Slurm Overview

Moe Jette - SchedMD
SC16

Outline

- Roles of resource manager and job scheduler
- Slurm description and design goals
- Slurm architecture and plugins
- Slurm configuration files and commands
- Accounting

Outline

- Roles of resource manager and job scheduler
- Slurm description and design goals
- Slurm architecture and plugins
- Slurm config files and commands
- Accounting

Role of a Resource Manager

- The “glue” for a parallel computer to execute parallel jobs
- It should make a parallel computer as almost easy to use as a PC

On a PC.
Execute program “a.out”

```
a.out
```

On a cluster.
Execute 8 copies of “a.out”

```
srun -n8 a.out
```

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

Roles of a Resource Manager

- Allocate resources within a cluster

Memory

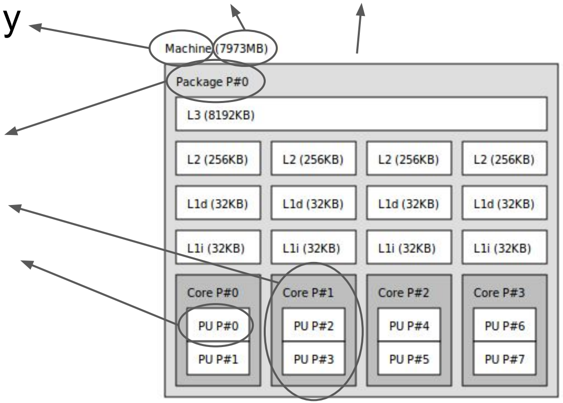
NUMA boards

Nodes (typically
1 IP address)

Sockets

Cores

HyperThreads



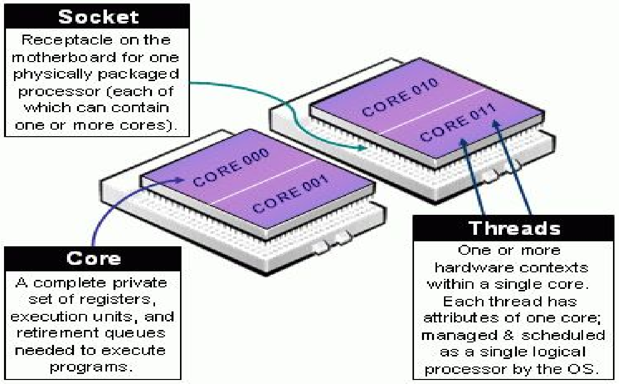
Interconnect/Switch resources

Licenses

Generic Resources (e.g. GPUs)

- Launch and otherwise manage jobs

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



Socket

Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).

Core

A complete private set of registers, execution units, and retirement queues needed to execute programs.

Threads

One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

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), backfill scheduling, etc.
    - Job can be prioritized using highly configurable parameters such as job age, job partition, job size, job QOS, etc.
  - Supports resource limits (by queue, user, group, etc.)

Examples

<table>
<tr><th><u>Resource Managers</u></th><th><u>Schedulers</u></th></tr>
<tr><td>ALPS (Cray)</td><td>Maui</td></tr>
<tr><td>Torque</td><td>Moab</td></tr>
<tr><td>LoadLeveler (IBM)</td><td></td></tr>
<tr><td colspan="2">Slurm</td></tr>
<tr><td colspan="2">LSF</td></tr>
<tr><td colspan="2">PBS Pro</td></tr>
</table>

Many span both roles

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

Outline

- Roles of resource manager and job scheduler
- Slurm description and design goals
- Slurm architecture and plugins
- Slurm config files and commands
- Accounting

What is Slurm?

