Slurm Overview

Felip Moll - felip.moll@schedmd.com

SchedMD LLC
www.schedmd.com

Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- Accounting

Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- Accounting

Role of a Resource Manager

- The “glue” for a parallel computer to execute parallel jobs

On a PC:
```
Execute program
"a.out":
a.out
```

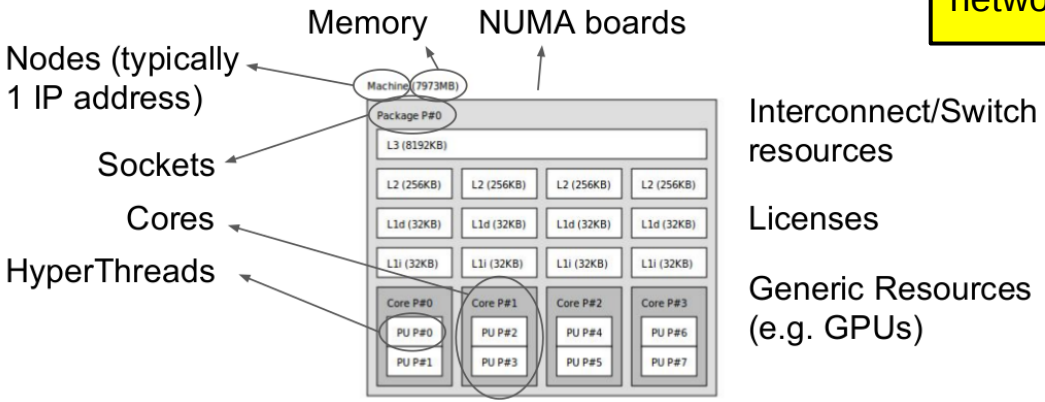
On a cluster:
```
Execute 8 copies of
"a.out":
srun -n8 a.out
```

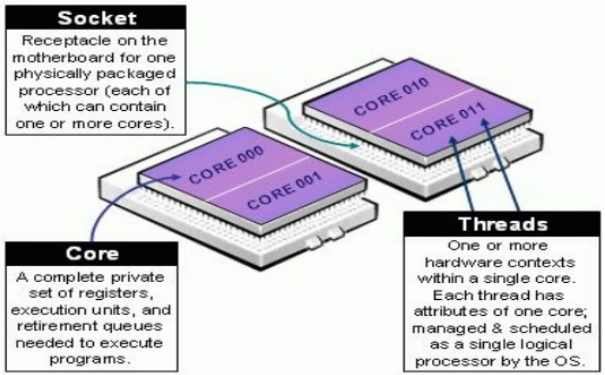
- MPI or UPC would typically be used to manage communications within the parallel program

Role of a Resource Manager

- Allocate resources within a cluster

Requires extensive knowledge about the hardware and system software (e.g. to alter network routing or manage switch window)



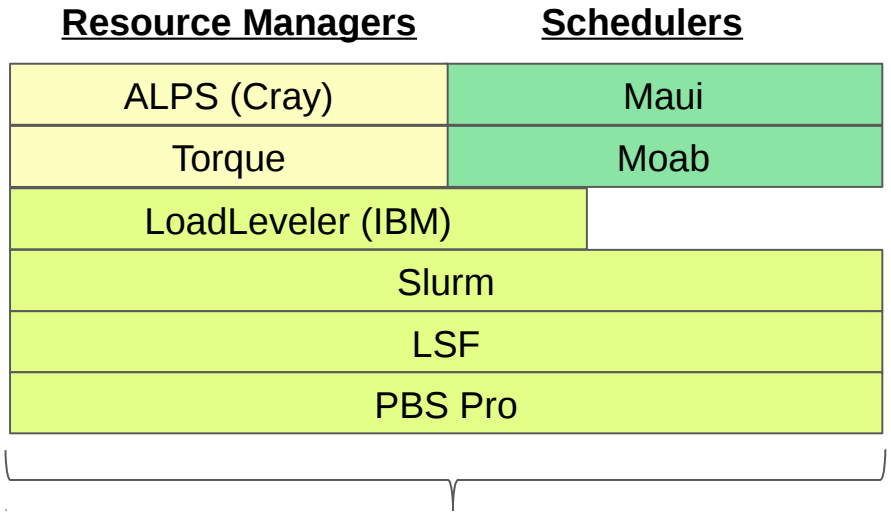


- Launch and otherwise manage jobs

Role of a Job Scheduler

- When there is **more work than resources**, the job scheduler manages queue(s) of work:
  - Supports complex scheduling algorithms
    - Optimized for network topology, fair share scheduling, advanced reservations, preemption, gang scheduling (time-slicing jobs), etc.
  - Supports resource limits, QoS: by queue, user, group, etc.

Examples



Slurm started as a resource manager (the “rm” in “Slurm”) and added scheduling logic later

Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- Accounting

What is Slurm?

