Slurm Workload Manager
Introductory User Training

David Bigagli
david@schedmd.com

SchedMD LLC
http://www.schedmd.com

Outline

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

This presentation represents the contents of Slurm version 14.03.
Other versions of Slurm will differ in some respects.

Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- File movement
- 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 or UPC would typically be used to manage communications within the parallel program

Role of a Resource Manager

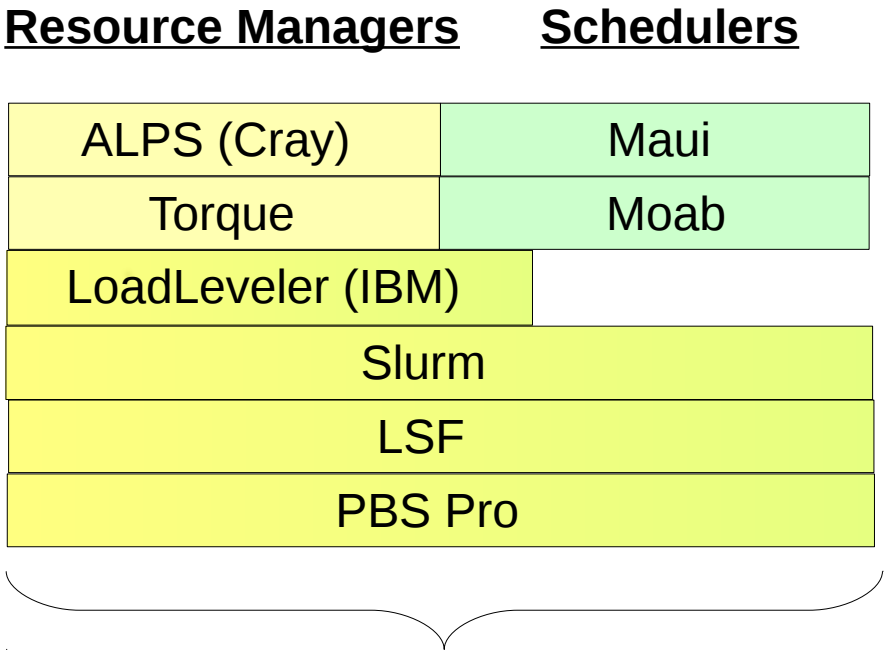
- Allocate resources within a cluster
  - Nodes (typically a unique IP address)
    - NUMA boards
      - Sockets
        - Cores
          - Hyperthreads
      - Memory
    - Interconnect/switch resources
    - Generic resources (e.g. GPUs)
  - Licenses
- Launch and otherwise manage jobs

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

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 (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
- File movement
- Accounting

What is Slurm?

- **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
- Supports AIX, FreeBSD, Linux, Solaris, other Unix variants
- 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

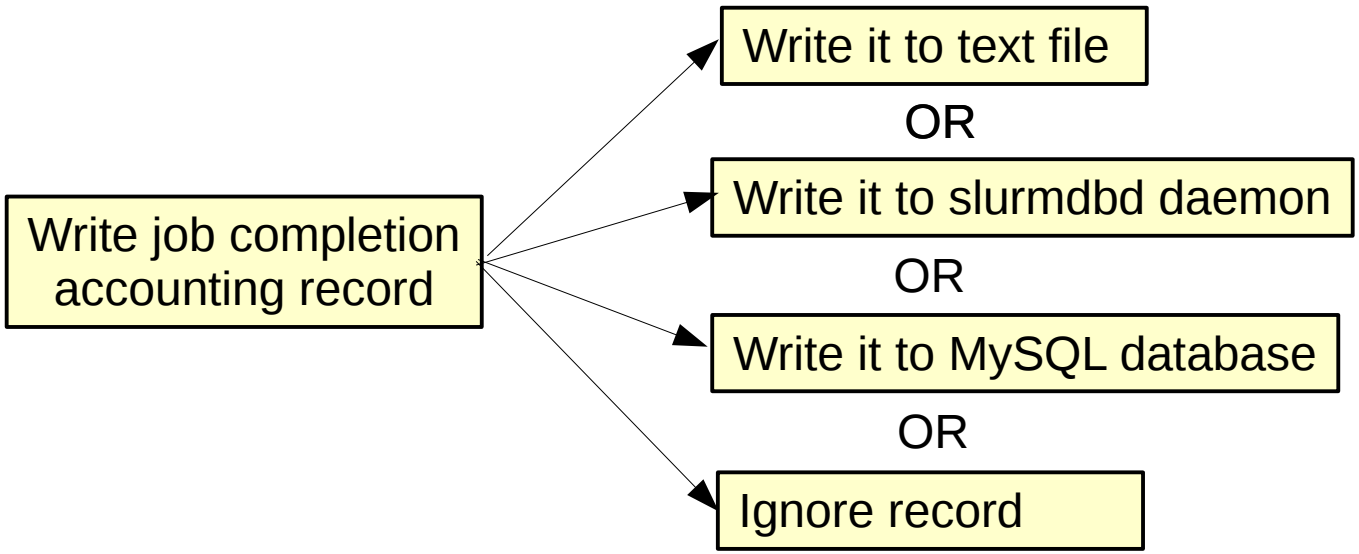
Plugins

- 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, lustre usage, etc

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

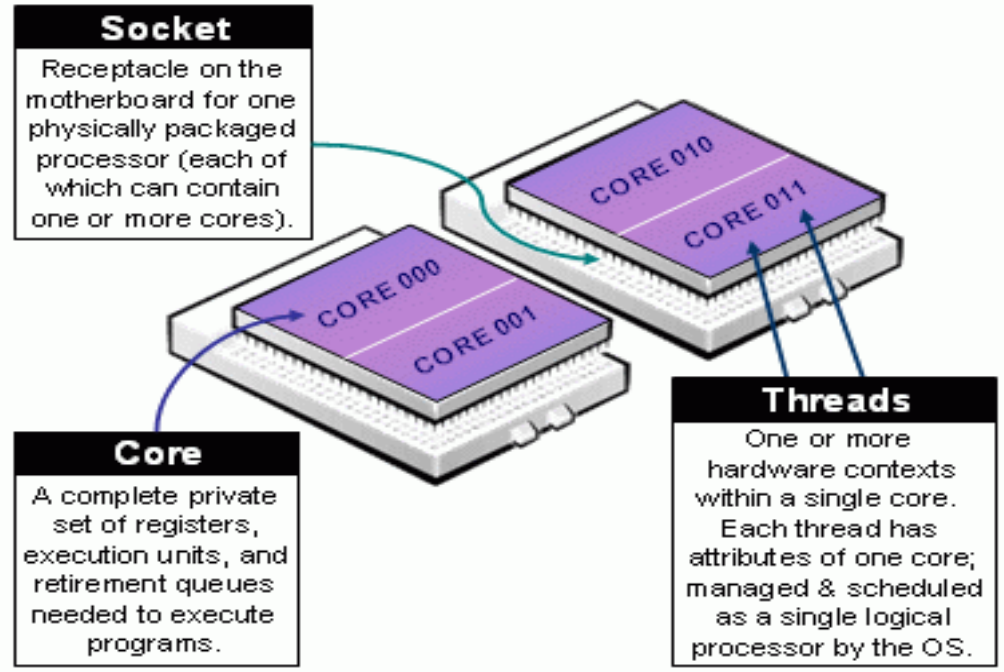
Plugin Design

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



Slurm Resources

- Nodes
  - NUMA boards
    - Sockets
      - Cores
        - Hyperthreads
    - Memory
    - Generic Resources (e.g. GPUs)
- Licenses (generic global resources)