- Historically Slurm was an acronym standing for
  - **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Development started in 2002 at Lawrence Livermore National Laboratory as a resource manager for Linux clusters
- Sophisticated scheduling plugins added in 2008
- About 500,000 lines of C code today (plus test suite and doc)
- Used on many of the world's largest computers
- Active global development community

Slurm Design Goals

- Highly scalable (managing 3.1 million core Tianhe-2, tested to much larger systems using emulation)
- Open source (GPL version 2, available on Github)
- System administrator friendly
- Secure
- Fault-tolerant (no single point of failure)
- Portable

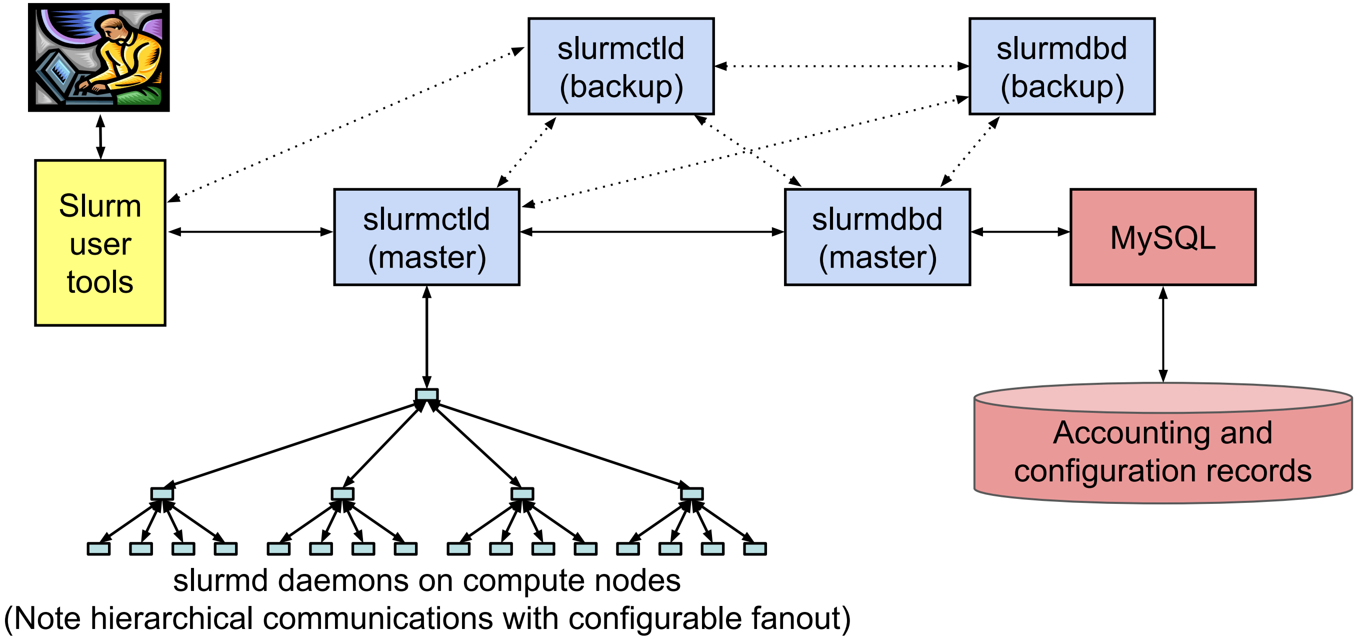
Slurm Portability

- *Autoconf* configuration engine adapts to environment
- Provides scheduling framework with general-purpose plugin mechanism. System administrator can extensively customize installation using a building- block approach
- Various system-specific plugins available and more under development (e.g. *select/bluegene*, *select/cray*)
- Huge range of use cases:
  - Intel's “cluster on a chip”: Simple resource manager
  - Sophisticated workload management at HPC sites

