



Field Notes From the Frontlines of Slurm Support


Jason Booth
SchedMD



Slurm User Group Meeting 2020

Agenda

All times are US Mountain Daylight (UTC-6)



Time	Speaker	Title
9:00 - 9:50	Jason Booth	<u>Field Notes 4: From The Frontlines of Slurm Support</u>
10:00 - 10:25	Brian Christiansen	<u>Cloud and Stuff</u>
10:30 - 10:50	Nate Rini	<u>REST API</u>
11:00 - 11:50	Tim Wickberg	<u>Slurm 20.11 and Beyond, Open Q+A</u>

Welcome



- Four separate presentations, four separate streams
- Presentations will remain available for one week after SLUG'20 concludes
- Presentations are available through the SchedMD Slurm YouTube channel
 - <https://youtube.com/c/schedmdslurm>
- Or through direct links from the agenda
 - https://slurm.schedmd.com/slurm_ug_agenda.html

Asking questions



- Feel free to ask questions throughout through YouTube's chat
- Chat is moderated by SchedMD
- Questions will be relayed to the presenter by the moderators
 - Some may be deferred to the end if they cannot be relayed in a timely fashion
- Note there is a ~5 second broadcast delay
 - By the time you've asked your question, the presenter may have moved ahead to a different topic, and may defer the question until the end



Field Notes From the Frontlines of Slurm Support

Knowledge Nuggets

A collection of random useful Slurm knowledge nuggets



Purpose



To show you how to get the most out of your support experience and point you in a better direction while working with Slurm.

Slurm

- The use of “Slurm”, not SLURM or any other variation is preferred.
- Simple Linux Utility for Resource Management (Historic)
- Slurm in all capitals describes earlier days of the software when Slurm was just a resource manager.

Field Notes - Overview



- Random notes, observations and configuration preferences
 - Upgrading
 - Node Addition and Removal
 - Configless
 - Scalability
 - Slurm SchedulerParameters
 - Log rotation / DB archiving

Field Notes - Overview continued



- Random notes, observations and configuration preferences
 - A few scattered suggestions
 - PMIX / PMI2
 - Coredumps
 - Fair tree and classic algorithm



Upgrading

Upgrading



- There is a specific sequence to use when moving between major Slurm releases.

Upgrading

slurmdbd

>=

slurmctld

>=

slurmd

>=

commands



Must stay within 3 major releases.

E.g. {20.02, 19.05, 18.08} is okay,
but {20.02, 19.05, 18.08, 17.11} is not.

Within each major release, you can mix the
maintenance release versions without issue.
E.g. {20.02.5, 20.02.4, 20.02.3,
20.02.2, 20.02.1, 20.02.0} is okay.

Upgrading



- RPMs do make this process difficult to do with the system live.
- While we ship and support the slurm.spec file, we do not actually recommend using RPMs to install Slurm.
- We suggest structuring installs in version-specific directories, and using symlinks and/or module files to manage versions.
 - This makes rolling upgrades much simpler.

Upgrading

```
# ./configure --prefix=/apps/slurm/20.02.5/ --sysconfdir=/apps/slurm/etc/  
# ln -s /apps/slurm/20.02.5 /apps/slurm/dbd  
# ln -s /apps/slurm/20.02.5 /apps/slurm/ctld  
# ln -s /apps/slurm/20.02.5 /apps/slurm/d  
# ln -s /apps/slurm/20.02.5 /apps/slurm/current
```

Use the appropriate symlink in each service file,
and add /apps/slurm/current symlink into \$PATH
(through /etc/profile.d/ or a module file).

This makes a rolling upgrade much simpler, just
move the symlink when ready to move that component
forward onto the newer release.

Upgrading

- Backing up the MySQL database used by slurmdbd is strongly encouraged when upgrading.
 - You should probably be doing this already as part of a regular backup strategy, but this would be a good time to make sure it works.
 - For larger databases, or more unusual systems, you may want to test the upgrading/conversion on a copy of the full production database on a separate machine.
 - Older MySQL versions (5.5 and before) have had problems with later conversion processes.