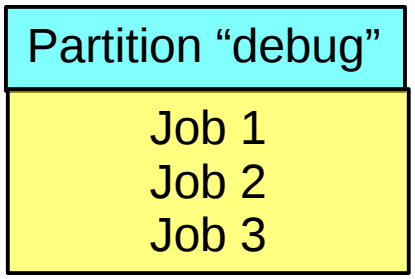


Slurm Entities

- 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
- Partitions: Job queues with limits and access controls

Slurm Entities Example

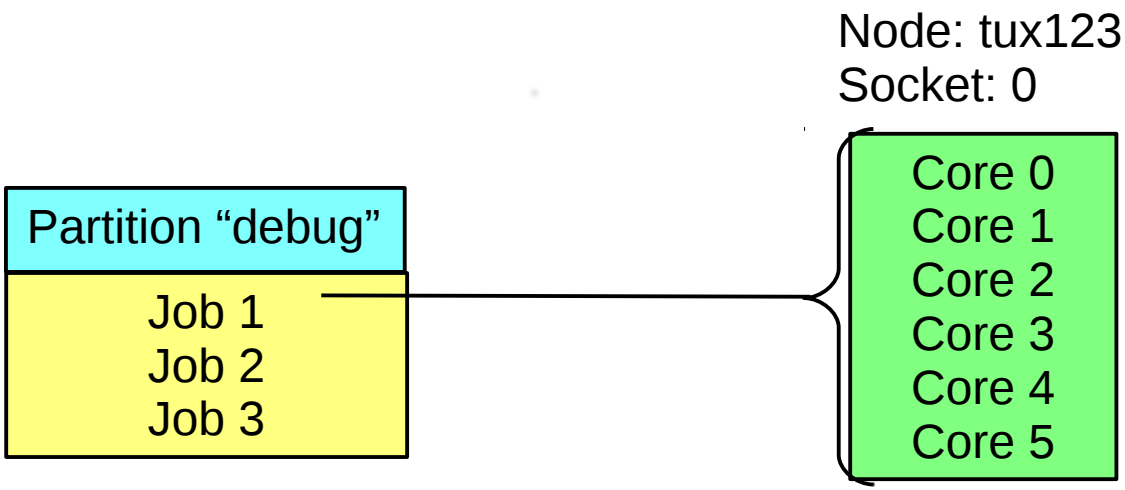
- Users submit jobs to a partition (queue)



Priority ordered queue of jobs

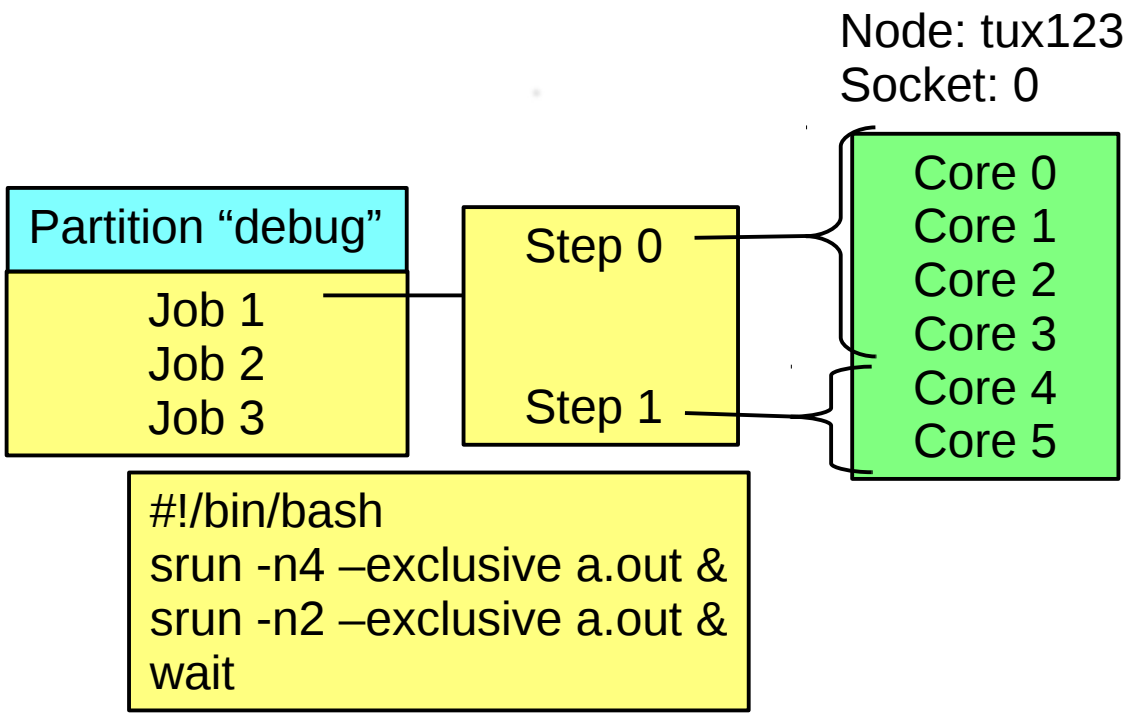
Slurm Entities Example

- Jobs are allocated resources



Slurm Entities Example

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



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]”)

Node Name Expression

- Numeric suffix with comma separated numbers or ranges
  - Used in all commands and configuration files to make managing large clusters easier
  - Bracketed value be at end of name (with optional range elsewhere in the name)
  - tux[0,8]
  - tux[0-15,101]
  - tux[000-127]
  - rack[0-7]_blade[0-63]

