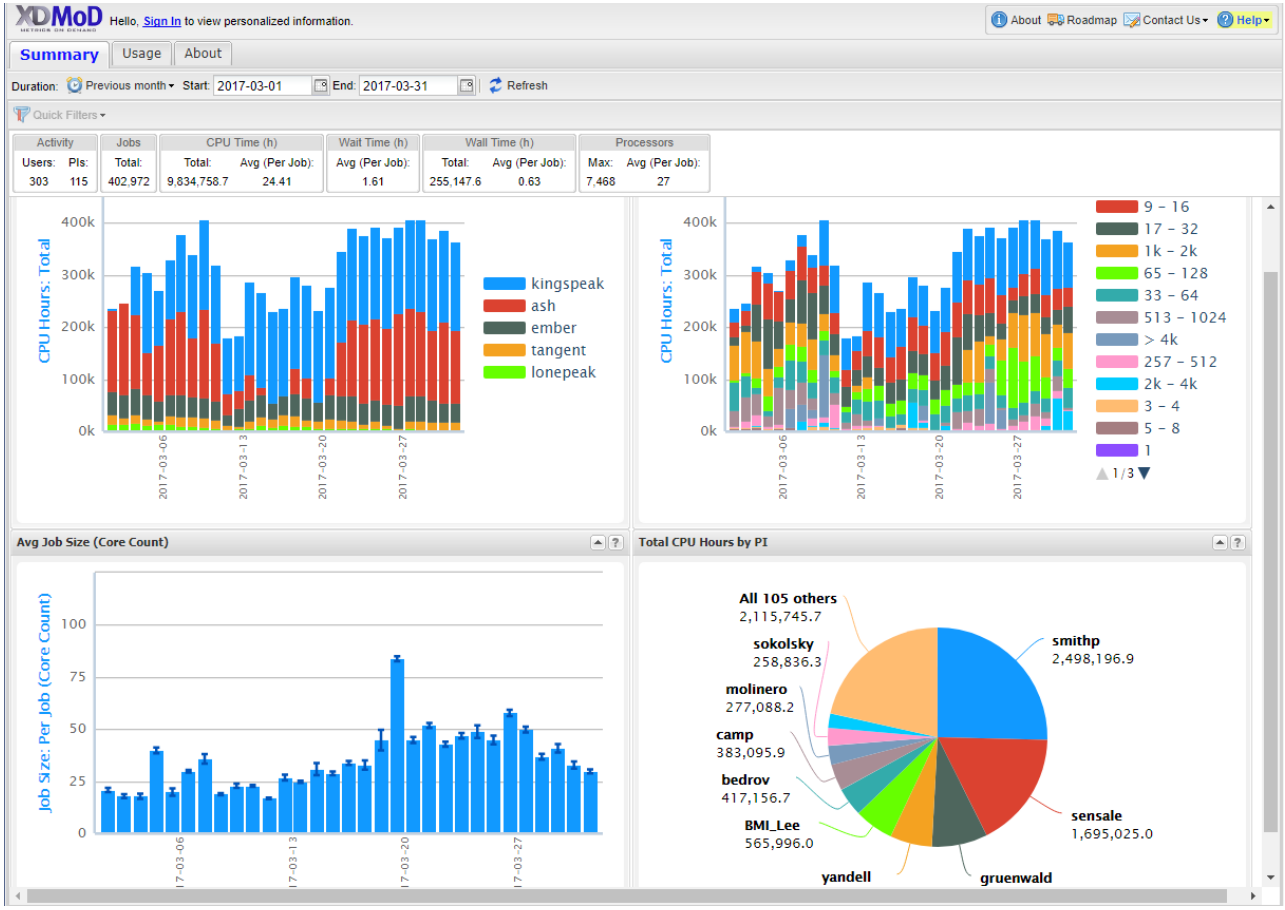
Info: Additional Job Log Record

- In addition to using slurmdbd to archive job records you can also configure slurm to save a 'similar' record in a parsable text file.
  - slurm.conf:
    - 'JobCompType=jobcomp/filetxt'
    - 'JobCompLog=/var/log/slurm/slurm.job.log'
  - Value of text file for quick info
    - Lookup last user(s) of a node(s).
    - See or review job exit codes'
    - Look for common events or other usage patterns.
    - Other...
    - NOTE: Nodes are listed in 'ranges' which can at time make simple parsing more difficult.
- JobCompLog parsing script:
  - Written by Robben Migacz (A student employee at Univ of Utah, CHPC):
  - https://github.com/robbenmigacz/SlurmJobLogQuery
  - Offers a number of search parameters and conditions to simplify usage.

