Field Notes From A MadMan

Tim Wickberg
SchedMD

Slurm User Group Meeting 2019

Field Notes - Overview

- Best practices, interesting configuration and deployment tricks, common pitfalls, future potential projects, and other random content that didn't have a better home.
- Also: this is what happens when you let the sales folks pick a title for you.
- TLDR: an ~hour of me rambling. :)

Field Notes - Overview

- Successor to last years talk - "Field Notes Mark Two: Random Musings From Under A New Hat"
- https://slurm.schedmd.com/SLUG18/field_notes2.pdf
- Which was a successor to last-last years talk - "Field Notes From The Frontlines of Slurm Support"
- https://slurm.schedmd.com/SLUG17/FieldNotes.pdf
- … which serves, in part, as a best-practices guide
- Some of the older content is intentionally re-hashed here

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., {19.05, 18.08, 17.11, 17.11} is okay,
but {19.05, 18.08, 17.11, 17.02} is not.

Within each major release, you can mix the maintenance release versions without issue.
E.g., {19.05.0, 19.05.3, 19.05.2, 19.05.0} 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.

Example installation hierarchy

```
# ./configure --prefix=/apps/slurm/19.05.3/ --sysconfdir=/apps/slurm/etc/
# ln -s /apps/slurm/19.05.3 /apps/slurm/dbd
# ln -s /apps/slurm/19.05.3 /apps/slurm/ctld
# ln -s /apps/slurm/19.05.3 /apps/slurm/d
# ln -s /apps/slurm/19.05.3 /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

Accounting

- Why doesn't Slurm just adopt gold?
  - Doesn't fit Slurm's approach to tracking and split responsibility between slurmdbd and slurmctld
  - And doesn't philosophically match up with our goals
  - There are ways to emulate this if you really want to

Accounting

- IMNSHO, fixed allocation of cpu-minutes does not match reality
- Computers don't work that way
- CPU time is used or is lost, you can't set it aside for later
- There is no "cpu-hours" coal-pile out back that you're shoveling into the cluster on demand

Accounting

- Fairshare is, IMNSHO, a better mapping to reality
  - Adapt priorities in response to use patterns
  - Set high level targets, and real-world usage patterns will track towards your ideal balance over the long term
- Avoids problems as you near the credit reset period
  - Nearly every system running on fixed allocations ends up with liabilities massively exceeding the remaining possible usage
  - At which point you're falling back to other prioritization schemes anyways

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 as 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 cons_tres plugin will eventually let us change this
- But we need to get rid of cons_res first

Node Naming Conventions

- IMNSHO, simpler is better
  - node[0001-1000] is nice
- Your users shouldn't care what the nodes are called
- Contiguous ranges make the configs much cleaner
  - As well as the output from 'sinfo' and other commands

Node Naming Conventions

- I see a lot of sites injecting node locality info into the naming convention
  - And I don't care for it
  - You users don't care where in the datacenter your hardware lives
  - And whomever handles hardware maintenance should be able to work out the mapping for themselves
  - Use Features to add rack / chassis annotations if necessary

Node Naming Conventions

```
PartitionName=general Nodes=nid[00009-00063,00072-00075,00077-00127,00137-00191
,00193-00195,00200-00203,00208-00255,00261-00263,00268-00319,00321-00323,00328-
00383,00393-00447,00456-00459,00461-00463,00465-00466,00468-00511,00521-00575,0
0577-00579,00584-00587,00592-00639,00645-00647,00652-00703,00705-00707,00712-00
767,00776-00831,00845-00895,00897-00899,00901-00959,00961-00963,00968-00971,009
76-01023,01029-01031,01036-01091,01096-01099,01105-01106,01108-01151,01161-0121
5,01224-01231,01233-01234,01236-01279,01281-01283,01285-01343,01345-01347,01352
-01355,01360-01407,01413-01415,01420-01475,01489-01490,01492-01535,01545-01599,
01608-01611,01613-01663,01665-01727,01729-01731,01736-01739,01744-01791,01797-0
1799,01804-01859,01864-01867,01873-01874,01876-01919,01929-01983,01992-02047,02
049-02051,02053-02111,02113-02115,02120-02123,02128-02175,02181-02183,02188-022
43,02248-02251,02257-02258,02260-02319,02324-02379,02384-02443,02452-02507,0251
2-02575,02580-02635,02640-02703,02708-02763,02768-02827,02836-02891,02896-02959
,02964-03019,03024-03087,03092-03147,03152-03211,03220-03275,03280-03343,...
```

Node Naming Conventions

- I don't expect you to change your existing naming schemes
  - Just something to think about on your next install
- While I'd rather see simple naming, I have been thinking about how to make this slightly less painful
  - One potential idea to simplify things follows

NodeSet syntax for slurm.conf

- Possible new "NodeSet" syntax for slurm.conf
- Define a NodeSet as a list of Nodes
- And then use the node sets interchangeably with the node names to define partitions
  - Rather than error-prone copy+pasting long lists

NodeSet syntax for slurm.conf