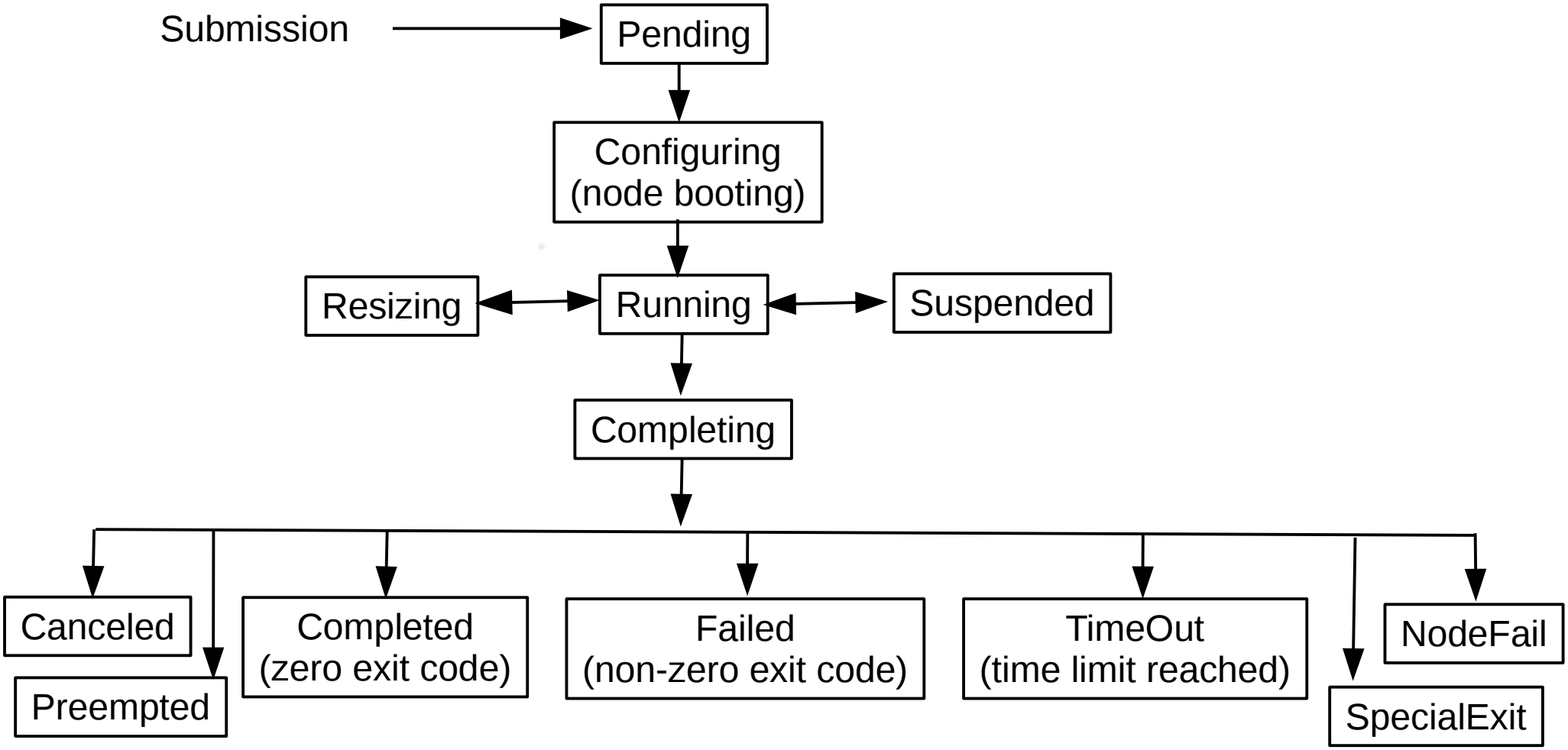
Queue/Partition State Information

- 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 bank 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

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

Cluster Architecture
Typical Linux Cluster

Slurm user tools

Slurmctld (backup)

Slurmctld (master)

Slurmdbd (backup)

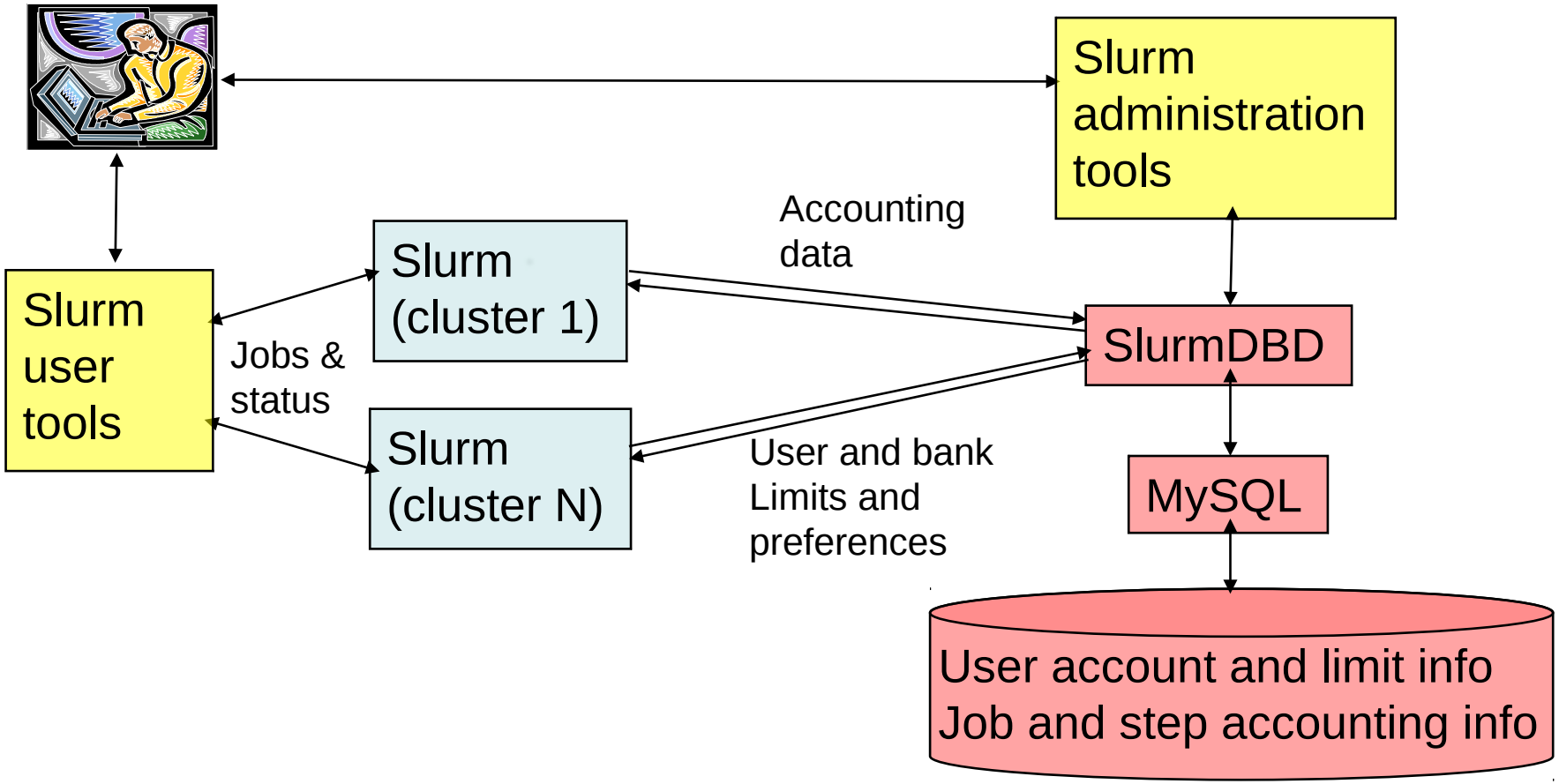
Slurmdbd (master)

MySQL

Accounting and configuration records

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

Typical Enterprise Architecture



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.

Outline

- Roles of resource manager and job scheduler
- Slurm design and architecture
- Submitting and running jobs
- Managing jobs
- File movement
- 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, -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

Time interval formats

- Valid time formats (with a few exceptions)
  - days-hours:minutes:seconds
  - hours:minutes:seconds
  - hours:minutes
  - minutes
  - days-hours
  - days-hours:minutes

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 or job step

Job allocation options

- The job allocation options (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)

sbatch Command

- Used to submit script for later execution
- Convenient to submit work that may execute for an extended period if time
- Convenient for managing a sequence of jobs using dependencies
- The script can contain job options (e.g. size and time limits)
- Job arrays

salloc Command

- Used to get a resource allocation and use it interactively from your computer terminal
- Typically spawns a shell with various Slurm environment variables set
- Depending upon Slurm configuration
  - The shell can be on the login node OR
  - The shell can be an allocated compute node
- Job steps can be launched from the salloc shell

srun Command

- If invoked from within *salloc* shell or *sbatch* script (i.e. within an existing job allocation), *srun* launches application on compute nodes
- Otherwise, *srun* creates a job allocation (like *salloc)* and then launches application on compute nodes
- By default, *srun* uses entire job allocation based upon job allocation specification (from *salloc* or *sbatch)*
- *srun* can use a subset of the job's resources
- Thousands of job steps can be run serially or in parallel within the job's allocation

Flexible Job Scheduling

- Jobs can be submitted to more than one partition
  - The job will run in the queue where it can start soonest
  - sbatch –partition=debug,batch …
- Jobs can specify a minimum and maximum node count
  - The job will start as soon as possible, reducing size as needed to be started by the backfill scheduler
  - sbatch –nodes=16-32 ...