Outline

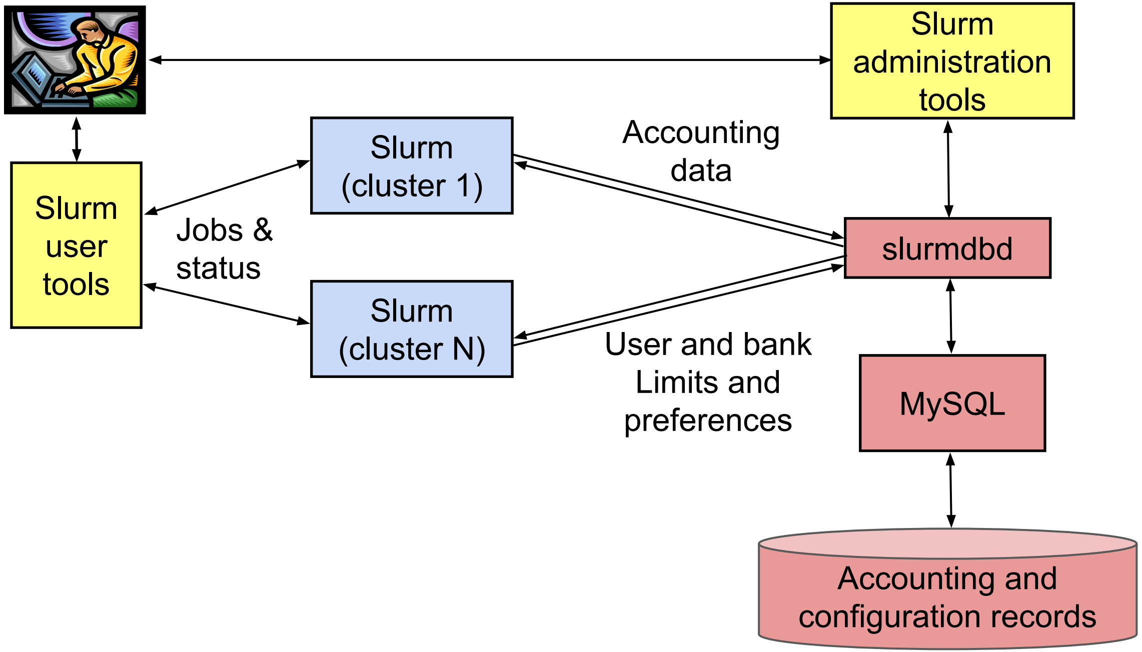
- Roles of resource manager and job scheduler
- Slurm description and design goals
- Slurm architecture and plugins
- Slurm config files and commands
- Accounting

Cluster Architecture



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

Typical Enterprise Architecture


Slurm administration tools
Slurm user tools
Slurm (cluster 1)
Jobs & status
Slurm (cluster N)
Accounting data
slurmdbd
User and bank Limits and preferences
MySQL
Accounting and configuration records

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
  - Uploads configuration information (limits, fair-share, etc.) to slurmctld

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 application I/O, signals, etc.

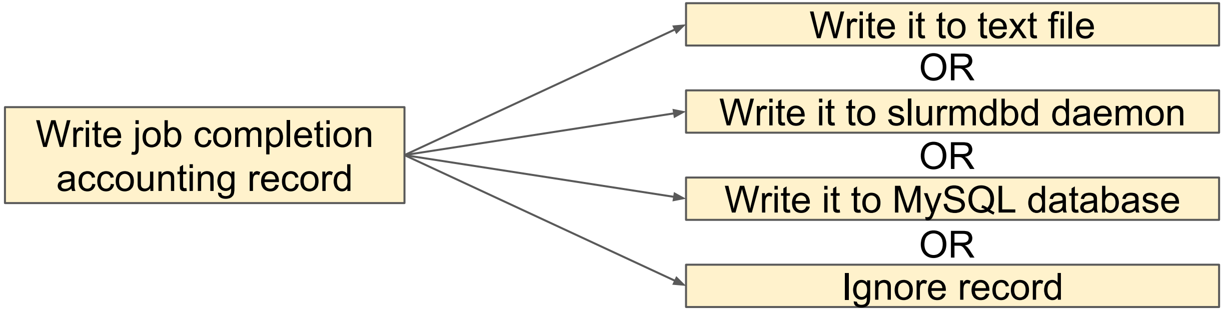
Plugins

- Dynamically linked objects loaded at run time based upon configuration file and/or user options
- 100+ plugins of 26 different varieties currently available
  - Network topology: 3D torus, tree, etc
  - MPI: OpenMPI, MPICH1, MVAPICH, MPICH2, etc
  - External sensors: Temperature, power consumption, etc.