Upgrading

- slurmdbd will automatically convert the MySQL schema.
 - This can take ~10-15 minutes or more depending on the size of the database.
 - Taking a backup of StateSaveLocation is also recommended.
 - Once a daemon has been upgraded, you cannot roll back to a prior major version without loss of data and your job queue



Node Addition and Removal

Node Addition and Removal



- Adding and removing nodes in Slurm is a sensitive operation.
 - This seems to cause problems for each site at least once early on.
- Certain internal data structures are built off the node list at startup, and are used within the communication subsystems.
- Changing the Node definitions, and restarting only the slurmctld, will usually lead to communication errors as messages are misrouted internally.

Node Addition and Removal



Safe procedure:

1. Stop slurmctld
2. Change configs
3. Restart all slurmd processes
4. Start slurmctld

Node Addition and Removal



Less-Safe, but usually okay, procedure:

1. Change configs
2. Restart slurmctld
3. Restart all slurmd processes really quickly

Node Addition and Removal



- We do have plans to make this less painful long-term.
- The new `cons_tres` plugin has split some of these data structures apart, and will eventually let us change this.
- This is blocked until `cons_res` is removed.



Configless

Configless



"Configless" Slurm is a feature that allows the compute nodes — specifically the slurmd process — and user commands running on login nodes to pull configuration information directly from the slurmctld instead of from a pre-distributed local file.

Configless



- There are no extra steps required to install this feature. It is built in by default starting with Slurm 20.02.
- **SlurmctldParameters=enable_configless** in `slurm.conf` and restarting `slurmctld`.

Configless



- Enabled with one of the following (slurmd's):
 - `--conf-server <srv_name>`
 - Or by setting a DNS SRV record and ensuring there is no local configuration file on the compute node.

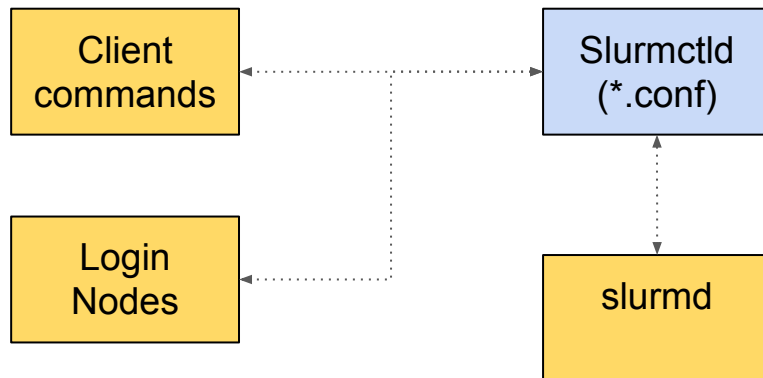
Configless - example --conf-server

- `--conf-server`
 - `slurmd --conf-server slurmctl-primary:6817`
- DNS SRV record
 - `_slurmctld._tcp 3600 IN SRV 10 0 6817 slurmctl-backup`
 - `_slurmctld._tcp 3600 IN SRV 0 0 6817 slurmctl-primary`

Configless

Key

- Configless 
- Config server 



Configless - Client Commands



- On nodes with slurmd running
 - It is assumed that you won't have default conf files
 - The commands will check the synced location for config files
 - Config files will be in SlurmdSpoolDir under the /conf-cache/, and a symlink to this location will be created automatically in /run/slurm/conf
 - If slurmd is not running it will check DNS
 - WARNING - RPC flood is possible




Configless - Client Commands

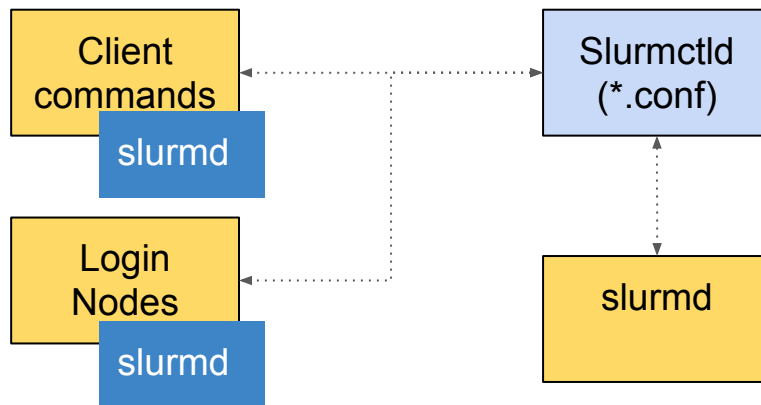


- We generally suggest that you run a slurmd to manage the configs on those nodes that run client commands, including submit or login nodes
- Without a slurmd to cache configs it can cause a bit of an RPC storm if the site has client commands request configs directly from the server.