Flexible Job Scheduling

- Jobs can specify a minimum and maximum wall time limit
  - The job's time limit will be set to the maximum possible value in the specified range and start as soon as possible
  - The job's time limit will be established within that range when scheduled and not be changed later
  - For a job desiring a 10 hour time limit, but willing to start earlier with a 4 hour time limit:

    sbatch –time=10:00:00 –time-min=4:00:00 ...

MPMD – Multiple Program, Multiple Data

- Different programs may be launched by task ID with different arguments
- Use “--multi-prog” option and specify configuration file instead of executable program
- Configuration file lists task IDs, executable programs, and arguments (“%t” mapped to task ID and “%o” mapped to offset within task ID range)

```
> cat master.conf
#TaskID  Program              Arguments
0        /usr/me/master
1-4      /usr/me/slave      --rank=%o

> srun –ntasks=5 –multi-prog master.conf
```

MPI Support

- Many different MPI implementations are supported:
  - MPICH1, MPICH2, MVAPICH, OpenMPI, etc.
- Many use srun to launch the tasks directly
- Some use “mpirun” or another tool within an existing Slurm allocation (they reference Slurm environment variables to determine what resources are allocated to the job)
- Details are online:
  http://www.schedmd.com/slurmdocs/mpi_guide.html

Outline

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

Job Management Commands

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

When will my jobs run?

- Use “squeue –start” command
- Jobs will be listed in order expected start time
- Requires backfill scheduler to estimate times
  - Depending upon configuration, not all jobs may have a start time set
- Times are only estimates and subject to change
- Lists one (possibly of several) reasons why the job has not started yet

```
> squeue --start
JOBID PARTITION  NAME  USER  ST  START_TIME          NODES NODELIST(REASON)
10104  debug     foo   adam  PD  2014-02-13T12:20:35      1 (Resources)
10100  debug     bar   bob   PD  2014-02-13T12:33:57      1 (Priority)
10107  debug     a.out dave  PD  2014-02-13T17:00:00      1 (Reservation)
```

Why is my job not running?

- As soon as some reason is found why a job can not 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 bank account job limit reached
AssociationResourceLimit	User or bank account resource limit reached
AssociationTimeLimit	User or bank 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

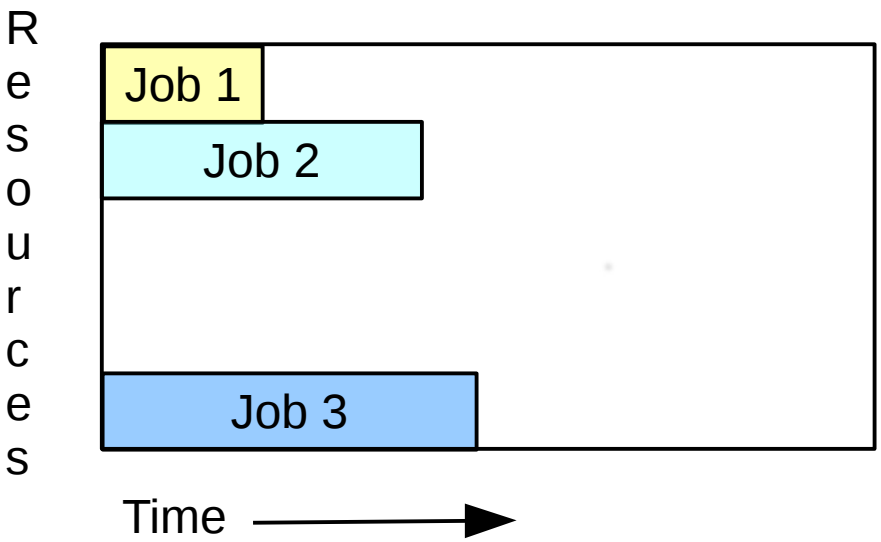
Scheduling Algorithms

- Slurm is designed to perform a quick and simple scheduling attempt at frequent intervals
  - At each job submission
  - At job completion on each of it's compute nodes
  - At configuration changes
- Slower and more comprehensive scheduling attempts less frequently
  - Frequency is configuration dependent

Backfill Scheduling

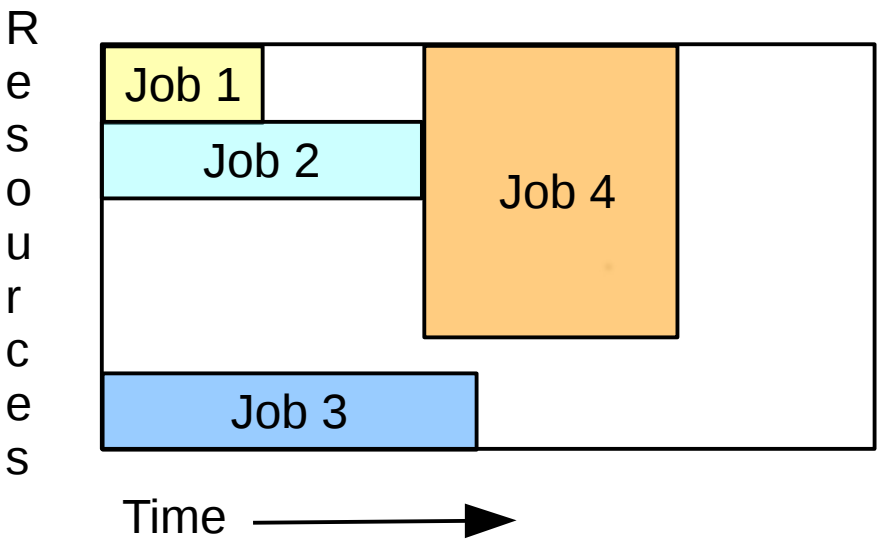
- Backfill scheduling plugin is loaded by default
- Without backfill, each partition is scheduled strictly in priority order
- Backfill scheduler will start lower priority jobs if doing so does not delay the expected start time of any higher priority job
- Since the expected start time of pending jobs depends upon the expected completion time of running jobs, reasonably accurate time limits are critical for backfill scheduling to work well