Slurm Kernel (65% of code)				
Authentication Plugin	MPI Plugin	Checkpoint Plugin	Topology Plugin	Accounting Storage Plugin
Munge	pmi2	BLCR	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



Write job completion accounting record

Write it to text file

OR

Write it to slurmdbd daemon

OR

Write it to MySQL database

OR

Ignore record

Plugin Development

- APIs are all documented for custom development (e.g. GreenSpot for optimized use of green energy sources)
- Most plugins have several examples available
- Some plugins have a LUA script interface

Job Submit Plugin

- Call for each job submission or modification
- Can be used to set default values or enforce limits using functionality outside of Slurm proper

Two functions need to be supplied:

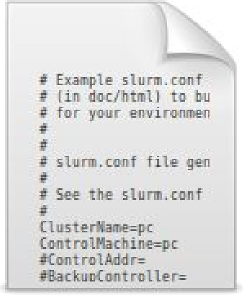
```
int job_submit(struct job_descriptor *job_desc, uint32_t submit_uid);
int job_modify(struct job_descriptor *job_desc, struct job_record *job_ptr);
```

Outline

- Roles of resource manager and job scheduler
- Slurm description and design goals
- Slurm architecture and plugins
- Slurm config files and commands
- Accounting

Slurm Configuration

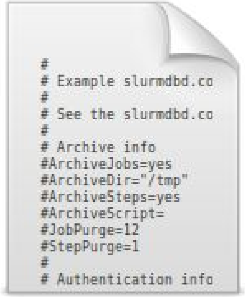
slurm.conf



```
# Example slurm.conf
# (in doc/html) to bu
# for your environmen
#
#
# slurm.conf file gen
#
# See the slurm.conf
#
ClusterName=pc
ControlMachine=pc
#ControlAddr=
#BackupController=
```

- General conf
- Plugin activation
- Sched params
- Node definition
- Partition conf

slurmdbd.conf

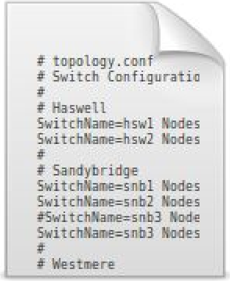


```
#
# Example slurmdbd.co
#
# See the slurmdbd.co
#
# Archive info
#ArchiveJobs=yes
#ArchiveDir="/tmp"
#ArchiveSteps=yes
#ArchiveScript=
#JobPurge=12
#StepPurge=1
#
# Authentication info
```

- Describes slurmdbd
- Archive/Purge parameters
- Storage options

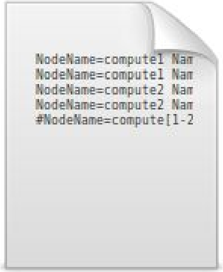
Slurm Configuration

topology.conf



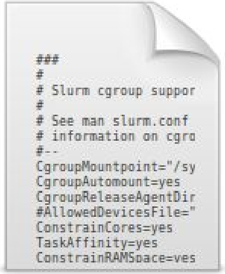
```
# topology.conf
# Switch Configuratio
#
# Haswell
SwitchName=hsw1 Nodes
SwitchName=hsw2 Nodes
#
# Sandybridge
SwitchName=snb1 Nodes
SwitchName=snb2 Nodes
#SwitchName=snb3 Node
SwitchName=snb3 Nodes
#
# Westmere
```

gres.conf



```
NodeName=compute1 Nam
NodeName=compute1 Nam
NodeName=compute2 Nam
NodeName=compute2 Nam
#NodeName=compute[1-2
```