```
NodeName=node[0001-0005] Gres=gpu:v100:2,tmpdisk:5g
Features=e2620v4,xeon,qdr,v100
NodeName=node[0006-0010] Features=e2620v5,xeon,qdr
NodeName=node[0011-0015] Gres=gpu:v100:2,tmpdisk:5g Features=e2620v6,xeon,ndr

NodeSet=v100nodes Feature=v100
NodeSet=random Nodes=node[0001,0002,0003,0005,0008]
NodeSet=imaginary Feature=e2620v6

PartitionName=v100 Nodes=v100nodes
# these two sets overlap in part, but the slurmctld would de-duplicate for us:
PartitionName=somestuff Nodes=random,imaginary
```

Demo #1

NodeSet syntax for slurm.conf

- Would this be useful?
- One other potential idea:
  - Add "not" syntax to drop nodes back out of the set, something like "PartitionName=foo Nodes=group,!node1"
- Is there any other syntax extensions that would be helpful?

Development and Patch Submission

- We love to see external submissions, and do try to provide timely feedback regardless of who submitted
  - You do not need to be a SchedMD customer to submit patches
- While SchedMD provides the majority of development, some major subsystems have originated externally
  - E.g., pam_slurm_adopt was conceived by BYU
  - cli_filter was NERSC's idea

Development and Patch Submission

- Please read CONTRIBUTING.md for style guidelines and notes on submission and review.
- Patches that add new functionality should be of general interest and usable by other sites.
  - We generally do not land features only usable by a specific site.
- New functionality is only permitted in the next release; please make sure patches are based on the master branch.

Development and Patch Submission

- Bug fixes should be based on the most recent stable release branch - right now that's slurm-19.05
- Please be patient with the review process; we do thoroughly review all patches, and this can be a significant investment of time and effort on our end
- Minor style corrections we will generally address; more significant issues we'll bring back to you to discuss and correct

Development and Patch Submission

- Documentation is critical, and must land alongside any new features.
  - Testsuite additions are also appreciated.
- When submitting, please attach patches to a new bug in our Bugzilla instance.
  - https://bugs.schedmd.com
  - Set the Severity to “C - Contributions”
  - ‘git format -am’ strongly preferred for patches.

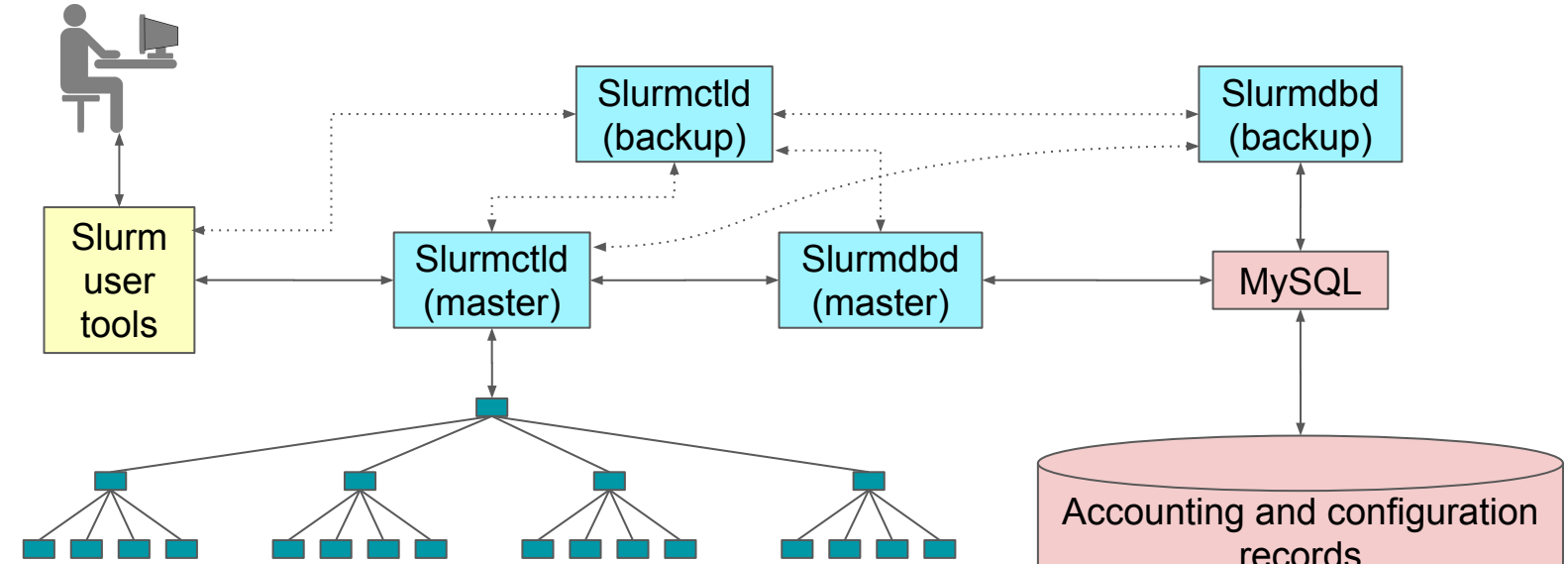
macOS build?

- Would anyone use a native macOS build for testing if available?
  - It's… 80% working today
    - With a few out-of-branch patches the user commands do work
  - Shared libraries work in frustratingly different ways, lead to annoying symbol resolution issues
    - And seemingly insane hacks like this one:

macOS is weird