Backfill Scheduling



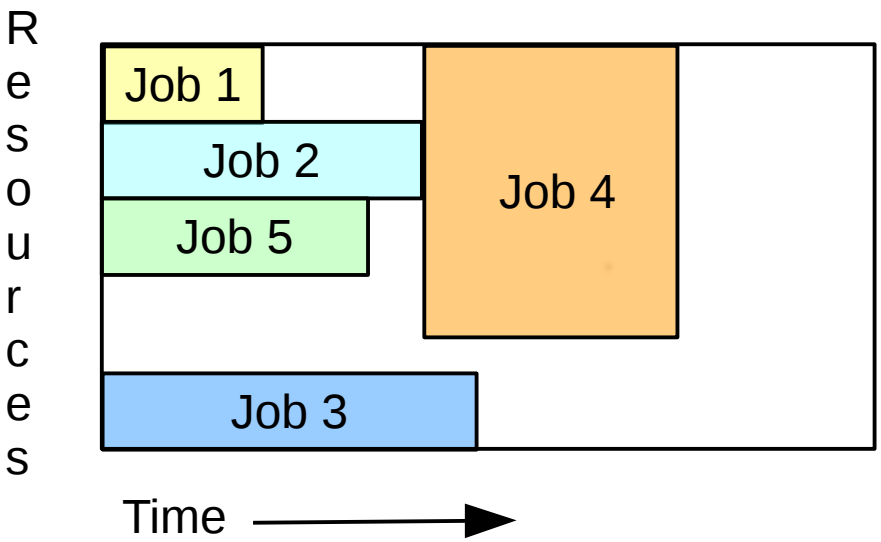
1. Jobs 1, 2 and 3 running now

Backfill Scheduling



1. Jobs 1, 2 and 3 running now
2. Job 4 can not start now, waiting for job 2

Backfill Scheduling

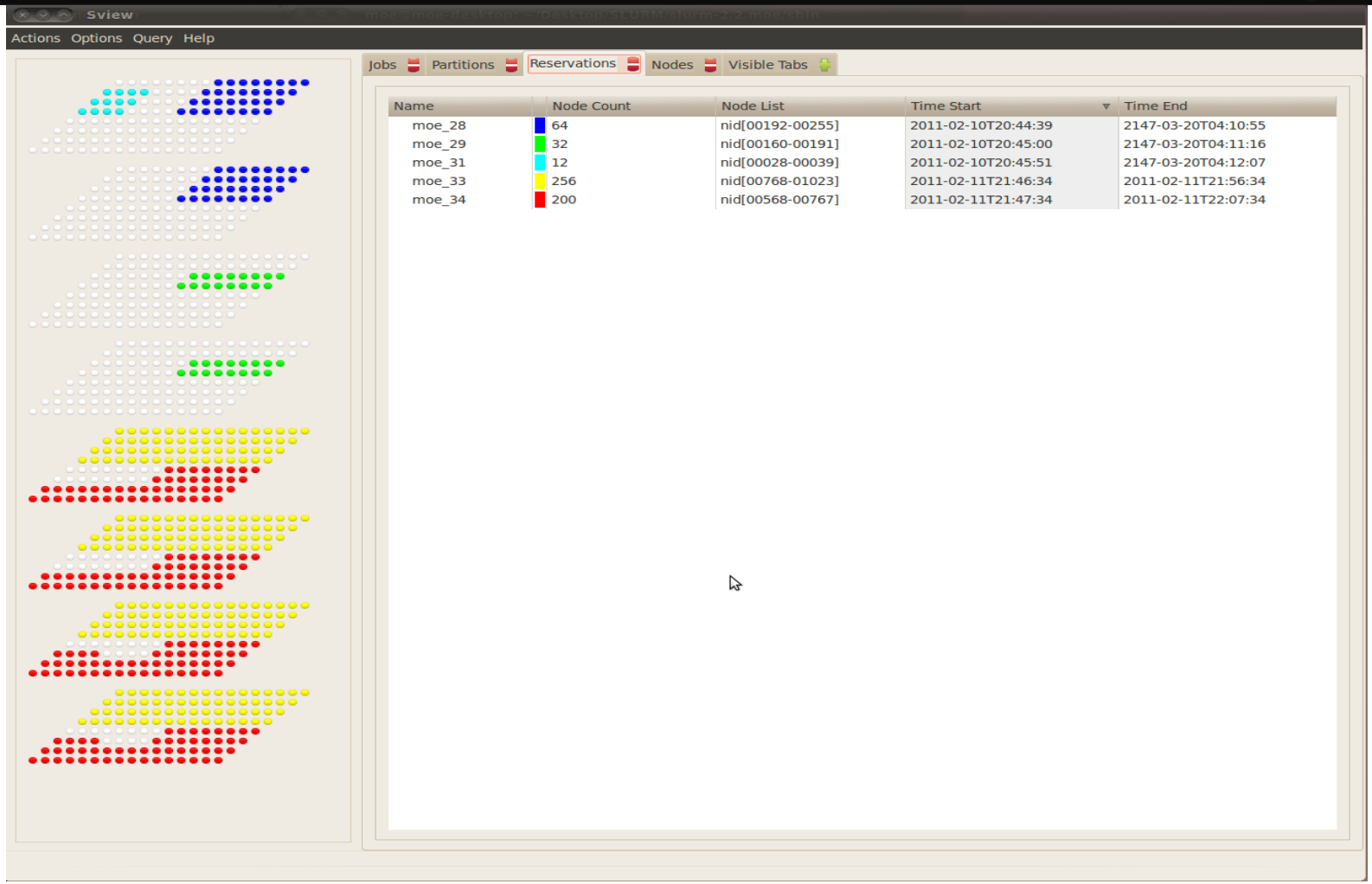


1. Jobs 1, 2 and 3 running now
2. Job 4 can not start now, waiting for job 2
3. Job 5 can start now on available resources, but only if it completes no later than job 2
4. Backfill scheduler goes down queue a configurable depth, determining where and when each pending job will run

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 on Cray (3-D torus)



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.

Outline

- Roles of resource manager and job scheduler
- SLURM design and architecture
- Submitting and running jobs
- Managing jobs
- File movement
- Accounting

No Implicit File Movement

- Slurm provides commands to move files to or from compute nodes (*sbcast* and *sgather* respectively)
- A global file system (e.g. Lustre) may provide better performance than Slurm

sbcast Command

- Copy a file to local disk on allocated nodes
  - Execute command after a resource allocation is made
- Uses *slurmd* daemons and hierarchical communications

```
> salloc -N100 bash
salloc: Granted job allocaiton 45201
> sbcast --force my_data /tmp/moe/my_data  (overwrite old files)
> srun a.out
> exit  (terminate spawned “bash” shell)
```

sgather Command

- Copies file from allocated nodes to login node
- Uses *scp* to move files
- Source node name is appended to local file name
- New in Slurm version 14.03

```
> salloc -N2 bash
salloc: Granted job allocation 45202
> srun a.out
> sgather /tmp/stdout.45202 slurm-45202.out.
> ls slurm-45202.out.*
slurm-45202.out.tux12     slurm-45202.out.tux13
> exit  (terminate spawned “bash” shell)
```