Concept: Accounting with slurmdbd

- Slurmdbd is an optional daemon that provides a database backed accounting framework for Slurm:
  - Preferred DB backends are MySQL or MariaDB.
  - Current and historical records are stored for reporting, trending, etc…
  - Clusters, accounts, users, qos, and their associations can be defined in a centralized location.
    - Accounts, qos, or users can be used as ACL points for partitions and reservations.
    - Qos definitions can define priority, default/override limits, fairshare, preemption, etc...
    - Admin and Operator privilege levels can be defined.
    - Comprehensive associations can be made between clusters, accounts, qos, users.
      - Associations can also define additional limits, fairshare, def qos, available qos, etc…
  - Can be integrated with XDMoD (open.xdmod.org) to provide rich resource utilization reports via a web portal.
  - Can be used to support an XSEDE (www.xsede.org) like allocation system.

Info: XDMoD Resource Utilization Reports



Concept: Account Coordinator

- An account coordinator can do the following for users and subaccounts. under the account:
  - Set limits (CPUs, nodes, walltime, etc.).
  - Modify fairshare “Shares” to favor/penalize certain users.
  - Grant/revoke access to a QOS.
  - Hold and cancel jobs.
- BYU sets faculty to be account coordinators for their accounts.
  - End-user documentation: https://marylou.byu.edu/documentation/slurm/account-coord.

Concept: Node Health Check

- Slurm supports a “node health check” framework:
  - slurm.conf config:
    - ‘HealthCheckNodeState=${statelist}’
      - List of node states that ‘HeathCheckProgram” is called. Multiple states can be listed.
      - Supported states: ALLOC, ANY, CYCLE, IDLE, and/or MIXED.
    - ‘HealthCheckInterval=${seconds}’
      - Interval in seconds between executions of ‘HealthCheckProgram’.
    - ‘HealthCheckProgram=/path/to/nhc.sh’
      - Fully qualified pathname of script to be executed as root.
      - Run on all nodes NOT in NOT_RESPONDING state.
      - Program can verify node health and DRAIN a node or send email if a problem is detected.
      - Will kill ‘HealthCheckProgram’ if it does not terminate normally within 60 seconds.
  - LBNL Node Health Check is one possible tool to use as the ‘HealthCheckProgram’.
    - Formerly known as: Warewulf Node Health Check
    - https://github.com/mej/nhc

Concept: Troubleshooting

- What to do and where to start when things go wrong?
  - First, and possibly the most important, is to know where your logs are.
    - Slurm can use your system's log facility (syslog/rsyslog) to locally and or remotely log.
      - Log/debug levels can be increased when needed with options in slurm.conf:
        - 'DebugFlags'
        - 'SlurmctldDebug'
        - 'SlurmdDebug'
        - 'SlurmSchedLogLevel'
    - slurmctld - Involved in most activities so often proves to be a good first place to look.
    - slurmd - Each node's slurmd logs often capture node or peer level issues.
    - munge - Since munge is needed on all system running slurm daemons or where slurm commands are run these logs can be key in identifying if there is an authentication issue.
      - For example, munge can be quite sensitive to system clock consistency.
    - slurmdbd - Review slurmctld, slurmdbd, and back end database logs for issues.
- Try to understand the scope of the issue you're facing. Narrow things down to the obvious involved components and start there.
  - Keep detailed notes as you get a feel for troubleshooting things.