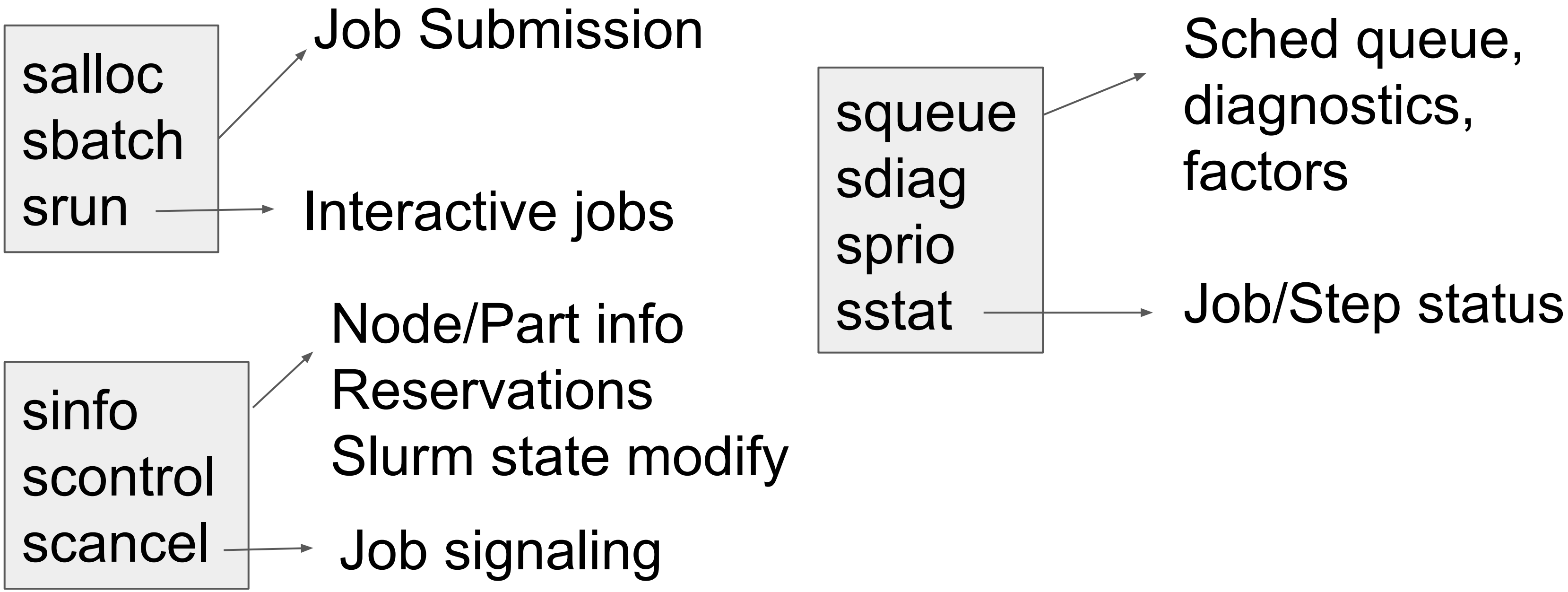
cgroup.conf



```
###
#
# Slurm cgroup suppor
#
# See man slurm.conf
# information on cgro
#--
CgroupMountpoint="/sy
CgroupAutomount=yes
CgroupReleaseAgentDir
#AllowedDevicesFile="
ConstrainCores=yes
TaskAffinity=yes
ConstrainRAMSpace=yes
```

- Others: burst_buffer.conf, acct_gather.conf, knl.conf, etc.

Commands Overview

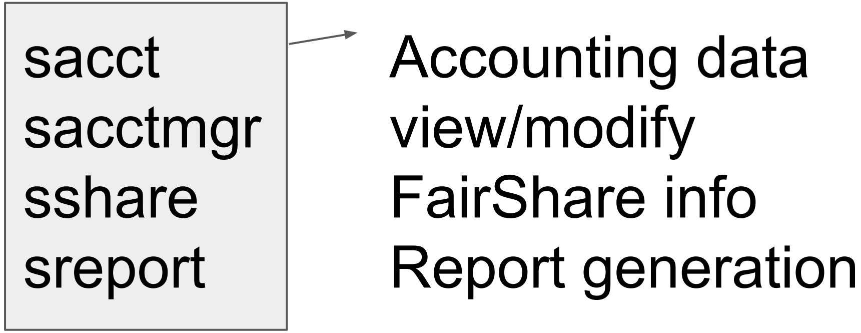


salloc
sbatch → Job Submission
srun → Interactive jobs

sinfo
scontrol → Node/Part info
Reservations
Slurm state modify
scancel → Job signaling

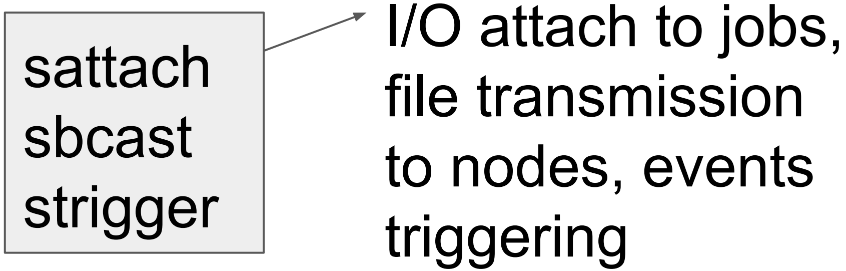
squeue → Sched queue, diagnostics, factors
sdiag
sprio
sstat → Job/Step status

