LCI workshop
on high performance clustered computing

LCI Advanced Workshop 2025: **Slurm High-Availability & Federation**

Josh Burks
HPC Systems Analyst
Arizona State University Research Computing
josh.burks@asu.edu

This document is a result of work by volunteer LCI instructors and is licensed under CC BY-NC 4.0 (https://creativecommons.org/licenses/by-nc/4.0/

ASU Research Computing

Learning Goals

- Learn what Slurm High-Availability is and isn't, and when it makes sense to use
- Understand how to configure and verify a multi-controller HA setup
- Appreciate the trade-offs that led our site to abandon Slurm HA
- Learn what Slurm Federation is, and isn't, and when it makes sense to use it
- Understand basic federation setup (sacctmgr commands, cluster roles) and federated job submission
- Apply best practices for testing, monitoring, and maintaining both HA and federation setups

HA Architecture

- Slurm has always supported HA. The code for it exists in the 1.0 branch of slurm, with references in the changelog as bar back as SLURM 0.2.12.
- Was significantly revised in 18.08 to support more than one backup controllers.
- Slurm HA is a fallback system - one controller is deemed primary, all other controllers are backup

HA Architecture

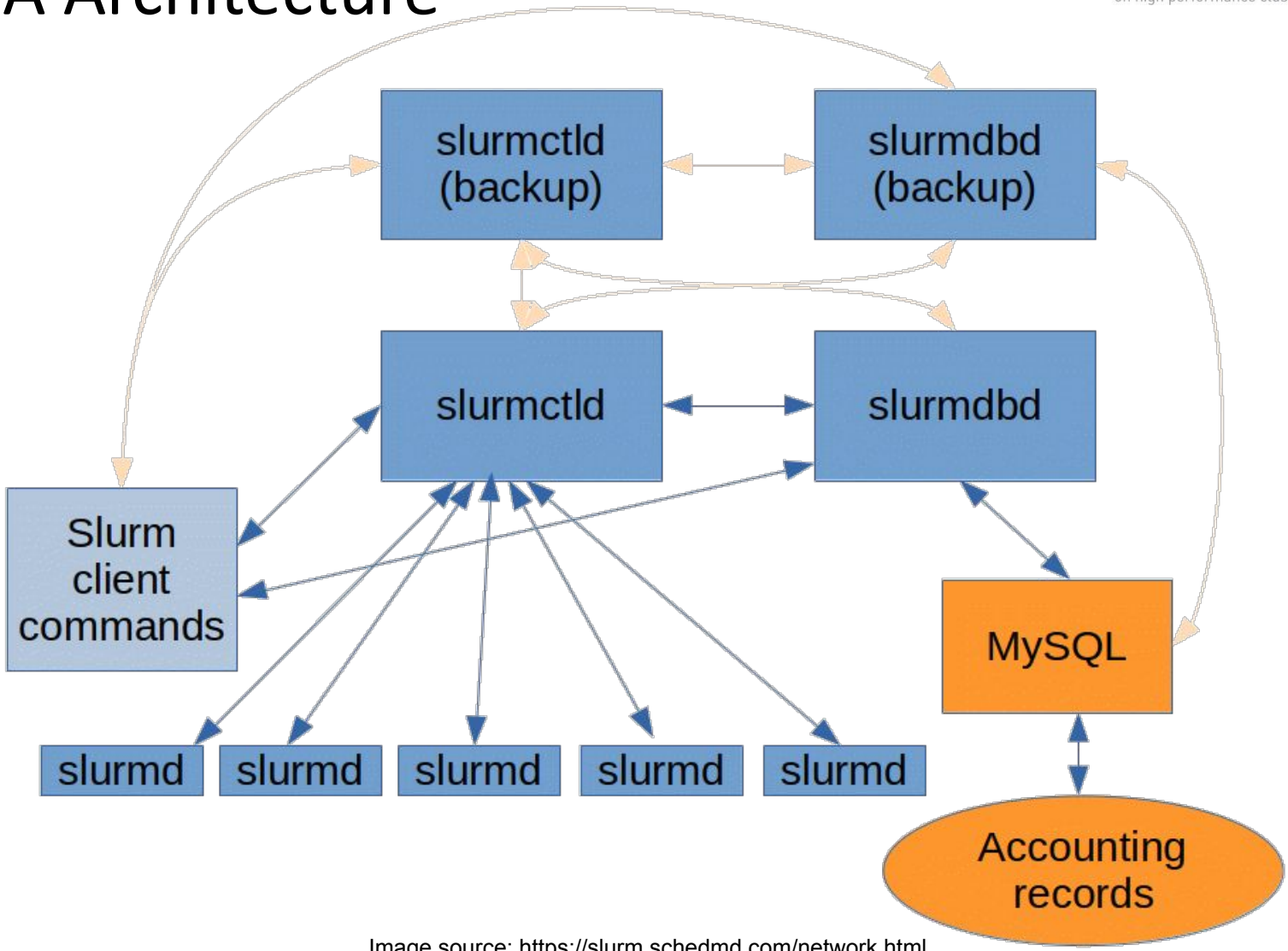


Image source: https://slurm.schedmd.com/network.html

HA Architecture

- A shared filesystem is required on all slurmctld hosts (StateSaveLocation)