```
#ifndef strong_alias
#  if USE_ALIAS
#    define strong_alias(name, aliasname) \
     extern __typeof (name) aliasname __attribute ((alias (#name)))
#  else
#    define strong_alias(name, aliasname) \
     __asm__(".global _" #aliasname); \
     __asm__(".set _" #aliasname ", _" #name); \
     extern __typeof (name) aliasname
#  endif
#endif
```

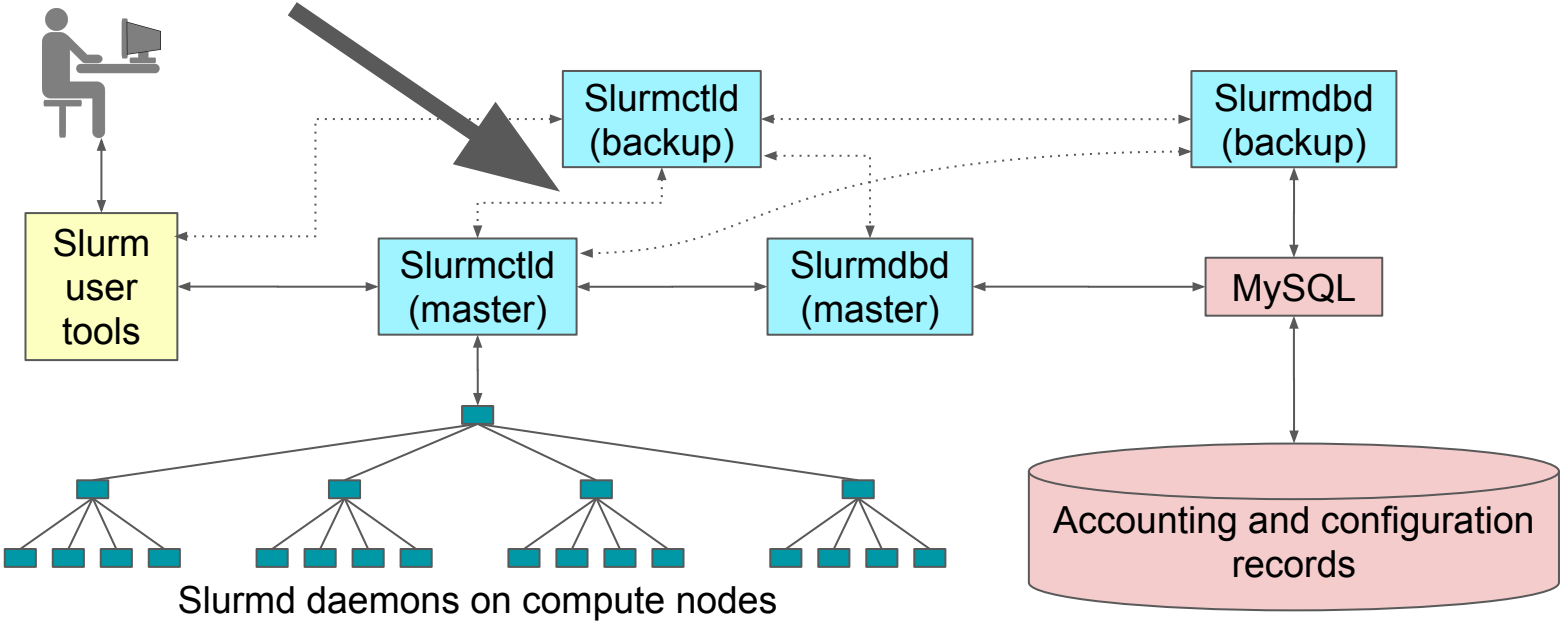
Cluster Architecture - Typical Linux Cluster



Slurmd daemons on compute nodes
(note hierarchical communications with configurable fanout)

Cluster Architecture - Typical Linux Cluster

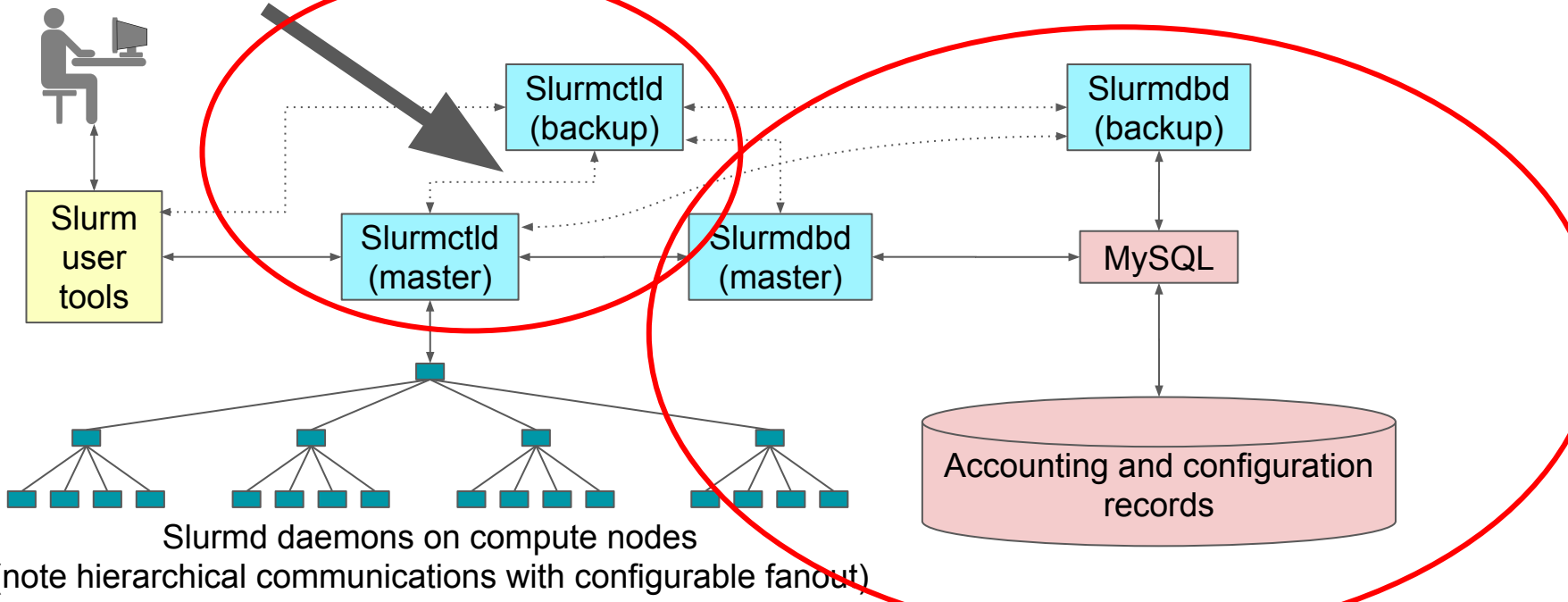
StateSaveLocation is hiding in here, and establishes your failure domain for slurmctld.



Slurmd daemons on compute nodes
(note hierarchical communications with configurable fanout)

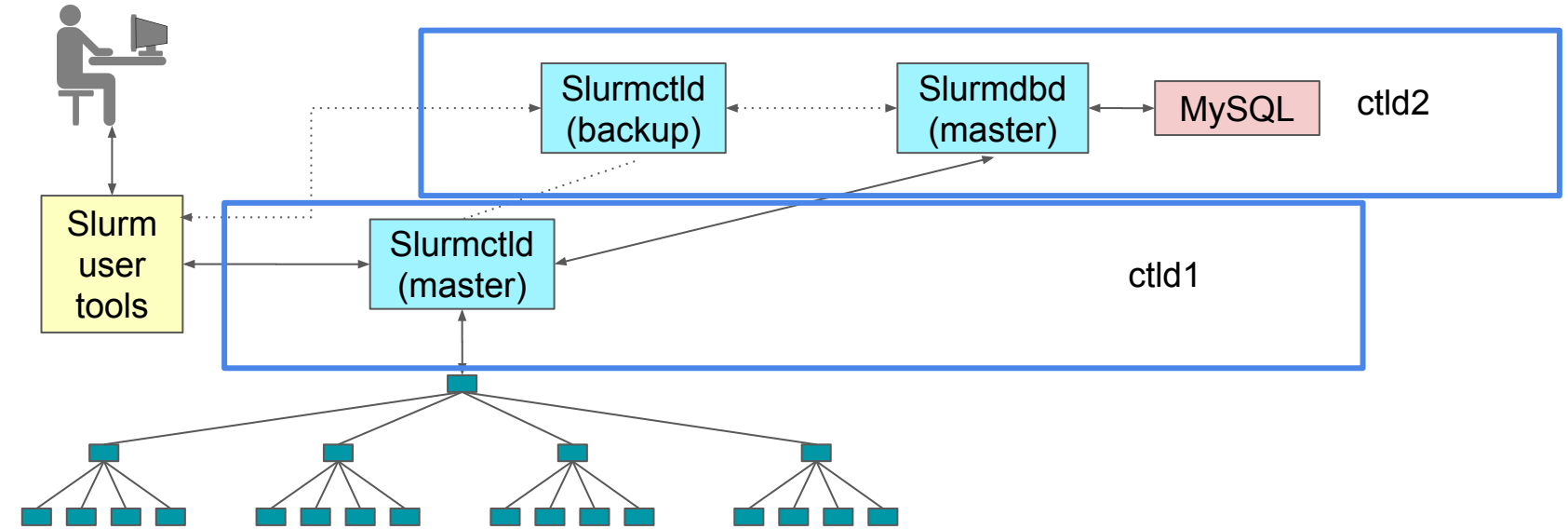
Cluster Architecture - Typical Linux Cluster

StateSaveLocation is hiding in here, and establishes your failure domain for slurmctld.



Slurmd daemons on compute nodes
(note hierarchical communications with configurable fanout)

My Preferred Deployment Pattern



Slurmd daemons on compute nodes
(note hierarchical communications with configurable fanout)

Justification

- Separating slurmctld and slurmdbd in normal production use is recommended.