Outline

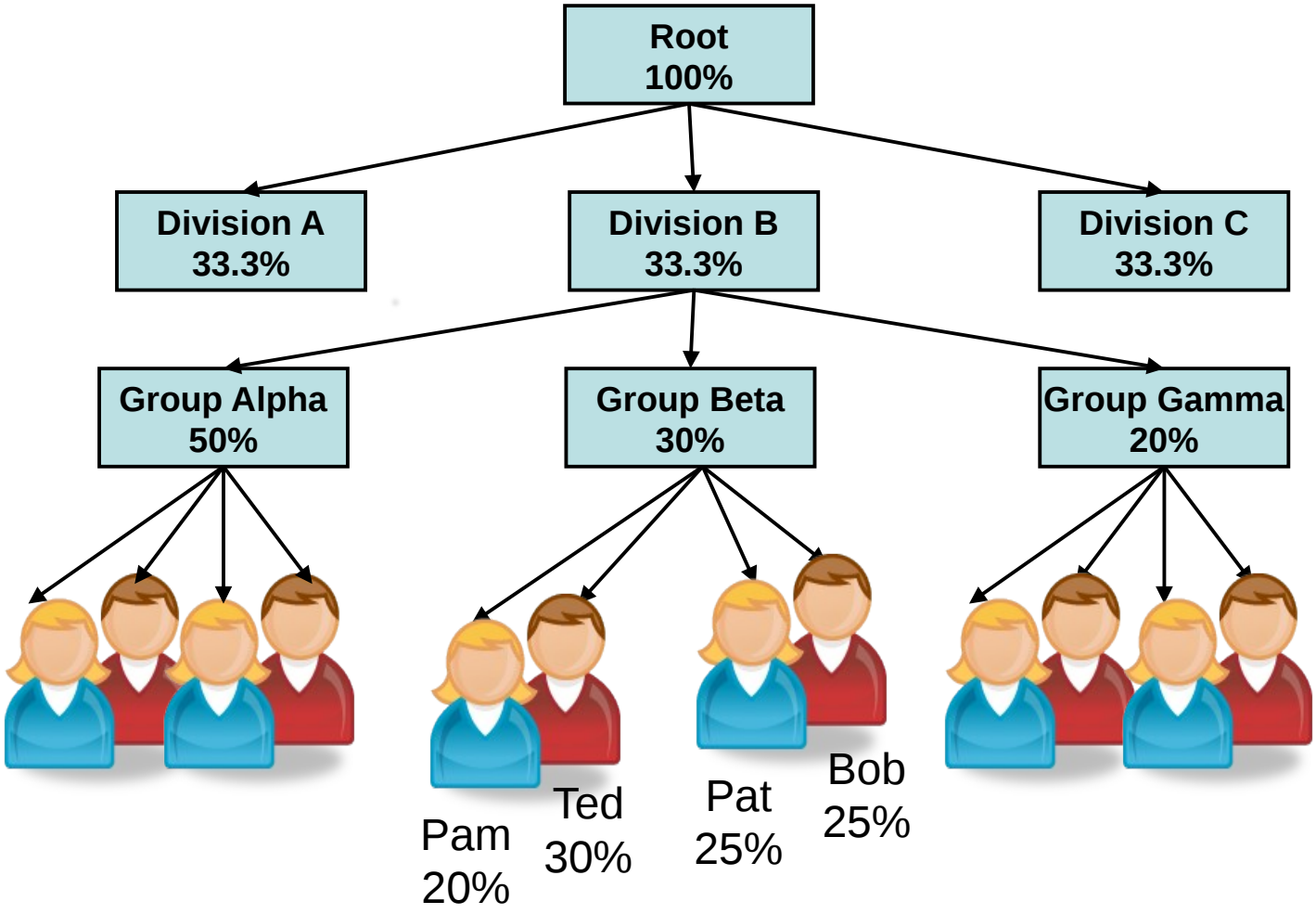
- Roles of resource manager and job scheduler
- SLURM design and architecture
- Submitting and running jobs
- Managing jobs
- File movement
- Accounting

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 banks
    - Limits by user AND by banks

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

Hierarchical Bank Example



Hierarchical Banks

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

Accounting Commands

- **sshare –** Reports bank account hierarchy
- **sprio** – Report components of job priority
- **sacctmgr** – Get and set account data and limits
- **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 understanding of Slurm
  - How it works
  - How to run and manage jobs
- Advanced usage covered in separate class
- Documentation available at http://slurm.schedmd.com
- Questions?

- Demonstration

Slurm Workload Manager
Introductory Admin Training

David Bigagli
david@schedmd.com

SchedMD LLC
http://www.schedmd.com

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

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]”)

Node Configuration Parameters

- NodeName – Name in Slurm
- NodeHostname – Local name, /bin/hostname
- NodeAddr – Name or IP address for communications
- Port – Port number used for slurmd communications
- Features – Arbitrary strings

Node Configuration Parameters

- Boards=# - Baseboard count
- CoresPerSocket=#
- CPUs=#
- RealMemory=# - Real memory in MB
- Sockets=#
- SocketsPerBoard=#
- ThreadsPerCore=#
- TmpDisk=# - Temporary storage size in MB

Node Configuration Parameters

- Features – Arbitrary strings
- Gres – Generic resource names and counts
- Reason – String describing reason for node state (e.g why done or drained)
- State – Typically “UNKNOWN” to recover from saved state, could explicitly be “DOWN”, “DRAIN”, “FAIL”, etc
- Weight=# - Schedules jobs to lowest weight nodes which satisfy job's requirements

DownNodes Parameters

- Can identify bad nodes in separate lines of slurm.conf
- DownNodes=<name> – Same name as NodeName
- Reason – Reason for node being down
- State – Node state

Sample Node Configuration

```
# Node configuration in slurm.conf

NodeName=DEFAULT Sockets=2 CoresPerSocket=8 ThreadsPerCore=2
NodeName=tux[0-1023] RealMemory=2048 Weight=2
NodeName=tux[1024-2000] RealMemory=4096 Weight=4
DownNodes=tux123 State=DOWN Reason=PDU
DownNodes=tux126 State=DOWN Reason=Fan
```

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

Partition Configuration Parameters

- PartitionName – Name of partition/queue
- Default=YES|NO – Default partition for submitted jobs
- Alternate – If this partition can not be used, automatically send jobs to the alternate partition
- Hidden – Don't show by default unless user can use
- LLN – Schedule jobs on to the Least Loaded Node

Partition Configuration Parameters

- AllocNodes – Only jobs submitted from these nodes can use this partiiton
- AllowAccounts – Comma separated list of accounts permitted to use this partition
- DenyAccounts – Accounts prevented from use
- AllowGroups – Comma separated list of groups permitted to use this partition
- AllowQos – Comma separated list of QOS permitted to use this partition
- DenyQos – QOS prevented from use
- DisableRootJobs – Root not permitted to run jobs in this partition

Partition Configuration Parameters

- DefaultMemPerCPU – Default memory allocated to jobs per CPU
- DefaultMemPerNode – Default memory allocated to jobs per node
- MaxMemPerCPU – Maximum memory allocated to jobs per CPU
- MaxMemPerNode – Maximum memory allocated to jobs per node

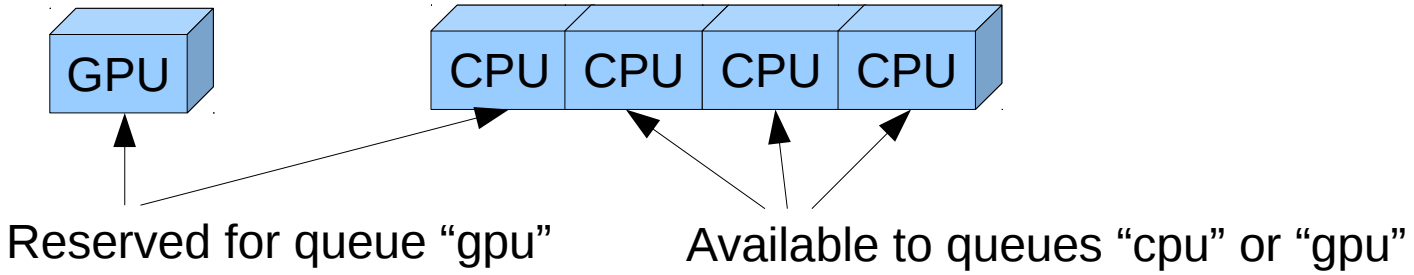
Use per CPU or per node values, not both

Partition Configuration Parameters

- DefaultTime – Default job time limit
- GraceTime – How much the time limit can be exceeded before purging the job
- MaxCPUsPerNode – Useful for nodes in multiple partitions, for example jobs using GPUs on a node can use all of its CPUs, but jobs not using the GPUs must leave some CPUs available for jobs in the partition using GPUs

