Field Notes From the Frontlines of Slurm Support

Alejandro Sanchez
SchedMD

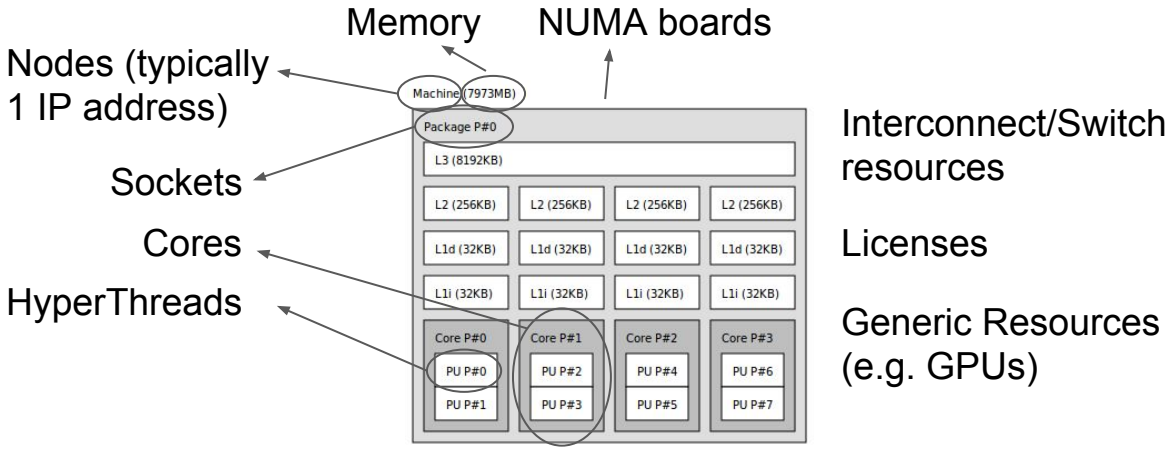
HPC Knowledge Meeting ‘18

Slurm overview

- Resource manager and scheduler
- Open source (GPL v2, available on GitHub)
- Highly scalable and configurable
- System administrator friendly
- Secure, fault-tolerant and portable
- Active global development community
- Used on many of the world's largest computers

Role of a Resource Manager

- Allocate resources within a cluster

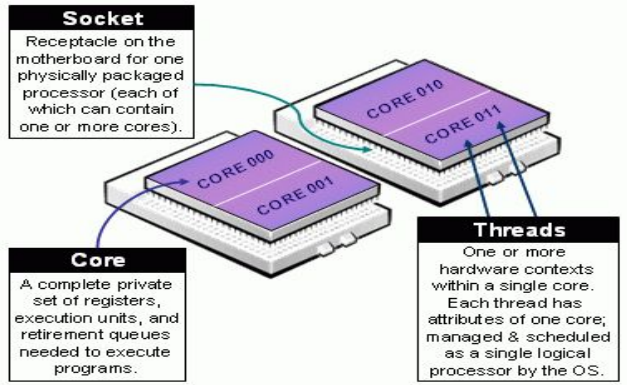


Interconnect/Switch resources

Licenses

Generic Resources (e.g. GPUs)

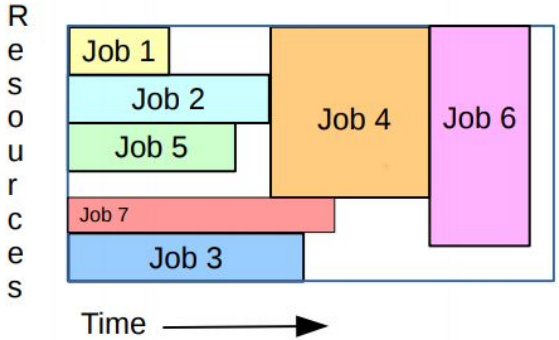
This can require an extensive knowledge about the system hardware and software (autoconf, hwloc)



- Launch and otherwise manage jobs

Role of a Job Scheduler

- When there's more work than available resources
  - Prioritize jobs based on different policies
  - Manage time on resources
  - Enforce configurable limits
  - Coordinate with the resource manager



Field Notes - Overview

- Random notes, observations, and configuration preferences.
- My opinions, not those of my employer.
  - Although if you file a support case, you'll see a lot of the same opinions repeated back. :)
- … and I can be somewhat opinionated.
  - They're not strict requirements, you may have other valid approaches.
  - Slurm has a lot of flexibility in setup and configuration to suit disparate environments, and not all or any of this may apply to you.