Commands Overview



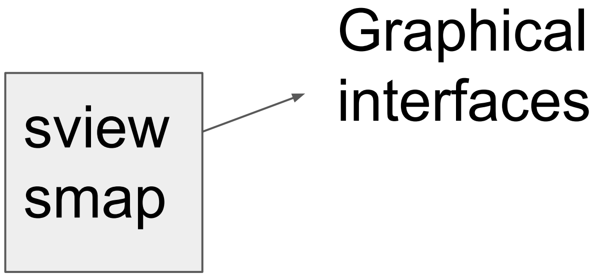
sacct
sacctmgr
sshare
sreport

Accounting data
view/modify
FairShare info
Report generation



sattach
sbcast
strigger

I/O attach to jobs, file transmission to nodes, events triggering



sview
smap

Graphical interfaces

- --help, --usage
- man pages
- APIs make new tools development easier

Outline

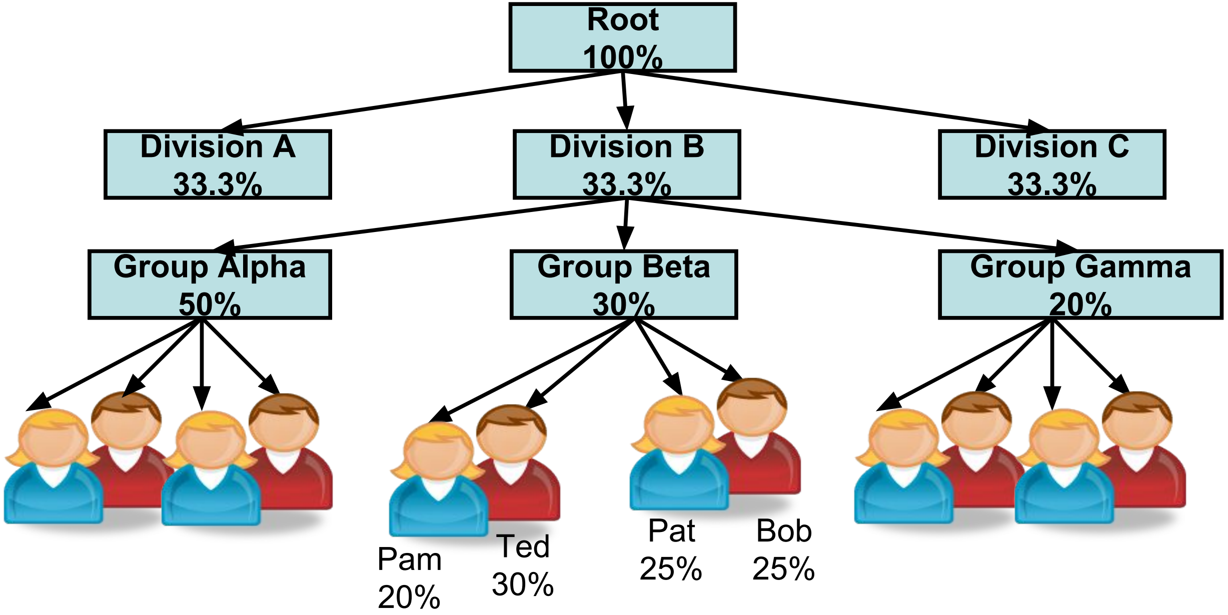
- Roles of resource manager and job scheduler
- Slurm description and design goals
- Slurm architecture and plugins
- Slurm config files and commands
- Accounting

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

> *"All I can say is wow – this is the most flexible, useful scheduling tool I've ever run across."*
> Adam Todorski, Rensselaer Polytechnic Institute

Hierarchichal Account 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, account or QOS (e.g. GrpJobs)
  - A single user may have different shares and limits in different accounts, QOS or partitions

Summary

- Brief overview to have a small mental picture of what is Slurm
- Many more features
  - Job dependencies
  - Fine-grained task layout
  - Wrappers for others WLM commands
  - Burst Buffers, TRES, KNL support, etc.
- Documentation https://slurm.schedmd.com
- Github https://github.com/SchedMD/slurm