- Historically, Slurm was an acronym of:
  - **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Dev. started in 2002 @ Lawrence Livermore National Lab as a resource manager for Linux clusters
- Sophisticated scheduling plugins added in 2008
- About 550,000 lines of C code today
- Supports Linux and limited support for other Unix variants
- Used on many of the world's largest computers
- Active global user community

Who is using Slurm?

www.schedmd.com
SLUG Sept. 25&26th, 2018

Slurm Design Goals

- Highly scalable
  - Managing 3.1 million core Tianhe-2
  - Tested to much larger systems using emulation
- Open source GPLv2, available on Github: https://github.com/SchedMD/
- System administrator friendly
- Secure
- Fault-tolerant (no single point of failure)
- Portable - Targeting POSIX2008.1 and C99

Slurm Portability

- *Autoconf* configuration engine adapts to environment
- Provides scheduling framework with general-purpose plugin mechanism. Extensively customize using a building-block approach.
- Various system-specific plugins available (e.g. *`select/cray`*)
- Huge range of use cases:
  - Sophisticated workload management at HPC sites
  - Scalable HTC environments (**14k jobs/minute sustained)**

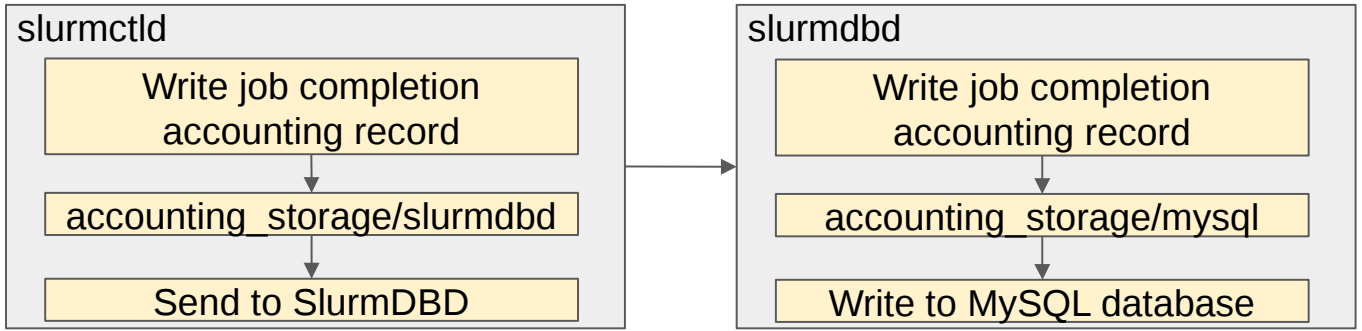
Plugins

- Dynamically linked objects loaded at run time based upon configuration file and/or user options
- 110+ plugins of 30 different varieties currently available
  - Network topology: 3D-torus, tree, etc
  - MPI: OpenMPI, PMI2, PMIx
  - Process tracking: cgroup, linuxproc, pgid, ipmi, etc.

Slurm Kernel (65% of code)				
Authentication Plugin	MPI Plugin	Job Submit Plugin	Topology Plugin	Accounting Storage Plugin
MUNGE	PMI2	Lua	Tree	MySQL

Plugin Design

- Plugins typically loaded when the daemon or command starts and persist indefinitely
- Provide a level of indirection to a configurable underlying function



Slurm Entities

- **Jobs:** Resource allocation requests
- **Job steps:** Set of (typically parallel) tasks
  - Typically an MPI, UPC and/or multi-threaded application program
  - Allocated resources from the job's allocation
  - A job can contain multiple job steps which can execute sequentially or concurrently
  - Use cases with thousands of job steps are common
  - Lighter weight than jobs
- **Partitions:** Job queues with limits and access controls
- **QoS:** Limits and polices

Slurm Entities Example

- Users submit jobs to partitions (queue):

Priority ordered queue of jobs

Partition *debug*
Job 1 Job 2 Job 3

Slurm Entities Example

- Jobs are allocated resources:

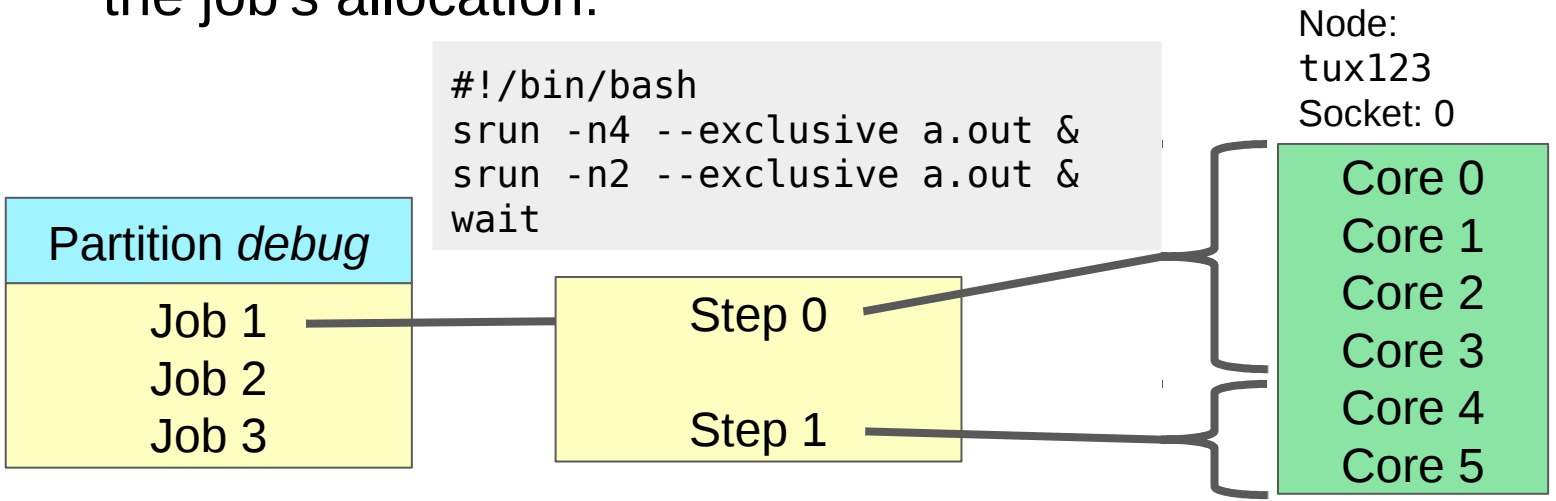
```
#!/bin/bash
#SBATCH -n6
..
```

Node: `tux123`
Socket: 0

Slurm Entities Example

- Jobs spawn steps, which are allocated resources from within the job's allocation:

```
#!/bin/bash
srun -n4 --exclusive a.out &
srun -n2 --exclusive a.out &
wait
```



Node State Information

- Baseboards, Sockets, Cores, Threads
- CPUs (Core or thread count depending upon configuration)
- Memory size
- Generic resources (with names and counts)
- Features (arbitrary string, e.g. OS version or CPU type)
- State (e.g. drain, down, etc.)
  - Reason, time and user ID
    - e.g. “`Bad PDU [operator@12:40:10T12/20/2013]`”

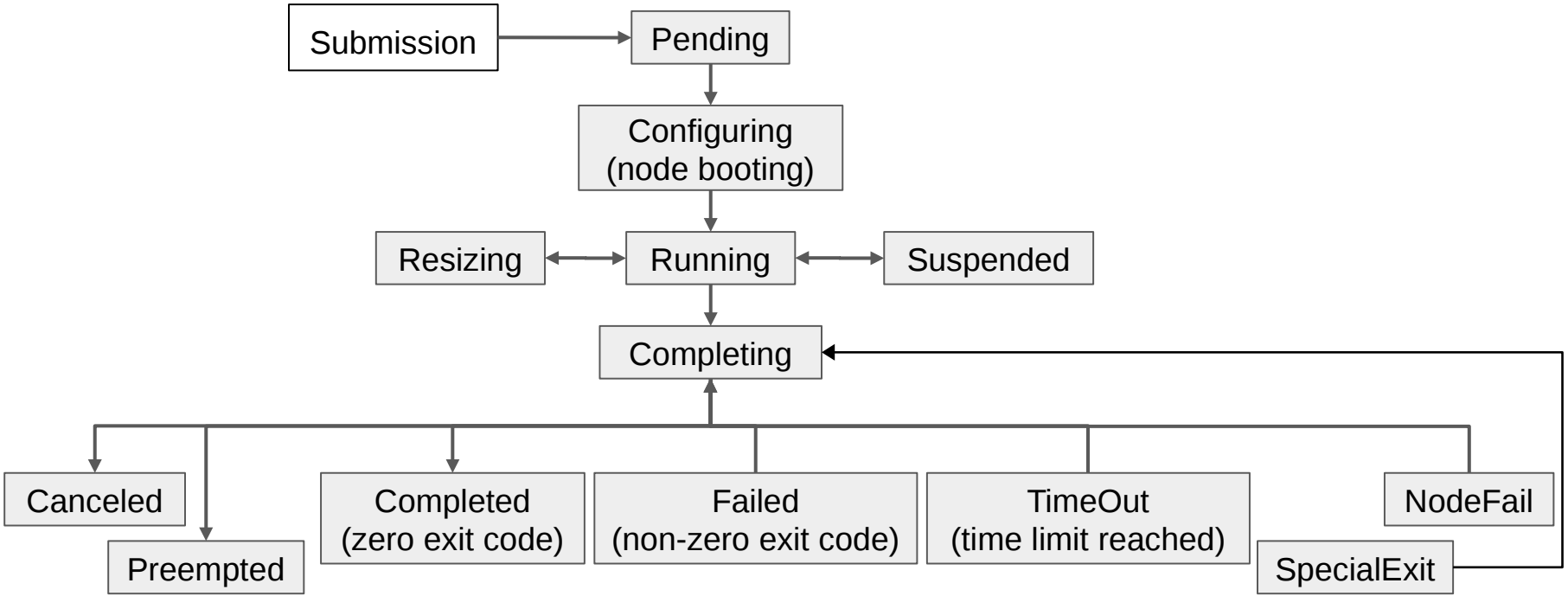
Queue/Partition State Information

- Associated with specific set of nodes
  - Nodes can be in more than one partition
- Job size and time limits (e.g. small size and time limits for some partition and larger limits for others)
- Access control list (by account, Quality Of Service, or Linux group)
- Preemption rules
- State information (e.g. up, down, drain, etc.)
- Over-subscription and gang scheduling rules

Job State Information

- ID (a number)
- Name
- Time limit (minimum and/or maximum)
- Size specification (minimum and/or maximum; nodes, CPUs, sockets, cores, and/or threads)
- Specific node names to include or exclude in allocation
- Node features required in allocation
- Dependency
- Account name
- Quality Of Service (QOS)
- State (Pending, Running, Suspended, Canceled, Failed, etc.)