Concept: Troubleshooting

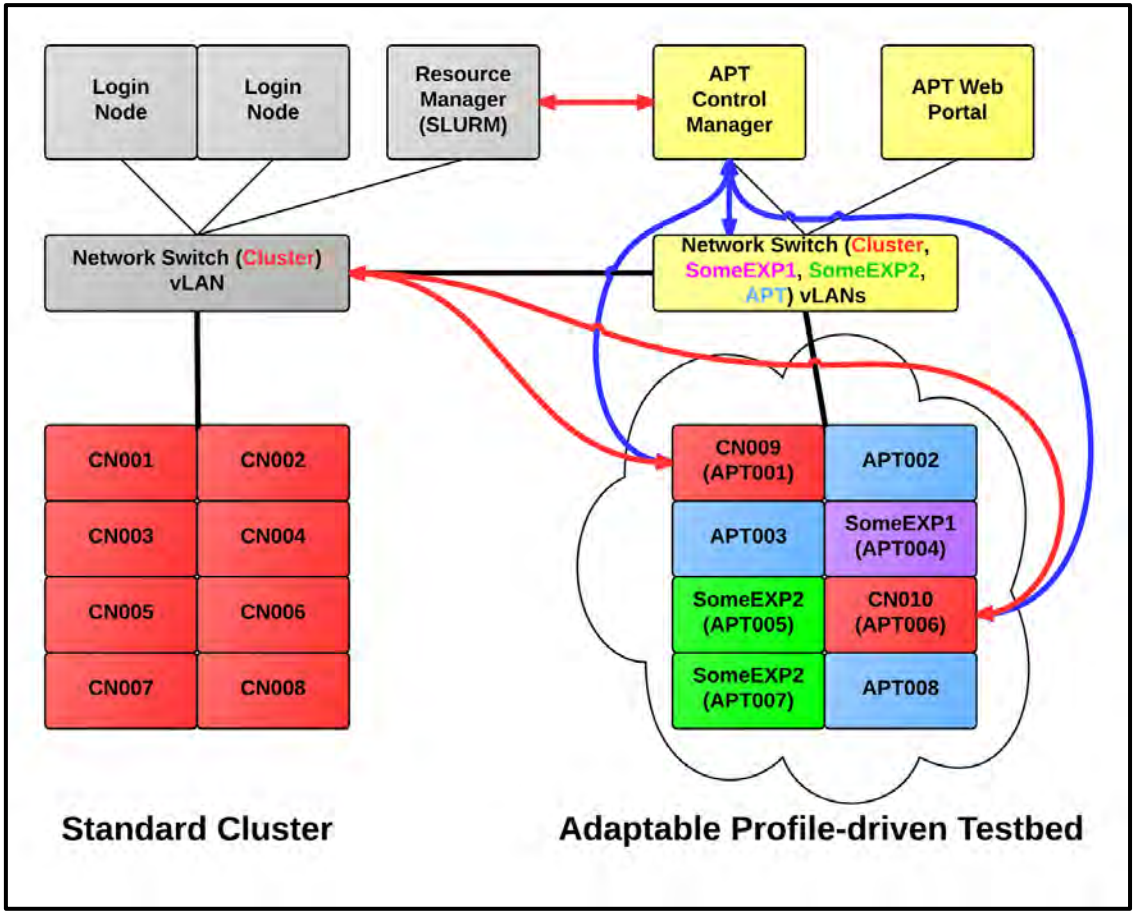
- Schedmd's Ticket System: https://bugs.schedmd.com/
  - Even if you do not have a support contract you may find valuable information by searching existing/previous tickets other have submitted.
- Additional Info: https://slurm.schedmd.com/troubleshoot.html

Concept: Slurm Elastic / Cloud / PowerSave

- Slurm has built in support for dynamic resources from both the Elastic/Cloud sense as well as for Power Saving modes of operation.
- Why?
  - Ability to burst to private or public cloud resources when needs exceed capacity.
  - Limit or reduce power consumption for resources left idle or to maintain power budget requirements.
  - Maybe you want to share a pool of resources between your HPC systems and some other usage cases.
  - Need to be able to boot nodes into alternate configs based on job requirements.
- Clusters can be composed of either or both dedicated and dynamic resources.
- Flexible boot/provision mechanism… You get to write your own! ;)

Concept: Slurm Elastic / Cloud / PowerSave

- Key configuration points:
  - 'SuspendProgram' deprovision resources.
  - 'ResumeProgram' provision resources.
  - Various timers…
  - 'SuspendTime' to deprovision idle resources.
  - Tail-End reservations.
  - Synchronize slurm and providor states.
  - Insuring nodes provision to a "healthy" state.



Info: OSG & Slurm Integration

- **A CPU cycle is a terrible thing to waste!**
- More info/documentation at: www.opensciencegrid.org
- OSG can integrated into your Slurm based cluster:
  - Maintain full local control of job prioritization and usage.
  - OSG “Glidein” jobs are submitted to slurm.
  - Glidein jobs starts up OSG framework on nodes and processes OSG workloads.
  - Able to run as low prio backfill workloads with preemption.
  - HTC vs HPC focused.
- Share what would be otherwise wasted/unused cycles with others.
- Harvest cycles from other sources for your HTC/OSG friendly workloads.

Info: User Education

- Slurm (mostly) speaks #PBS and has many wrapper scripts.
  - Maybe this is sufficient?
  - BYU switched from Moab/Torque to Slurm before notifying users of the change.
    - Yes, they were that crazy. Yes, it worked great for >95% of use cases, which was their target. The other options/commands were esoteric and silently ignored by Moab/Torque anyway.
- Slurm/PBS Script Generator available: github.com/BYUHPC
  - LGPL v3, demo linked to from github
- https://slurm.schedmd.com/tutorials.html
  - "Introduction to Slurm Tools" video is linked from there
- https://www.chpc.utah.edu/documentation/software/slurm.php
- https://marylou.byu.edu/documentation/slurm/commands
- Many other institutions have quality slurm user guides just a google search away.

Info: Support

- SchedMD: https://www.schedmd.com/
  - Excellent commercial support option from the original developers.
  - Bugfixes typically committed to github within a day.
- Other support vendors listed on Slurm's Wikipedia page.
  - Usually tied to a specific hardware vendor or as part of a larger software installation.
- slurm-dev mailing list: https://slurm.schedmd.com/mail.html
  - You should subscribe.
  - “Hand holding” is extremely rare.
  - Don't expect to use slurm-dev for support.

Closing & Contact info:

Thank you for your participation!
Any Questions?

Brian Haymore, Sr IT Arch - HPC
Center for High Performance Computing, University of Utah
brian.haymore@utah.edu