- This is where slurms stores its “state” - queued, running and recently completed jobs, node state
- If a controller comes up without access to the state information, queued and running jobs will be cancelled.
- This shared file system should NOT be an NFS mounted filesystem
  - Slurm does constant read/write operations to this location, and NFS is not up to the task

HA Configuration

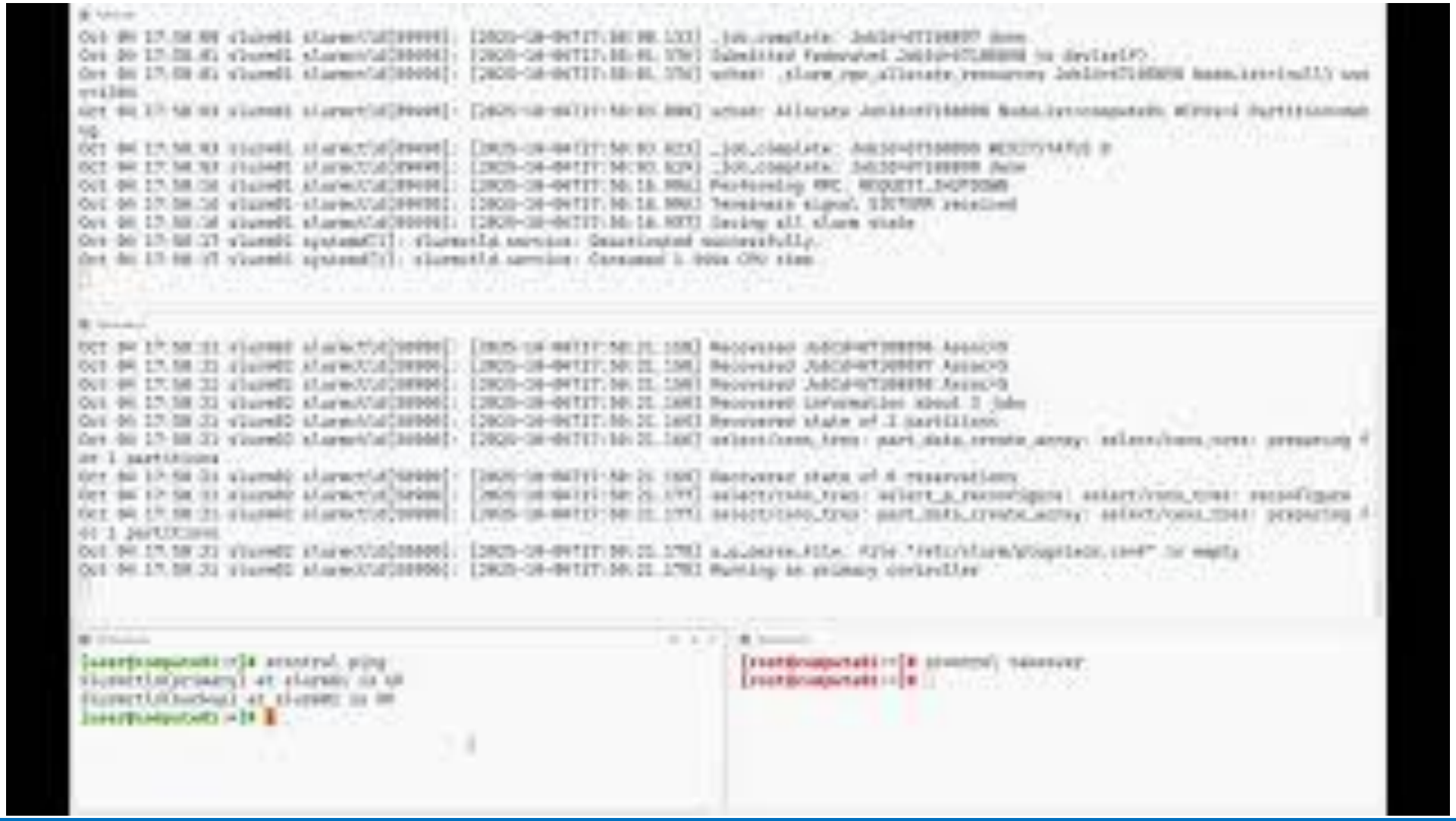
```
#/etc/slurm/slurm.conf
SlurmctldHost=slurm01(10.10.30.1)   # Primary controller

#/etc/slurm/slurm.conf
SlurmctldHost=slurm01(10.10.30.1)   # Primary controller
SlurmctldHost=slurm02(10.10.30.2)   # Secondary controller

# scontrol reconfig
```