Field Notes - Overview

- Feel free to ask questions throughout.
  - Although I may defer more involved discussions until afterwards in the interest of time.

Log Rotation

- Common request and a best practice
- Frequently mishandled though
- If using logrotate, make sure that the correct daemons are being signaled at the appropriate time.
- slurmdbd needs to be SIGHUP'd when slurmdbd.log is rotated, several sites have had this misconfigured and lost logs.
- All the daemons will restart on SIGHUP.
  - Make sure the config files are in a usable state, or things may crash.

Log Rotation

- The 17.11 release adds SIGUSR2 to {slurmctld, slurmdbd, slurmd} to drop + reopen the log files only. Avoids reconfiguration. Strongly preferred.
- One other mistake: at least one cluster manager has shipped logrotate scripts that call *scontrol reconfigure* when rotating local log files.
  - Causes O(N) reconfigurations within the cluster in quick succession, leading to some temporary performance issues.

Fixed limit on an account's usage

- QOS setting of Flags=NoDecay
  - Allows you to establish a fixed-size allocation of resource for a group.
  - A bank account, if you will.
  - GrpTRESMins, GrpWall, UsageRaw will not decay even when using PriorityDecayHalfLife.
  - Jobs running against the QOS will eventually hit a limit.
  - Reset the RawUsage value to zero to reset, or change the limits.
  - Thanks to Josh Samuelson (U. Nebraska Lincoln) for the patch.

Ways to keep your configs tidy

- For both Nodes and Partitions there is a reserved keyword of DEFAULT.
- Used to set default options that will apply to all successive lines.
  - Also why you can't have a partition named "default".
- Next DEFAULT line will override those previously set values.
  - But not reset all of them. E.g., if the first sets Features=foo,bar, and the second sets Weight=100, all values after the second line will get both settings by default.

Ways to keep your configs tidy

- Before

```
NodeName=node0 SocketsPerBoard=2 CoresPerSocket=4 ThreadsPerCore=2 RealMemory=32768 GRES=gpu:k80:4 Weight=10
NodeName=node1 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node2 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node3 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node4 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node5 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node6 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node7 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node8 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node9 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node10 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node11 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
```

Ways to keep your configs tidy

- After

```
NodeName=DEFAULT SocketsPerBoard=2 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node0 CoresPerSocket=4 GRES=gpu:k80:4
NodeName=node[1-5] CoresPerSocket=6
NodeName=node[6-11] CoresPerSocket=8 RealMemory=65536 Weight=2
```

Allocating Memory

- If you use CR_{CORE,CPU,SOCKET}_MEMORY, make sure to set DefMemPerCPU or DefMemPerNode.
  - Choose one or the other. DefMemPerCPU is usually preferable.
  - Set DefMemPerCPU roughly equal to Memory / CPUs.
    - But preferably round it down a bit. Usually to the closest GB.
    - Helps fill the node up better
  - With no setting, the first job allocated that did not specific --mem will receive all memory on the node.
    - Leaving 0 MB free for other jobs.

Node configuration

- Must have all settings less than or equal to the available hardware in the node.
- If a node comes up with more sockets/threads/cores/memory that's okay, although Slurm will still schedule based on the configured values.
- I encourage sites to round their Memory values down to the nearest GB though.