Configless

Key

- Configless 
- Config server 
- Slurmd (conf only) 



Configless - Verifying functionality



- Config files will be in SlurmdSpoolDir under the /conf-cache/
- A symlink to this location will be created automatically in /run/slurm/conf.
- You can confirm that reloading is working by adding a comment to your slurm.conf on the slurmd node and running scontrol reconfig and checking that the config was updated.

Configless - Order of precedence



1. The slurmd --conf-server \$host[:\$port] option
2. The -f \$config_file option
3. The SLURM_CONF environment variable (if set)
4. The default slurm config file (likely /etc/slurm.conf)
5. Any DNS SRV records (from lowest priority value to highest)

Configless - supported configs

- slurm.conf
- acct_gather.conf
- cgroup.conf
- cgroup_allowed_devices_file.conf
- ext_sensors.conf
- gres.conf
- knl_cray.conf
- knl_generic.conf
- plugstack.conf
- topology.conf

Configless - known issues / limitations



- Known issues / limitations
 - 9330 - Perl API does not function with it currently (integration scripts)
 - The Included configs will NOT be shipped to the slurmds.
 - Any additional config files will need to be shared a different way or added to the parent config.
 - Adding / Removing Nodes is still not supported even with configless

Traditional Configuration Setup



- Worthy mention - traditional distributed confs
 - If you see "conf mismatch" errors, that should be a priority to fix.



Scalability

Scalability - StateSaveLocation



- Your maximum system throughput, and overall Slurm controller responsiveness under heavy load, will be governed by latency reading/writing from StateSaveLocation.
- In high-throughput (~200k+ jobs/day) environments, you may be much better off with a local NVMe drive in a single controller.
 - Especially if the alternative is an NFS mount shared with users that gets hammered frequently. You're more likely to see performance issues related to this than an outage from the controller dying.

Scalability - nss_slurm



nss_slurm is an optional NSS plugin that can permit passwd and group resolution for a job on the compute node to be serviced through the local slurmstepd process, rather than through some alternate network-based service such as LDAP, SSSD, or NSLCD.

Scalability - nss_slurm



nss_slurm is not meant as a full replacement for network directory services such as LDAP, but as a way to remove load from those systems to improve the performance of large-scale job launches

Scalability - nss_slurm

Limitations

nss_slurm will only return results for processes within a given job step. It will not return any results for processes outside of these steps, such as system monitoring, node health checks, prolog or epilog scripts, and related node system processes.

Scalability - continued

- Of course there are other things you can do
 - nscd caching
 - sssd caching
 - Tuning slurm / SchedulerParameters

Scalability - continued slurmctld

- Hardware

- Fewer faster cores on the slurmctld host is preferred
- Fast path to the StateSaveLocation
 - IOPS this filesystem can sustain is a major bottleneck to job throughput
 - At least 2 directories and two files created per job
 - The corresponding unlink() calls will add to the load
 - Use of array jobs instead of individual job records will help significantly as only one job script and environment file is saved for the entire job array.

Scalability - continued slurmctld

- Hardware - example minimum system requirements ~ 100k jobs a day / 500 nodes.
 - 16 GB RAM
 - Dual core CPU with high clock frequency
 - Dedicated SSD or NVME (statesave)
- The amount of RAM required will increase with a larger workload / node count.

Scalability - continued slurmdbd



- Hardware - minimum system requirements
 - 16-32 GB RAM
 - The RAM requirement goes up in relation to the number of jobs you wish to store/query.
 - Dedicated SSD or NVME for the database



Slurm SchedulerParameters

Slurm SchedulerParameters



- This is not an all encompassing section around the scheduler parameters, but a starting point.