HA Configuration

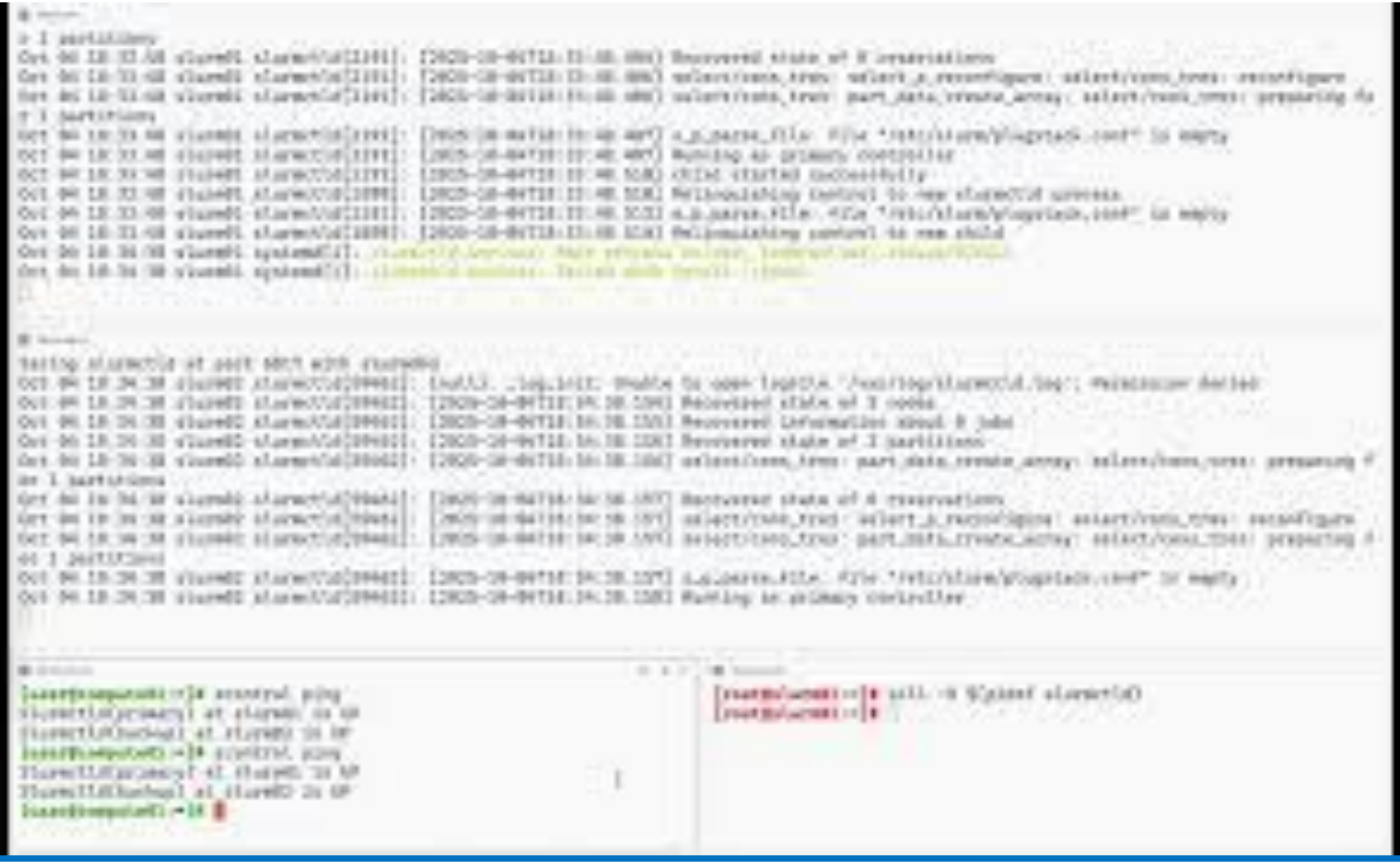
```
#/etc/slurm/slurm.conf
SlurmctldHost=slurm01(10.10.30.1)   # Primary controller
SlurmctldHost=slurm02(10.10.30.2)   # Secondary controller
SlurmctldPrimaryOnProg=/usr/local/bin/slurmctld_takeover.sh
SlurmctldPrimaryOffProg=/usr/local/bin/slurmctld_release.sh
SlurmctldTimeout=60
SlurmctldParameters=no_backup_scheduling
```

```
# scontrol reconfig
```

Demo: HA Failover (Manual)



Demo: HA Failover (Automatic)



HA Best Practices

- Perform failover drills (bring primary down, verify backup takes over)
- Use a small value for `SlurmctldTimeout` to ensure quick failover in the event of an unexpected shutdown
- Maintain tight time synchronization across controllers and nodes
- Use reliable, low-latency shared storage for StateSaveLocation
  - SAN, DRDB, LINSTOR, but not NFS

HA Alternatives

- Slurmctld in a VM
  - Less prone to hardware failures
  - Migratable
- Slurmctld in a Container
  - Use Kubernetes for HA
- Federation
  - Other cluster's slurmctld can take over some job scheduling

What Is Federation?