- Master/backup slurmctld is common, and - as long as the performance for StateSaveLocation is kept high - not that difficult to implement.

One aside on 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 frequently hammered… you're more likely to see performance issues related to this than an outage from the controller system dying.

Justification

- For slurmdbd, the critical element in the failure domain is MySQL, not slurmdbd. slurmdbd itself is stateless.
- slurmctld will cache accounting records (up to a limit) if slurmdbd is unavailable. This can be hours+ to days+ depending on your system without data loss.

Justification

- IMNSHO, the additional complexity of a redundant MySQL deployment is more likely to cause an outage than it is to prevent one.
- So don’t bother setting up a redundant slurmdbd, keep slurmdbd + MySQL local to a single server.

Related Networking Notes

- Semi-frequently reported issue: changes made through sacctmgr aren’t immediately showing up in slurmctld.
- But then restarting slurmctld fixes this somehow?

Related Networking Notes

- This is almost always an issue with a firewall in between slurmctld and slurmdbd.
- slurmdbd opens a new connection to slurmctld to push changes.
- If you've firewalled that off, the update will not be propagated.
- Restarting slurmctld forces it to open a connection to slurmdbd - the other direction - and pull down a full copy.

Related Networking Notes

- slurmctld ⇒ slurmdbd is the prevalent pattern
  - So it is easy to overlook issues in the other direction
  - And everything else will work.
- The slurmdbd logs should be telling you this is an issue… but easy to miss this as well.

Demo B

- Cross-platform support
- You can mix and match without issue
  - Well, from Slurm's perspective
  - Your application still needs to deal with the different architectures somehow

Demo B

Node	Architecture	Endian	Word Size
zoidberg	amd64	little	64
thirtytwo	i386	little	32
g5	ppc32	big	32
blackbird	ppc64le	little	64
dali	armv7l	little	32
cavium	aarch64	little	64

Demo B

This is "blackbird"

Demo B

This is "blackbird"

It's my POWER9 development workstation

Demo B

This is "blackbird"

It's my POWER9 development workstation

Yes, that's a Sun Ultra 24 chassis

Demo B

This is "blackbird"

It's my POWER9 development workstation

Yes, that's a Sun Ultra 24 chassis

… did you know Google Slides can do GIFs?

Demo III

- What if “pay-to-play” on your cluster had a physical manifestation?

Demo III

- Re-Introducing the Prioramatic 5000™

Demo III

- Things to highlight:
  - The "Submit" button is queuing up new jobs through Go making a REST call against a local slurmrestd
  - It's living outside the MUNGE security realm
    - If someone steals it, at least they won't have my MUNGE key