Node configuration - continued

- Rounding down the memory value does two things:
  - Avoids the node being marked down if the BIOS reports slightly less memory.
    - This can happen due to a firmware change, DIMM replacement, or even kernel upgrade. If the node has 4MB less than configured, it'll stay offline.
    - Gives the OS some breathing room when you're using CR_*_MEMORY.
      - If Slurm can't allocate that space, it'll never be divvied up and assigned. Helps avoid the dreaded OOM.

Node configuration - continued

- Alternate approach:
  - The MemSpec limit setting on a Node can set aside an amount of space for the slurmd process and OS.
    - The allocatable memory will be RealMemory - MemSpecLimit
    - The slurmd processes cgroup will be limited to that amount of space, so make sure it’s not too small though.

Node configuration

- You cannot add or remove nodes on the fly.
- Internal data structures related to scheduling and communication are designed around fixed-size bit field for performance reasons, and cannot be changed without a restart.

Constraining memory appropriately

- There are two mechanisms
  - JobAcctGather (poll mechanism)
  - TaskPlugin (kernel interruption)
- Our recommendation is to disable JobAcctGather memory enforcement (and just use it for account gathering) and enforce it with TaskPlugin task/cgroup.

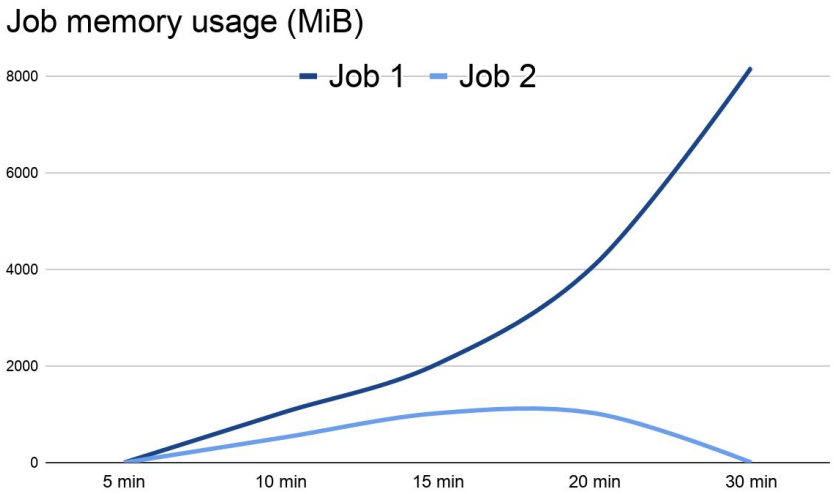
slurm.conf:

```
MemLimitEnforce=no
JobAcctGatherParams=NoOverMemoryKill
TaskPlugin=task/cgroup (although stacking
'task/cgroup,task/affinity' is better)
```

cgroup.conf:

```
ConstrainCores=yes
ConstrainRAMSpace=yes (AllowedRAMSpace)
ConstrainSwapSpace=yes (AllowedSwapSpace &
MemorySwappiness)
TaskAffinity=no
```

Constraining memory appropriately



- Usage > limit → memory.failcnt++
- Before 17.11.3, this sufficed to mark the job as OutOfMemory.
- Problem: some use-cases reported where Kernel reclaimed memory pages and process finished successfully, but the job was incorrectly marked as OutOfMemory.
- Solution1: cgroups-v2 comes with event counters:
  - low, high, max, oom, oom_kill
  - But many sites run OS with not so recent Kernel versions...
- Solution2: cgroups-v1 + eventfd() on memory.oom_control to improve live detection of OOM_KILL events.

Partition Priority

- Show of hands - how many people use the Priority setting on a partition in conjunction with the PriorityWeightPartition setting to adjust user job priorities?

Partition Priority

- Yes - this is a trick question. It's not doing what you may think it is. The change to user priorities is inconsequential.
- The scheduler generates a single queue in this order:
  1. Preemption
  2. Advanced reservations
  3. Partition PriorityTier
  4. Job Priority
  5. JobId

Job Priority =
PriorityWeightAge * AgeFactor +
PriorityWeightFairShare * FairShareFactor +
PriorityWeightJobSize * JobSizeFactor +
PriorityWeightPartition * PriorityJobFactor +
PriorityWeightQOS * QOSFactor +
PriorityWeightTRES * TRESFactor.