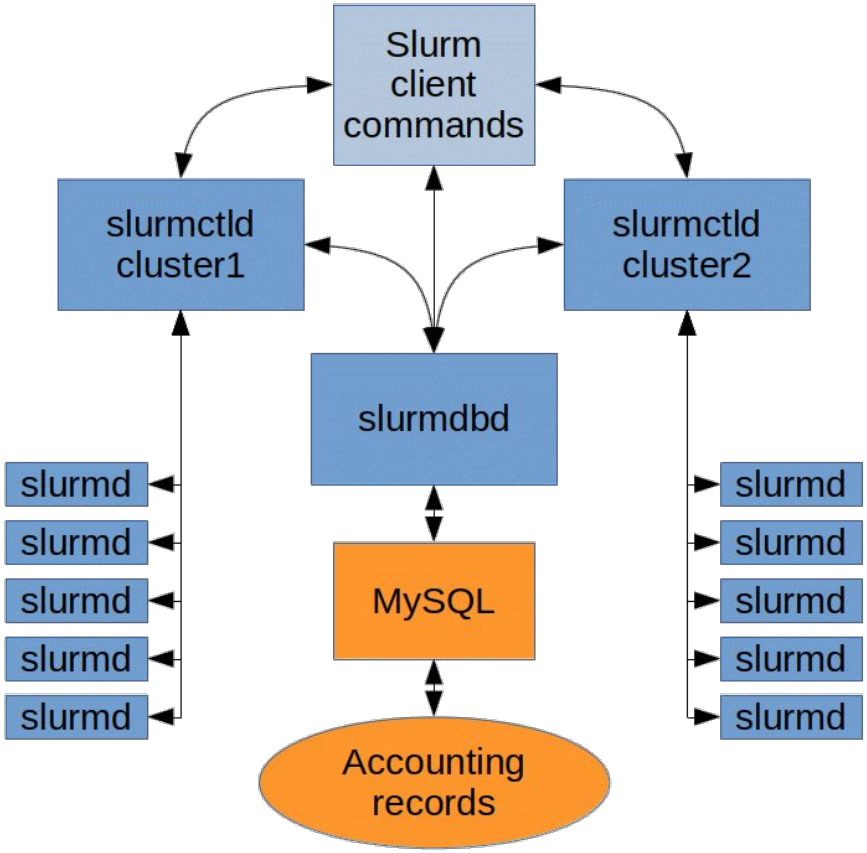
- Federation = loose coordination of multiple Slurm clusters.
- Allows job and job step ID consistency across clusters.
  - Bits 0-25: Local Job ID
  - Bits 26-31: Cluster Origin ID
- Enables job forwarding between clusters when local resources are insufficient.
- Federated clusters retain full autonomy; no shared state or database.
- No global scheduler - each cluster operates independently but shares limited information.

Multi-Cluster vs Federation

- Multi-Cluster
  - Each cluster has its own slurmctld(s) and may use a shared or separate slurmdbds
  - If slurmdbds are separate, they must be linked with with AccountingStorageExternalHost= in slurm.conf
  - Users have separate accounts for each cluster
  - Users who have an account on both clusters can submit jobs to either cluster with the -M flag

- Federated Clusters
  - Each cluster has its own slurmctld(s) but must use **shared** slurmdbd(s)
  - Joined via the shared database
  - Unified accounting structure