```
func submit_job() {
        b := []byte(`{"job": {"ntasks": 1, "name": "cashbox-test", "nodes": 1,
"current_working_directory": "/tmp/", "environment": {"PATH":
"/bin:/usr/bin/:/usr/local/bin/"}}, "script": "#!/bin/bash\necho testing"}`)

        client := &http.Client{}
        req, err := http.NewRequest("POST", "http://localhost:10000/slurm/v1/job/submit",
bytes.NewReader(b))
        req.Header.Add("X-SLURM-USER-NAME", "tim")
        req.Header.Add("X-SLURM-API-KEY", "testing")
        req.Header.Add("Content-Type", "application/json")
        req.Header.Add("Accept", "text/json")
        resp, err := client.Do(req)

       if err != nil {
                fmt.Printf("The HTTP request failed with error %s\n", err)
        } else {
                data, _ := ioutil.ReadAll(resp.Body)
                var f interface{}
                err = json.Unmarshal(data, &f)
                m := f.(map[string]interface{})

                str := fmt.Sprintf("%d", int((m["job_id"]).(float64)))
                newjob.SetLabel(str)
        }
}
```

Demo III

- Things to highlight:
  - Demo is showing three approaches to changing relative user priorities in an automatic way
    - $1 - Setting a site factor to boost job priority
    - $50 - Setting a QOS with a higher priority
      - One that the user does not have access to already
    - $100 - Adjusting the Association Factor through SlurmDBD
      - Permanent for all their jobs, not just this one

Demo III

- Things to highlight:
  - This is also why I don't develop GUIs

Questions?

Closing Remarks

Danny Auble
SchedMD

Slurm User Group Meeting 2019

Slurm @ SC19

Slurm is at Booth 1571

SC19

Denver, CO | hpc is now.

Birds of a Feather Session

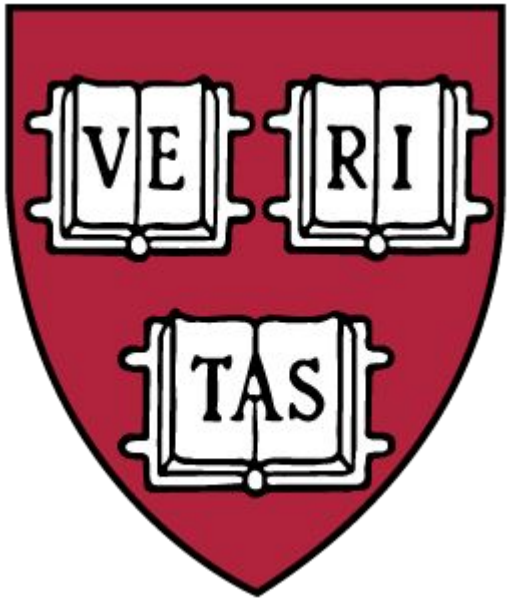
Thursday 12:15 - 13:15

Room 401-402-403-404 (Subject to Change)

SLUG 2020

Next Fall at Harvard University in Boston*

September 15-16th, 2020



*Technically in Cambridge, Massachusetts, USA

Thank you to the University of Utah