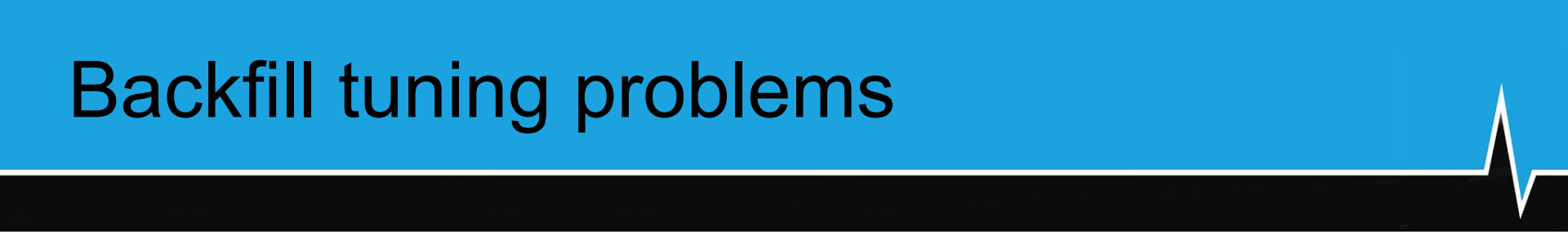
Partition Priority

- So the only effect of the PriorityWeightPartition has been to change the numbers around.
- But priority numbers only matter *when compared to other jobs at the same Tier.*

Partition Priority

- The 16.05 release split the Priority setting on a Partition into two parts:
  - PriorityTier - only sets this preemption tier, no effect on job priority values.
  - PriorityJobFactor - normalized, then used with PriorityWeightPartition to change job priority values. Does not set the Tier.
- For backwards compatibility, if Priority is set, it's applied to both PriorityTier and PriorityJobFactor.
  - Although the PriorityJobFactor setting doesn't accomplish much.

Backfill tuning problems



- Most common backfill problem: bf_window set too short.
- Should be equal to the largest time limit on any partition/QOS, within reason. Max of ~ 2 weeks is my preference.
  - Larger values lead to higher memory consumption, and slower backfill performance.
- Too small of a value will starve large jobs indefinitely.
- If you increase bf_window it is also advisable to proportionally increase bf_resolution.

Backfill tuning problems

- Other options to take into account:
  - bf_continue
  - bf_min_[prio|age]_reserve (optimize system utilization)
- And obviously… force all jobs to be submitted with a --time, otherwise backfill won't perform.
  - Defining a DefaultTime tends to end up with users not explicitly specifying a time and all will end up with the same TimeLimit. Avoid this.

On performance, time, and such matters

- The time() and gettimeofday() syscalls need to be fast for good slurmctld performance.
- slurmctld calls them **very** frequently.
- Certain VM platforms do not handle this well, as they disable the Linux vDSO that is usually in use.
  - The vDSO avoids a context switch to the kernel and back and makes time() a direct memory reference. slurmctld effectively assumes this is in use.
  - Xen especially guilty of this.

On performance, time, and such matters

- Hardware:
  - Fewer faster cores on the slurmctld host is preferred.
  - Fast path to StateSaveLocation.
    - IOPS this filesystem can sustain is a major bottleneck to job throughput.
      - At least 1 directory and two files created per job.
      - And the corresponding unlink() calls will add to the load.
    - Although using job arrays instead of individual job records will help significantly, as only one job script and environment file is saved per entire array.