Slurm SchedulerParameters

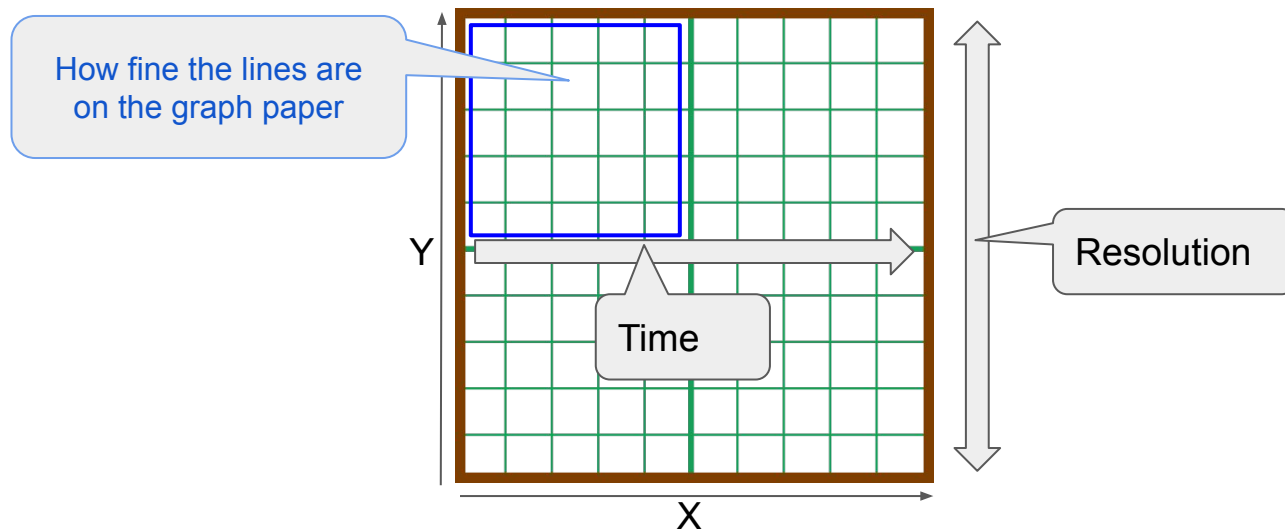


- Most common backfill problems

- bf_window set too short
 - Should be equal to the largest time limit on any partition/QOS within reason.
 - Larger values lead to higher memory consumption and slower backfill performance.
 - Too small of a value will starve large jobs indefinitely.
- bf_resolution should also be increased proportionally when adjusting bf_window.
 - For example: bf_window=11520 bf_resolution=600
 - A larger bf_resolution results in faster backfill scheduling
 - A resolution between 300-600 is the most common.
- Tiny jobs will not benefit as much from bf_resolution