MaxCPUsPerNode Use Case

- For cluster with 1 GPU and 4 CPUs per node
- Configure 2 queues: “cpu” and “gpu”
- Configure queue “cpu” with MaxCPUsPerNode=3
- Queue “gpu” use restricted to jobs using GPUs (enforce using job_submit plugin)



Partition Configuration Parameters

- MaxNodes=#
- MaxTime=#
- MinNodes=#

Partition Configuration Parameters

- Priority=# - Controls which partition is considered for scheduling first. Also used for preemption. Useful if nodes are in multiple partitions
- PreemptMode – What to do with preempted jobs (e.g. suspend, requeue, cancel, etc.)
- ReqResv – Partition may only be used when job specifies an advanced reservation
- RootOnly – Only user root may submit jobs to this partition (which will run as another user), useful with Moab or Maui scheduler

Partition Configuration Parameters

- SelectTypeParameters – Can be used to specify different resource allocation units, for example some partitions can allocated whole nodes, others sockets, others cores
- Shared – Controls over-subscription of resources, needed for gang scheduling
  - Exclusive, Force, Yes, No
- State
  - Up, Down, Drain, Inactive

Sample Partition Configuration

```
# Partition configuration in slurm.conf

PartitionName=DEFAULT State=UP GraceTime=10 DefMemPerCPU=512 Weight=1
PartitionName=debug Nodes=tux[0-31] MaxTime=30 MaxNodes=4 Default=yes
PartitionName=batch Nodes=tux[32-2000] MaxTime=infinite MaxNodes=infinite
PartitionName=all Nodes=tux[0-2000] MaxTime=30 AllowQos=critical Weight=8
```

Cluster Architecture
Typical Linux Cluster

Slurm user tools

Slurmctld (backup)

Slurmctld (master)

Slurmdbd (backup)

Slurmdbd (master)

MySQL

Accounting and configuration records

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

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

Daemons

- Each daemon is documented in a man page
- Commonly used options
  - -D Run the daemon in foreground (output to terminal)
  - -v Verbose logging, use multiple v's as desired
  - -c Cold-start, ignore any previously saved state
  - -h Print summary of daemon options

slurmctld Daemon

- This is where all of the current node, partition, job and step data is maintained
- It decides when and where to initiate jobs
- It processes almost all of the user commands (except for accounting records)
- Highly multi-threaded
- Typically configured to run as use special user (e.g. user “slurm”)
- Typically one per cluster plus optional backup

slurmdbd Daemon

- slurmdbd == Slurm DataBase Daemon
- Typically one daemon per enterprise
- An intermediary between user and the database
  - Avoid granting users direct database access
    - Database password in secure file
  - Authenticate communications between user and *slurmdbd* (using Munge)
  - Only *slurmdbd* needs permission to read/write the database

slurmdbd Daemon

- Stores accounting information to a database
- Securely returns accounting information
  - Supports access controls
- Processes user requests to set and get some configuration information (limits, fair-share, etc.)
- Uploads configuration information (limits, fair-share, etc.) to *slurmctld* daemon when changed in the database
- *slurmctld* daemon will use cache if *slurmdbd* not responding

More database information in advanced administration class

slurmd Daemon

- Compute node daemon (typically one per compute node)
- Simple, small and lightweight
- Quiescent after job launch except for optional accounting
- Supports hierarchical communications with configurable fanout (e.g. to quickly launch parallel jobs)
- Must run as user root (in order to spawn jobs as various users)

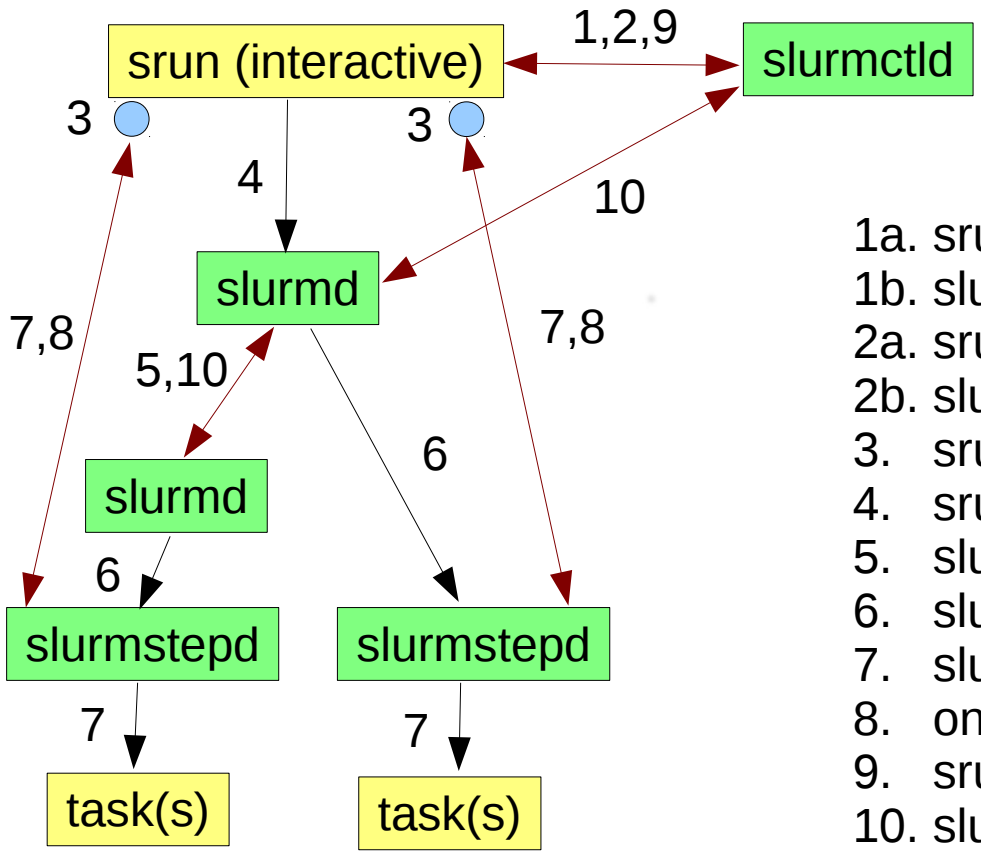
slurmstepd Daemon

- Job step shepherd
- Launched for batch job and each job step (not really a daemon)
- Launches user application tasks
- Manages application I/O, signals, etc.
- Relatively small and simple
- Runs as the job's user

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

Linux Job Launch Sequence



1a. srun sends job allocation request to slurmctld
1b. slurmctld grant allocation and returns details
2a. srun sends step create request to slurmctld
2b. slurmctld responds with step credential
3. srun opens sockets for I/O
4. srun forwards credential with task info to slurmd
5. slurmd forward request as needed (per fanout)
6. slurmd forks/execs slurmstepd
7. slurmstepd connects I/O to run & launches tasks
8. on task termination, slurmstepd notifies srun
9. srun notifies slurmctld of job termination
10. slurmctld verifies termination of all processes via slurmd and releases resources for next job

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

Log Files

- Each daemon writes its own log file
- The verbosity of those logs are control using Slurm configuration files and “-v” options on the execute line OR change any time using the *scontrol* command
- The *slurmd* (compute node daemon) logs should include the node name in the file name (e.g. “SlurmdLogFile=/tmp/slurmd.%n”, where Slurm replaces “%n” with the host name
- The *slurmd* (compute node daemon) logs should be on a local file system for performance reasons

Logging

- Increase verbosity of daemons using the “-v” option on the command line or modify the configuration file (e.g. “SlurmctldDebug=debug3”)
- Each increment in the debug level roughly doubles the volume of information logged and can negatively impact performance at some point
- Debug level of the *slurmctld* daemon can be changed in real time using the *scontrol* command (e.g. “scontrol setdebug debug2”)

Log Files

- Recommend use of the *logrotate* tool to compress and save older log files
- When in doubt about what happened, examine the relevant log file(s), searching for the relevant job ID or node name
- More details and examples later in the tutorial...