Job States



Step State Information

- ID (a number): `<jobid>.<stepid>`
- Name
- Time limit (maximum)
- Size specification (minimum and/or maximum; nodes, CPUs, sockets, cores, and/or threads)
- Specific node names to include or exclude in allocation
- Node features required in allocation

Summary

- Job is submitted to a Slurm queue/partition
- Job is allocated resources (cores, memory, etc.)
- Job steps execute applications using the job's resources
- Job and associated resources are monitored:
  - By plugins: JobAcctGather, Cgroup, NHC, etc.

Daemons

- **slurmctld** – Central controller (typically one per cluster)
  - Monitors state of resources
  - Manages job queues
  - Allocates resources

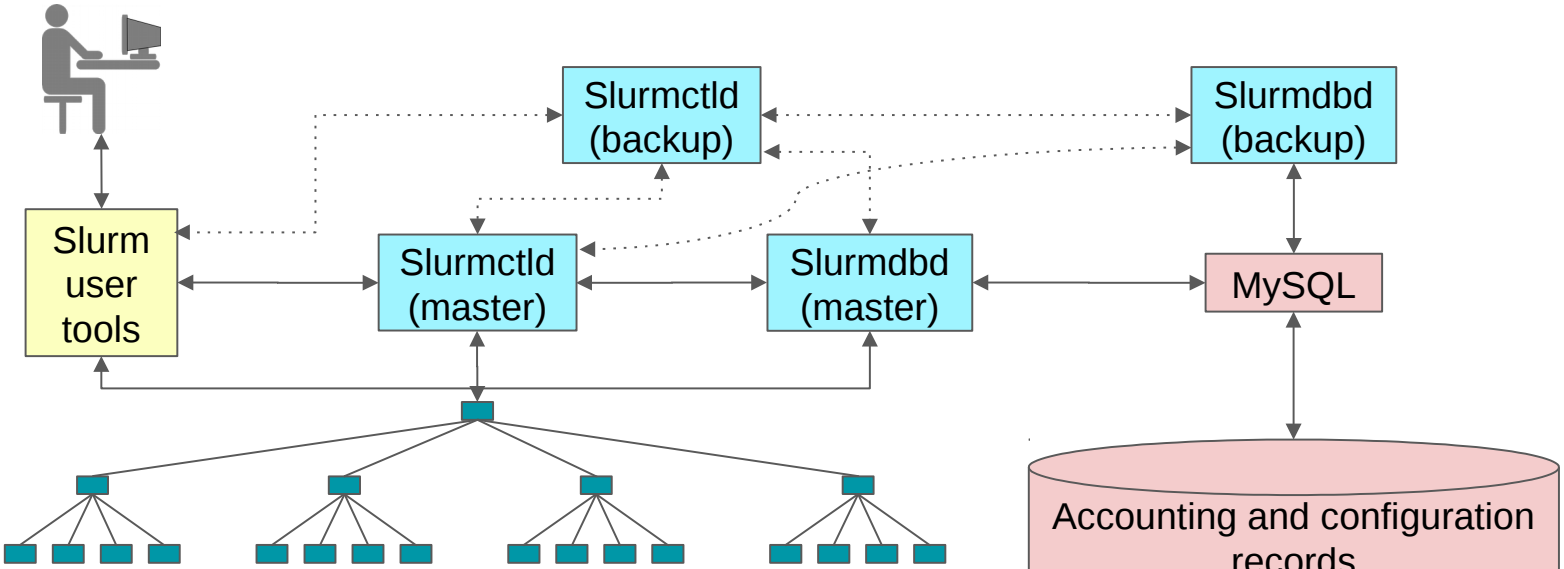
- **slurmdbd** – Database daemon (typically one per enterprise)
  - Collects accounting information
  - Manages accounting configuration (e.g. limits, fair-share, etc.)
    - Pushes to controller(s)

Daemons

- **slurmd** – Compute node daemon
  - Typically one per compute node
  - Launches and manages slurmstepd (see below)
  - Small and very light-weight
  - Quiescent after launch except for optional accounting
  - Supports hierarchical communications with configurable fanout