Multi-Cluster



Federation

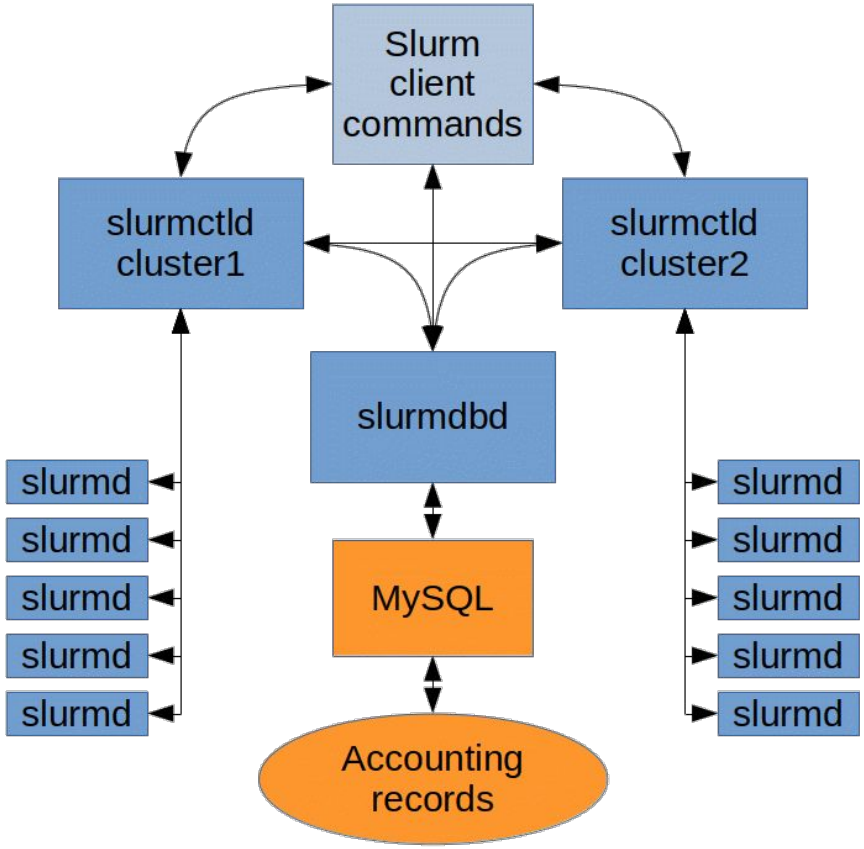


Diagram source: https://slurm.schedmd.com/network.html

Federation Architecture

- Each cluster runs its own SlurmDBD and controller (slurmctld).
- Federation relies on:
  - slurmctld communication
  - job submission tools (e.g., sbatch, srun)
- Uses slurmdbd federation support for job/account tracking.
- Consistent UID/GID & account naming is critical.
- Technically a shared file system is not required, but it is recommended
- Requires bidirectional network visibility between slurmctld instances.

Federation Configuration

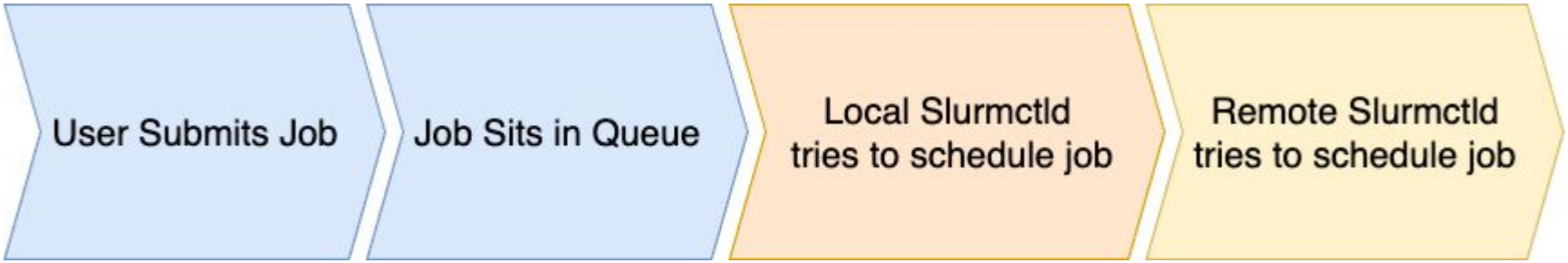
LCI workshop
on high performance clustered computing

- Enable federation with:

```
sacctmgr create federation name=myfed clusters=cluster1,cluster2
```

Job Lifecycle in Federation

- User submits job
- Job sits in queue
- Local slurmctld tries to schedule it
- If resources are unavailable, it considers forwarding (if configured)
- Remote slurmctld accepts and schedules job



Demo: Federated Submission

LCI workshop
on high performance clustered computing

ASU Research Computing

Federation Best Practices & Limits

LCI workshop
on high performance clustered computing

- Best Practices:
  - Ensure consistent UID/GID, accounts, and partitions across clusters.
  - Use identical job submission wrappers or front-ends to avoid surprises.
  - Test forwarding with low-priority jobs before production usage.
  - Monitor with tools like sacct, squeue with --clusters.

- Limitations:
  - No **global** job prioritization or fair-share enforcement.
  - No shared scheduling view across clusters.
  - Forwarded jobs can behave differently based on remote cluster policy.
  - Debugging issues is more complex across cluster boundaries.
  - Job arrays only run on the origin cluster.

Q & A