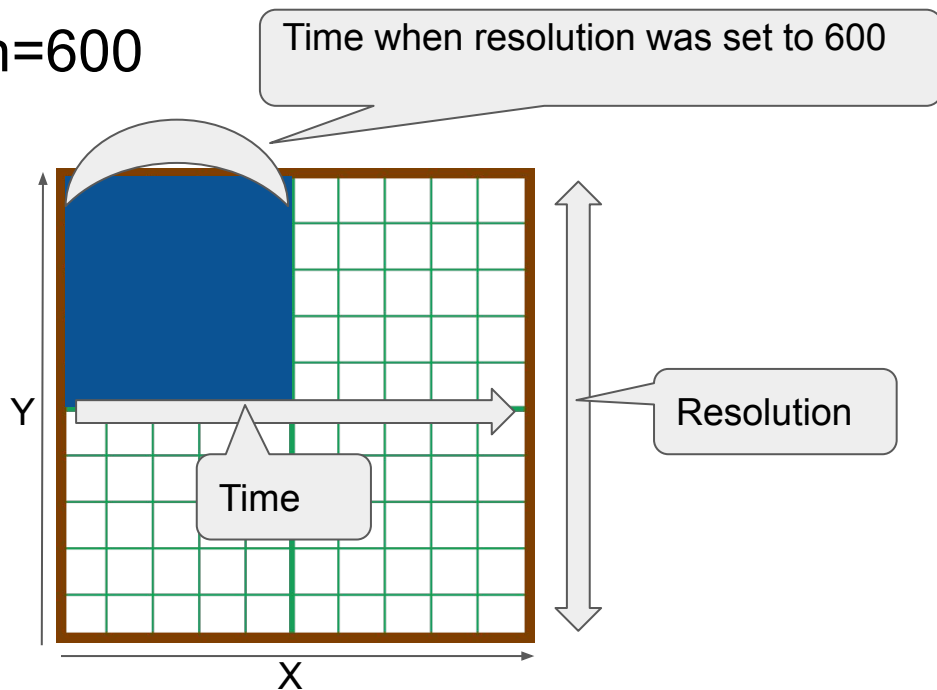
Slurm SchedulerParameters

- Example of bf_resolution=60



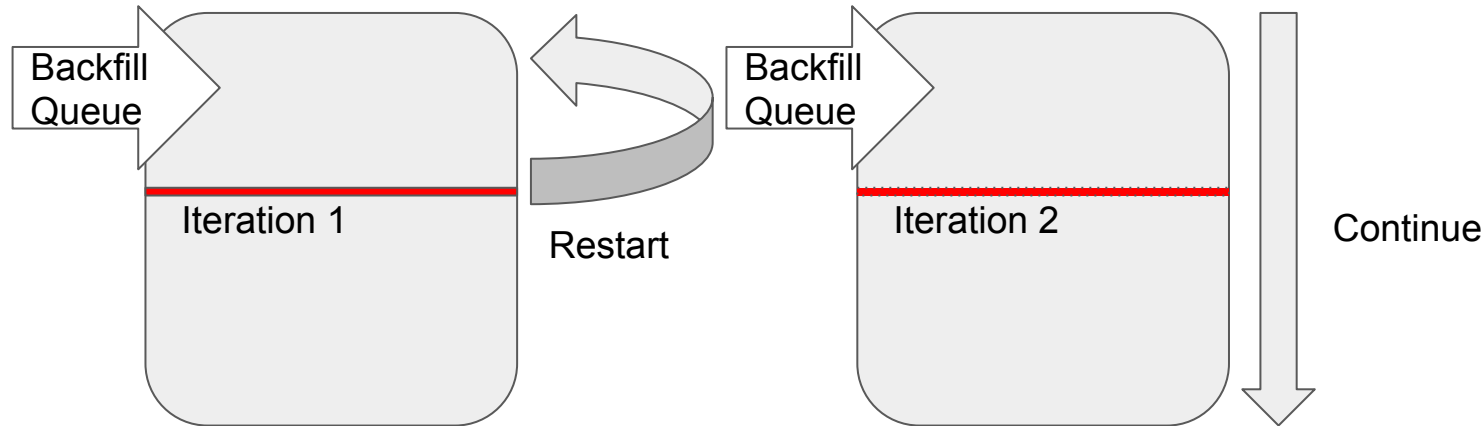
Slurm SchedulerParameters

- Example of `bf_resolution=600`
- A larger value reduces granularity
- Overall faster backfill



Slurm SchedulerParameters

- Other options to take into account
 - `bf_continue`



Slurm SchedulerParameters



- Worth mentioning
- Require all jobs to be submitted with --time, otherwise backfill will not work correctly.
- Defining a DefaultTime is the minimum you should do as many users may take the default and only use a fraction of that time. Users should be setting their time limits accurately.

Slurm SchedulerParameters

- sched_min_interval
 - microseconds
 - Faster scheduling at the cost of Higher CPU
- bf_yield_interval
 - microseconds
 - More responsiveness

```
# slurm.conf
SchedulerParameters=sched_min_interval=
50000,bf_yield_interval=1000000
```

Slurm SchedulerParameters



- There are other more fine grained options available to sites.
- We suggest opening a support case with us so that we can assist in fine tuning your cluster.