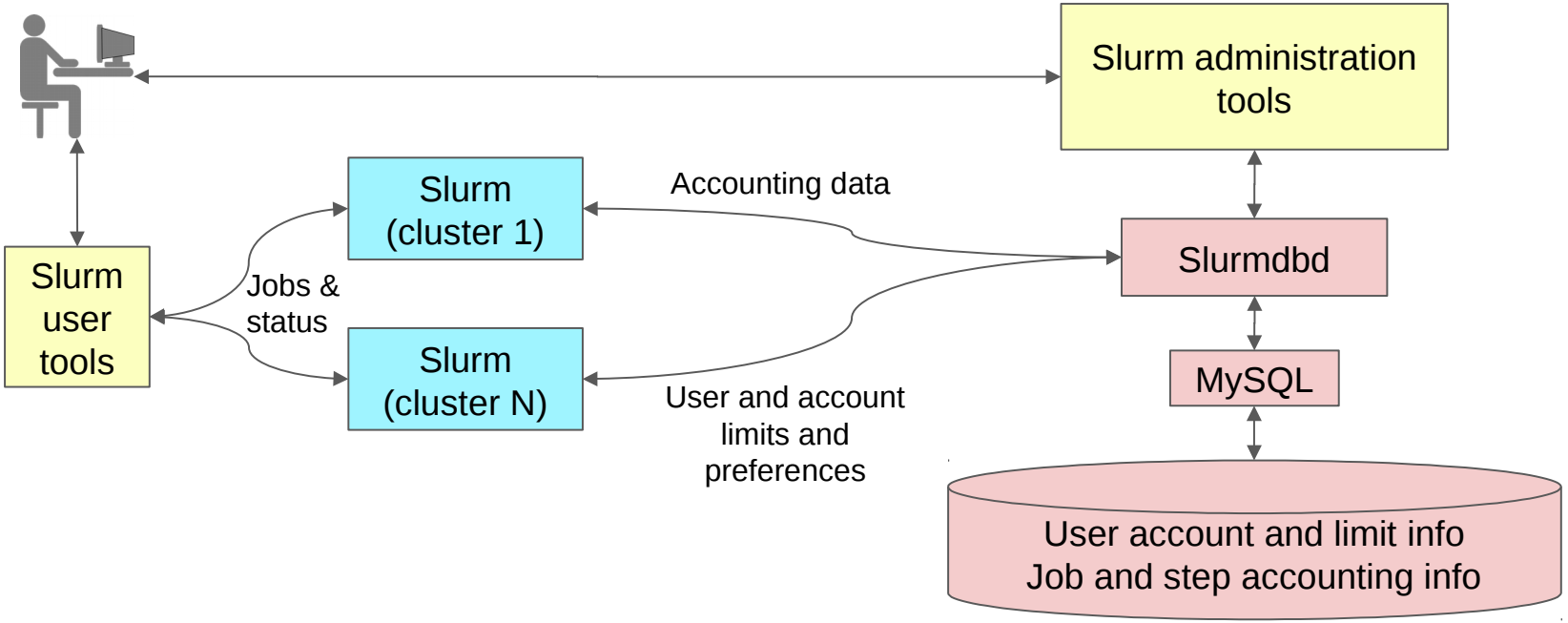
- **slurmstepd** – Job step shepherd
  - Launched for batch job and each job step
  - Launches user application tasks
  - Manages accounting, application I/O, signals, etc.

Cluster Architecture - Typical Linux Cluster



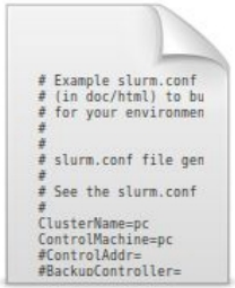
Slurmd daemons on compute nodes
(Note hierarchical communications with configurable fanout)

Typical Enterprise Architecture



Configuration

slurm.conf



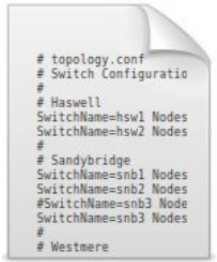
- **General conf**
- **Plugin activation**
- **Scheduler parameters**
- **Node definition**
- **Partition definition**

slurmdbd.conf

- **Slurmdbd params.**
- **Archive/Purge**
- **Storage options**

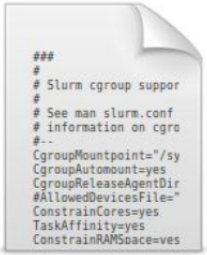
Configuration

topology.conf



gres.conf

cgroup.conf



- **Other files configure specific plugin options**
  - **burst_buffer.conf, acct_gather.conf, knl.conf...**

Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- Accounting

Commands: General Information

- Man pages available for all commands, daemons and configuration files
- `--help` option prints brief description of all options
- `--usage` option prints a list of the options
- Commands can be run on any node in the cluster
- Any failure results in a non-zero exit code
- APIs make new tool development easy
  - Man pages available for all APIs

Commands: General Information

- Almost all options have two formats
  - A single letter option (e.g. "`-p debug`" for partition "*debug*")
  - A verbose option (e.g. "`--partition=debug`")
- Almost all commands support verbose logging with "`-v`" option
  - Use more v's for more verbosity, e.g. `-vvvv`
- Many environment variables can be used to establish site-specific and/or user-specific defaults
  - For example "`SQUEUE_STATES=all`" for the squeue command to display jobs in any state, including `COMPLETED` or `CANCELLED`