On performance, time, and such matters

- slurmctld is highly threaded.
  - 10 - 100’s of threads in use at any time.
- Unfortunately, slurmctld is not highly concurrent.
  - Internal data structure locking will limit the concurrent thread count in a lot of circumstances to… one thread.
  - We’re working on this, and have made some minor improvements lately.
- Hardware:
  - Fewer, faster cores on the slurmctld host is vastly preferred.
  - Dual 16-core Xeons are overkill, the higher clock rate (and cheaper!) 6-cores will perform

Job Submit Plugins

- Allow you to add arbitrary business logic to filter/deny/modify user jobs at submission and update time.
- Run within the slurmctld process, and have access to all internal data structures.
  - One caveat: they can also potentially corrupt anything and everything if misdesigned… be careful.
- Optional job_submit.lua plugin uses an interpreted and “safe” script provided by the site, rather than needing to code in C.

Job Submit Plugins

- Lua (or a new C plugin) can do whatever you want.
- Potentially call out to external APIs.
- Enforce whatever arbitrary logic you like…
  - … For instance forcing users to submit their jobs with a TimeLimit.

Job Submit Plugins

- Caveats
  - Remember that I mentioned earlier that slurmctld is highly threaded, but not highly concurrent?
  - For safety reasons, the job_submit plugin operates with write locks held on the two main internal data structures.
  - So no other threads are able to run concurrently.
  - Make the scripts fast, or pay the price in system throughput and responsiveness.

SlurmDBD is hung

- Most common source of problems with slurmdbd is from overly-aggressive sacct requests.
  - Caused by mistake, or bad scripting.
- Depending on your system, ‘sacct -S 2010-01-01’ may need to return millions of rows.
  - May push the slurmdbd host out of memory, or just take an unreasonably long time to complete.

- New slurmdbd.conf option of MaxQueryTimeRange allows admins to restrict the maximum time range users can query in a single command. Requests over > 3GB will fail as well.

```
test@box:~/t$ sacct -S 2007-01-01
JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
sacct: error: slurmdbd: Too wide of a date range in query
test@box:~/t$
```

High-Availability

- My preferences:
  - Setup the primary and backup slurmctld hosts.
    - Should be directly attached to a fast shared filesystem for the StateSaveLocation
  - Co-locate the slurmdbd with its own MariaDB instance. Do not worry about running a backup slurmdbd.
  - If short on machines, host the backup slurmctld on the same machine as slurmdbd. But keep them split by default.

High-Availability

- Common issues:
  - StateSaveLocation needs to be available (and fast!) on both primary and backup at the same time.
  - If the StateSaveLocation filesystem crashes, your cluster will be unavailable. Choose setup carefully, and avoid introducing unneeded complexity.
  - E.g., I would prefer a single slurmctld with a local SSD in it for a lot of environments, over a more elaborate deployment involving DRDB + GFS.

High-Availability

- Similarly, for slurmdbd the real failure domain is the MySQL/MariaDB installation.
  - In my experience, HA approaches to MySQL/MariaDB are more likely to introduce outages than the hardware failures they're trying to prevent.
  - Just to be clear - you should still keep backups.
  - slurmctld can survive and keep the cluster running while spooling messages destined for the slurmdbd for a while.
    - Although this depends on your environment, and the outstanding messages in the spool are currently capped by: MAX(10000, 4xNodes + 2xMaxJobCount).

High-Availability

- Triggers within slurmctld can help supplement your monitoring.
- And yes, you should probably be doing some sort of cluster monitoring.
- The HealthCheck[Program|Interval] options can be useful to detect compute node problems and take appropriate actions such as marking nodes to DRAIN.

X11

- Since 17.11 Slurm internally supports x11 forwarding from compute nodes and an external SPANK plugin is not necessary anymore.
  - Requisites:
    - libssh2 package installed on the cluster.
    - libssh2-devel package on the node where Slurm is built.
  - Configuration:
    - PrologFlags=x11 in slurm.conf.
    - scontrol reconfigure
    - Setup password-less RSA keys for all the users.

SLUG'18

- Slurm User Group Meeting
  - 25th-26th September, 2018
  - Madrid, Spain
  - Host: CIEMAT
  - More info:
    - https://slurm.schedmd.com/meetings.html
    - https://slurm.schedmd.com/slurm_ug_cfp.html
    - https://slurm.schedmd.com/slurm_ug_agenda.html