Log rotation / DB archiving

Log rotation / DB archiving



- Common throughout many sites.
- All daemons will reload the configuration on SIGHUP
 - Make sure config files are correct and synced up or you may have bad things happen
 - Crash
 - Services won't start
- SIGUSR2 - All demons will reread the log level from the configs, and then reopen the log file.
Use with logrotate

Log rotation / DB archiving



- Do not run a scontrol reconfigure for this process, as this only sends the SIGHUP and not the SIGUSR2

Log rotation / DB archiving

```
/var/log/slurm/*.log {  
    compress  
    ...  
    pkill -x --signal SIGUSR2 slurmctld  
    pkill -x --signal SIGUSR2 slurmd  
    pkill -x --signal SIGUSR2 slurmdbd  
    exit 0  
endscript  
}
```

Log rotation / DB archiving - continued



- The slurmdbd database can grow very large with time, and more so with sites that run large numbers of jobs.
 - Backing up and Truncating tables can help performance, especially when you no longer need to immediately access jobs from past years.
 - Slurmdbd has a rich set of options to Purge/Archive data
 - For sites that need historical access, this information can be moved to a non-production slurmdbd/database for semi quick access.

Log rotation / DB archiving - continued

- Purge/Archive - slurmdbd.conf
 - [##months|##days|##hours]
 - PurgeUsageAfter="12hours"
 - PurgeJobAfter="12months"
 - PurgeEventAfter

Log rotation / DB archiving - continued



- Purge/Archive - slurmdbd.conf
 - ArchiveUsage=yes|no
 - ArchiveJobs=yes|no
 - ArchiveEvents=yes|no
 - ArchiveDir
 - Default /tmp
 - ArchiveScript changes this behavior

Log rotation / DB archiving - continued



- Side note about a archive slurmdbd servers
 - This slurmdbd instance is separate from the archive options mentioned previously
 - This is an isolated instance of a slurmdbd which runs a copy or part of a copy of the production database.
 - Can be used to query historical information



A few scattered suggestions

A few random suggestions



- PMIX / PMI2 - our recommendations
 - Use PMI2 for production
 - Recommendation due to its stability and heavy industry use
 - Use PMIX for testing and massive scale projects
 - Intel supports pmi2 in recent 2019 versions but not older.

A few random suggestions - coredumps

- Core Dumps

- Make sure you know the location of your core dumps.
- If slurmctld is started with the -D option, then the core file will be written to the current working directory.
- If SlurmctldLogFile is an absolute path, the core file will be written to this directory.
- Otherwise the core file will be written to the StateSaveLocation, or `"/var/tmp/"` as a last resort.

A few random suggestions - coredumps

- Core Dumps

- SlurmUser must have write permission for the directories. If none of the above directories have write permission for SlurmUser, no core file will be produced. For testing purposes the command "scontrol abort" can be used to abort the slurmctld daemon and generate a core file.

A few random suggestions - coredumps



- Core Dumps

- There are other considerations. Please see the FAQ for further details
- https://slurm.schedmd.com/faq.html#core_dump

A few random suggestions

- Fairshare tree algorithms
 - There are differences between the **fairtree** and **classic fairshare** algorithms
 - For **classic fairshare**
 - Priority can be all over the place
 - For **fairtree**
 - Usage is contained to the account and sub account user entities.

A few random suggestions

- Users can run `sshare -l` (lowercase "L") to view their usage

```
~$ sshare -l -o Account,User,FairShare,LevelFS
      Account      User  FairShare      LevelFS
-----
root
  sci
    sci          bob   0.750000      inf
  test                                     inf
```

A few random suggestions



- Classic algorithm
 - Usage in an account affects the priority of other users in the same account by default. With DEPTH_OBLIVIOUS flag each user's fairshare is only affected by their own usage.

A few random suggestions



- **FairTree** algorithm is the default since 19.05.
- Priorities are divided into tiers by account. Priorities of users within an account will be higher/lower relative to each other, but the priority of a user in a high utilization account will never be higher than a user in a lower utilization account.

A few random suggestions



- Final thoughts on **classic** versus **fairtree**
- Should you need classic :
 - **PriorityFlags=NO_FAIR_TREE**



Field Notes From the Frontlines of Slurm Support





Questions?

Next Session



- The next presentation is by Brian Christiansen with "Cloud and Stuff"
- Starts at 10am Mountain Daylight Time (UTC-6)
- And is on a separate YouTube Live stream
- Please see the SchedMD Slurm YouTube channel for links

End Of Stream



- Thanks for watching!

Jason Booth
SchedMD

Slurm User Group Meeting 2020