Slurm Commands: Job/step Allocation

- **`sbatch`** – Submit script for later execution (batch mode)
- **`salloc`** – Create job allocation and start a shell to use it (interactive mode)
- **`srun`** – Create a job allocation (if needed) and launch a job step (typically an MPI job)
- **`sattach`** – Connect stdin/out/err for an existing job step

Job allocation options

- The job allocation commands (*salloc*, *sbatch*, and *srun*) accept almost identical options

- There are a handful of options that only apply to a subset of these commands (e.g. batch job requeue options, cpu binding)

sbatch Example - Options in Batch Script

Submit a batch job

```
> sbatch my_work2.bash
Submitted batch job 44005
> cat my_work2.bash
#!/bin/bash
#SLURM --ntasks=128
#SLURM --cpus-per-task=2
#SLURM --mem-per-cpu=20
#SLURM --time=60
…
```

sbatch Example - Job Dependencies

Submit sequence of three batch jobs

```
> sbatch --ntasks=1 --time=10 pre_process.bash
Submitted batch job 45001

> sbatch --ntasks=128 --time=60 --dependency=afterok:45001 do_work.bash
Submitted batch job 45002

> sbatch --ntasks=1 --time=30 --dependency=afterok:45002 post_process.bash
Submitted batch job 45003

> sbatch --begin=17:00:00 -n1 -t10 night.bash
Submitted batch job 45004
```

Options used

`--ntasks or -n`	Number of tasks and by default the number of CPUs
`--time or -t`	Wall time limit (`minutes` in our example)
`--dependency or -d`	Job dependency
`--begin`	Delay job initiation at least until this date and time

srun Example - Multiple Serial Job Steps

```
> salloc -n128 bash
Granted job allocation 10090
> srun step.1
> srun step.2
> srun step.3
> exit
```

Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- Accounting

Job Management Commands

- **`scancel`** – Signal / terminate job and steps
- **`squeue`** – Report job and job step status

Why is my job not running?

- As soon as some reason is found why a job cannot be started, that is recorded in the job's "`reason`" field and the scheduler moves on to the next job

Some common reasons why jobs are pending:

Priority	Resources being reserved for higher priority job
Resources	Required resources are in use
Dependency	Job dependencies not yet satisfied
Reservation	Waiting for advanced reservation
AssociationJobLimit	User or account job limit reached
AssociationResourceLimit	User or account resource limit reached
AssociationTimeLimit	User or account time limit reached
QOSJobLimit	Quality Of Service (QOS) job limit reached
QOSResourceLimit	Quality Of Service (QOS) resource limit reached
QOSTimeLimit	Quality Of Service (QOS) time limit reached

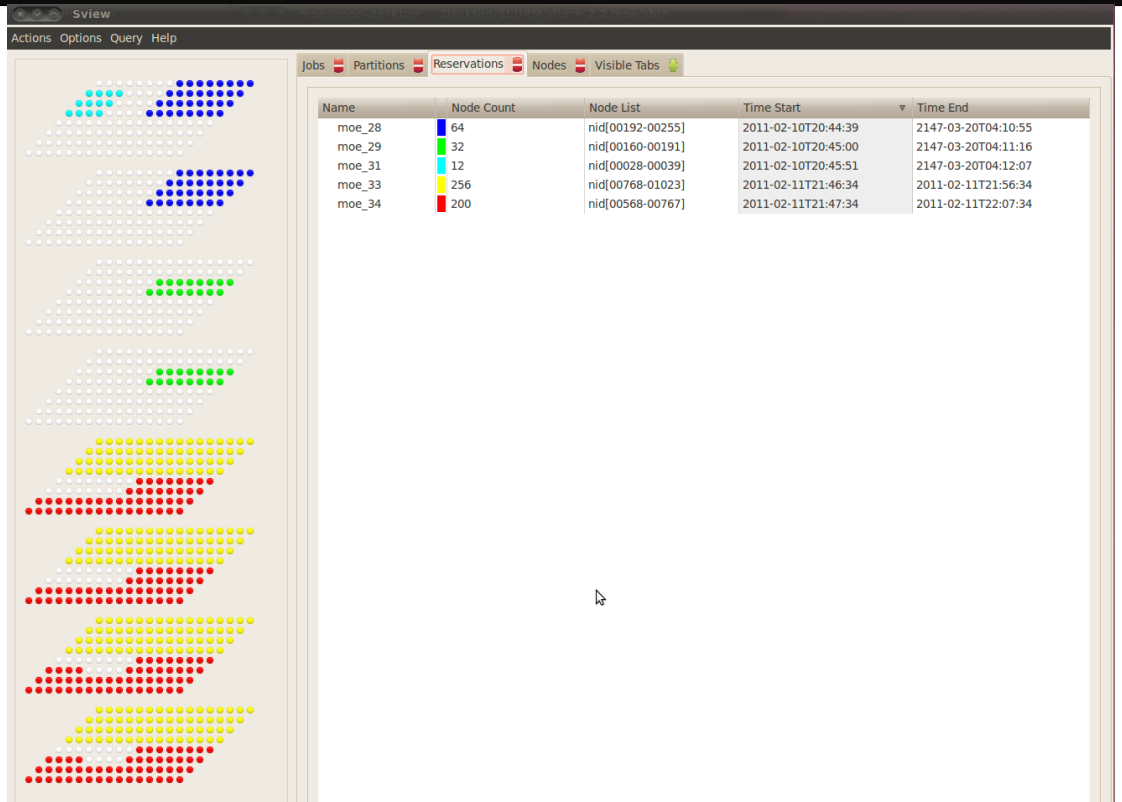
Slurm System Information

- **`sinfo`** – Report system status (nodes, queues, etc.)
- **`sview`** – Report and/or update system, job, step, partition or reservation status with topology (GTK-based GUI)
- **`smap`** – Report system, job or step status with topology (curses-based GUI), less functionality than sview
- **`scontrol`** – Administrator tool to view and/or update system, job, step, partition or reservation status

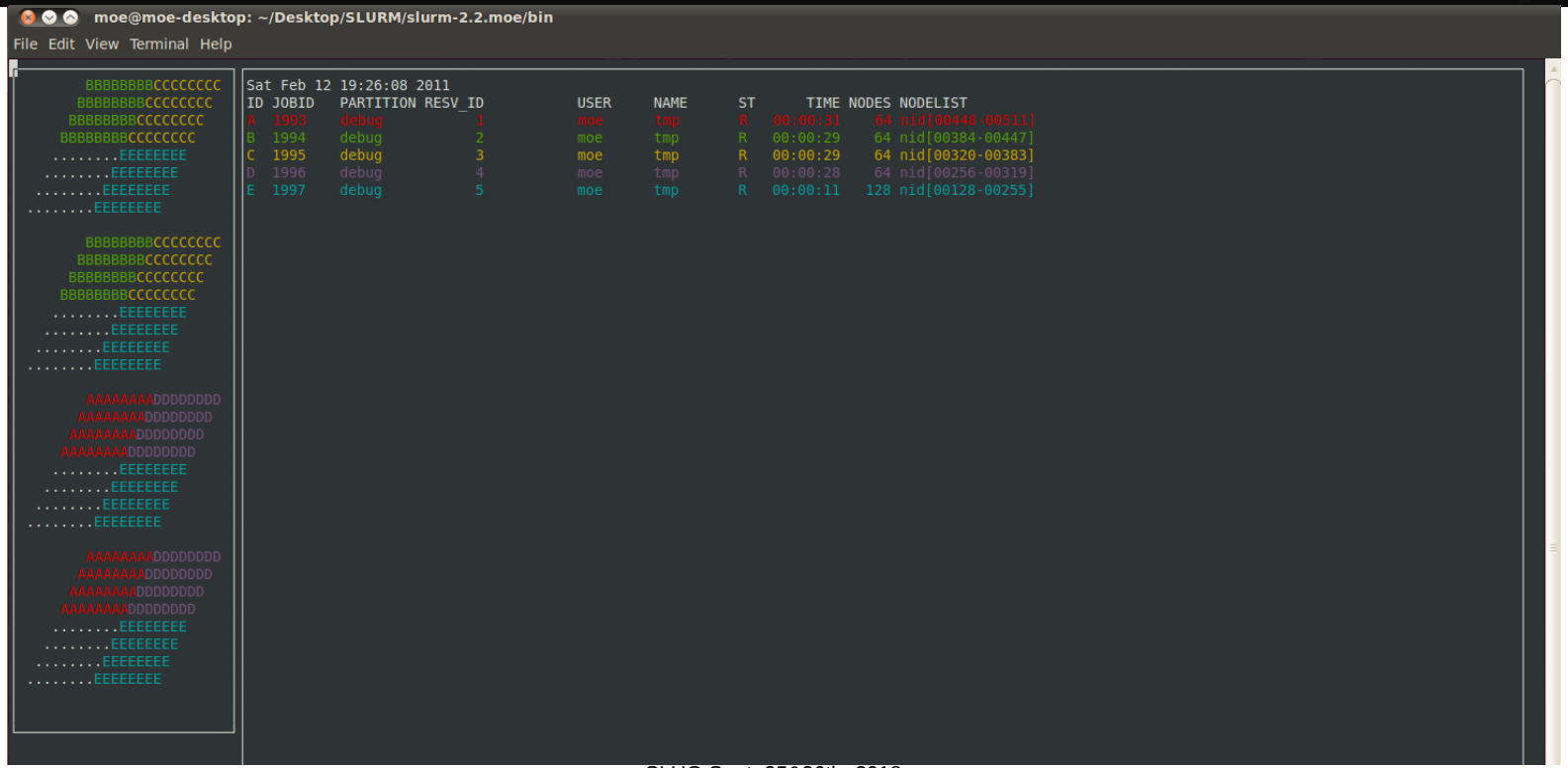
sview Functionality

- Get details about specific jobs, steps, nodes, etc.
- Can be used for job control functions
  - Cancel job
  - Hold/release job, etc.
  - Support for job submission with relatively few options
- Can be used for administrative functions
  - Change configuration, etc.

sview on Cray (3D torus)



smap on Cray (3D torus)



scontrol Command

- Designed primarily for system administrator use
- Shows all available fields, but no filtering, sorting or formatting options
- Many fields can be modified when pending or running

```
> scontrol show job 123
JobId=123 Name=tmp
UserId=jette(1000) GroupId=jette(1000)
Priority=4294901737 Nice=0 Account=(null) QOS=(null)
JobState=RUNNING Reason=None Dependency=(null)
RunTime=00:00:11 TimeLimit=00:30:00 TimeMin=N/A

> scontrol update JobId=123 TimeLimit=20
```

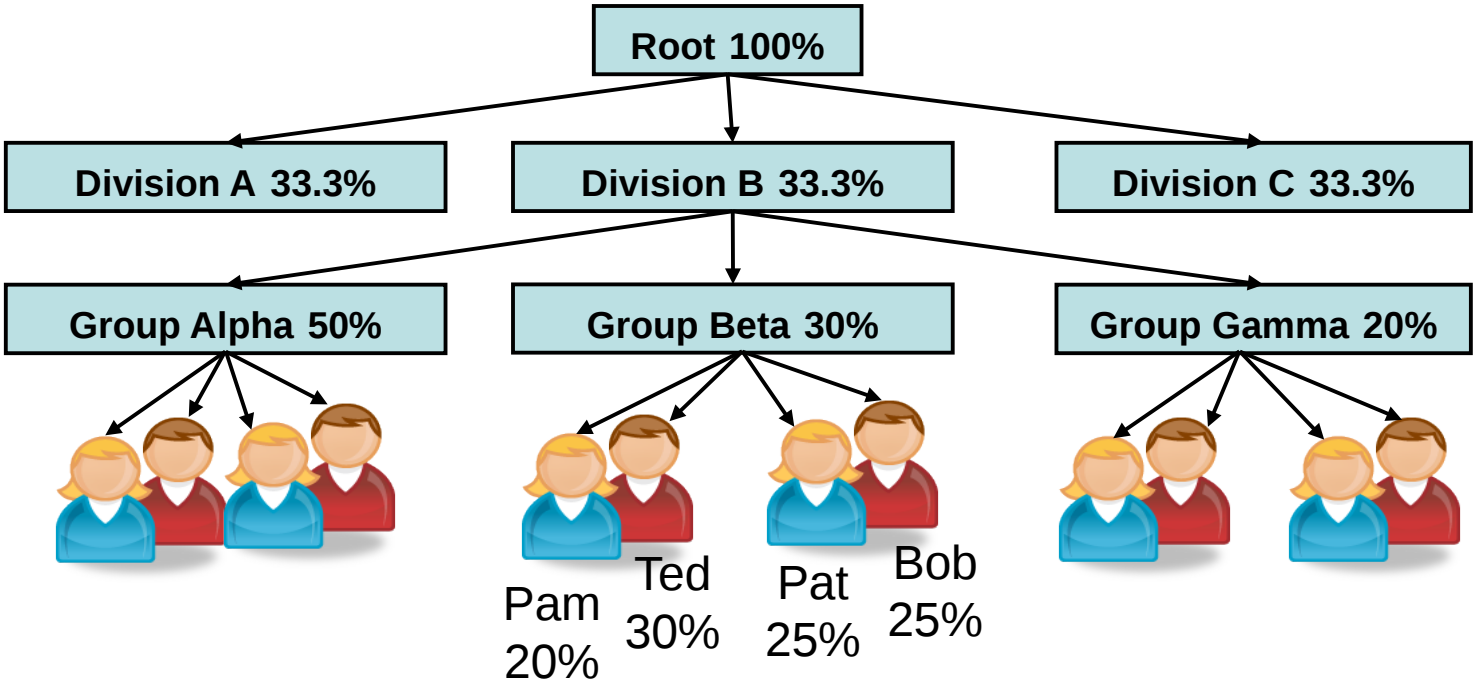
Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- Accounting

Database Use

- Accounting information written to a database plus
  - Information pushed out live to scheduler daemons
  - Quality of Service (QOS) definitions
  - Fair-share resource allocations
  - Many limits (max job count, max job size, etc)
  - Based upon hierarchical accounts
    - Limits by user AND by accounts

Hierarchical Accounts Example



Hierarchical Accounts

- All users are not created equal
  - Different shares of resources
  - Different measures of being over- or under-served
  - Different limits
- There are many limits available
  - Per job limits (e.g. `MaxNodes`)
  - Aggregate limits by user, accounts, or QOS (e.g. `MaxJobs`)
  - A single user may have different shares and limits in different accounts, QOS, or partitions

Accounting Commands

- **`sshare`** – Reports account hierarchy
- **`sprio`** – Report components of job priority
- **`sacctmgr`** – Get and set account data and limits
- **`sstat`** – Report accounting information of running job by individual job and job step
- **`sacct`** – Report accounting information by individual job and job step
- **`sreport`** – Report resources usage by cluster, partition, user, account, etc.

Summary

- You should now have a basic overview of Slurm concepts
- Documentation available at https://slurm.schedmd.com
- Questions?