Sub-system Specific Logging

- Many sub-systems can be configured to generate their own detailed logging without increasing the verbosity of logging in other sub-systems
- Configure with DebugFlags=
- Modify DebugFlags in the slurmctld daemon any time using *scontrol* (e.g. “scontrol setdebugflags +backfill”)

Available DebugFlag Values

Backfill	Backfill scheduling (the sched/backfill plugin)
BGBlockWires	BlueGene block wiring (switch details)
CPU_Bind	Binding of tasks to CPUs
Energy	Energy use
Gres	Generic Resource allocation and use
Gang	Gang scheduling
License	License scheduling
Priority	Job priority
Reservations	Advanced reservations
Steps	Job step activities
Switch	Network resources (e.g. the switch plugin)
Triggers	Event triggers

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

Job Prioritization

- Priority/multifactor plugin assigns priorities to jobs based upon configurable factors

```
JobPriority =
   PriorityWeightAge * AgeFactor +
   PriorityWeightFairshare * FairShareFactor +
   PriorityWeightJobSize * JobSizeFactor +
   PriorityWeightPartition * PartitionFactor +
   PriorityWeightQOS * QosFactor
```

Job Priority is 32-bit integer
Factors all floating point numbers between 0 and 1.0
PriorityWeight values all 32-bit integers

IMPORTANT: Set PriorityWeight values high to generate wide range of job priorities

Age Factor

- Represents time the job has been eligible to run and waiting
- Define **PriorityMaxAge** as job age at which **AgeFactor** reaches 1.0 (the maximum)

Job Size Factor

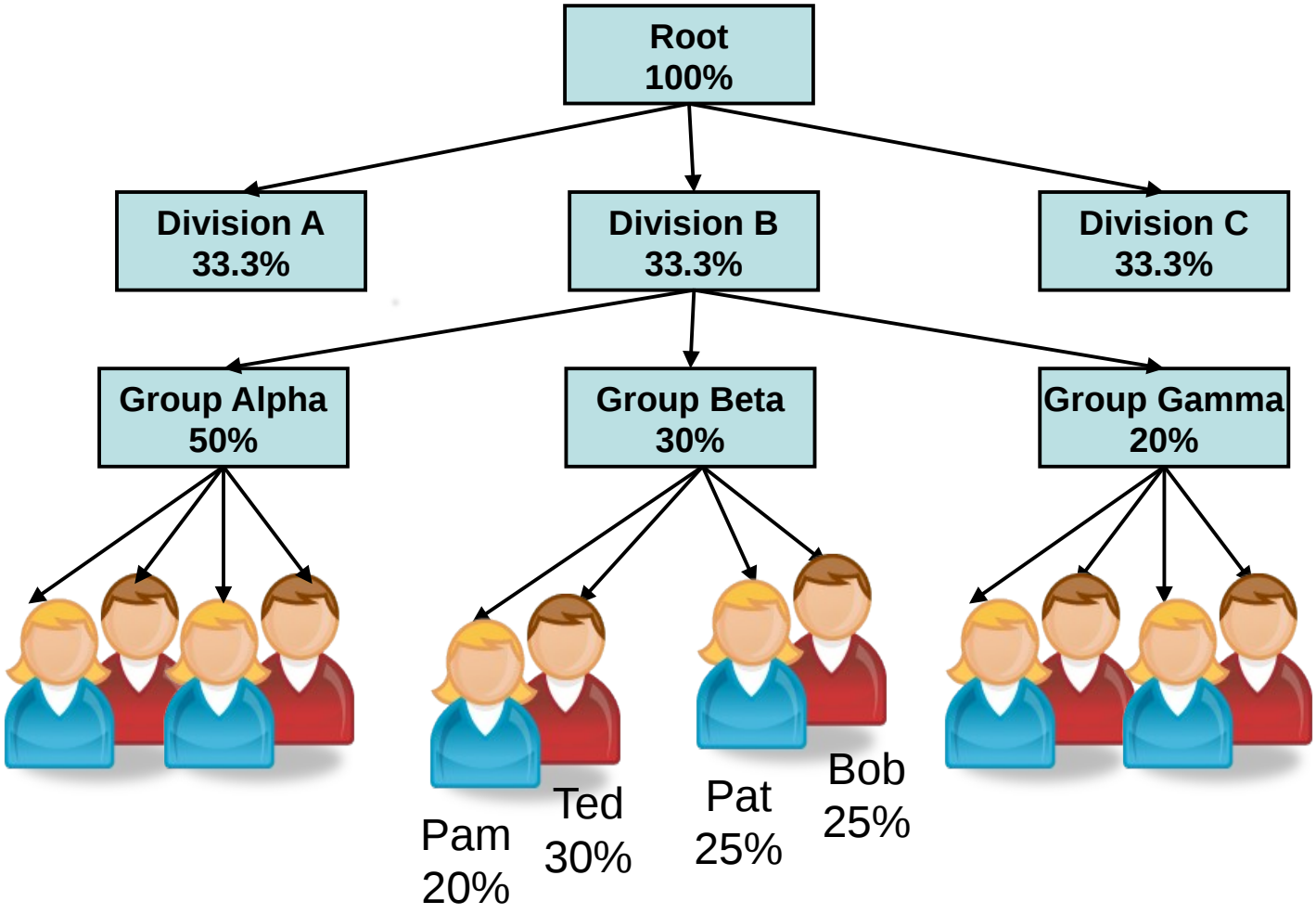
- Configuration parameter **PriorityFavorSmall** will give highest priority to smallest jobs. By default, larger jobs are favored
- **JobSizeFactor** ranges from 0.0 to 1.0 for 1 CPU to full-system job (or vice-versa if **PriorityFavorSmall** set)
- If **PriorityFlags** value of SMALL_RELATIVE_TO_TIME is set, then the job size in CPUs is divided by the time limit in minutes then divided by the number of CPUs in the system. A full system job with time limit of 1 minute will have **JobSizeFactor** of 1.0. Small jobs with large time limits will have a **JobSizeFactor** close to 0.

Partition and QOS Factors

- Each partition and QOS has a Priority factor
- This can be used to favor certain partitions or jobs with a certain QOS within a given partition

Partition normal
  Job 1 QOS high
  Job 2 QOS high
  Job 3 QOS normal
  Job 4 QOS normal
  Job 5 QOS low

Hierarchical Accounts Example



Fairshare

- Tries to allocate to each user and account the appropriate percentage of system resources (Currently CPU time)
- Configurable decay of recorded usage: **PriorityDecayHalfLife**
- Configurable job priority calculation interval: **PriorityCalcPeriod**
- Slurm considers each level in the hierarchy to generate a single FairShareFactor
  - What should be done if an account's share of resources is 20% and the only active user in that account has a 2% share of resources?
    - Give the user only 2% of resources and leave the other resources for other users in the account?
    - Give the user 20% of resources and deplete the resources available to other users in the account?

Fairshare – Normal Mode

- Calculate how over- or under-serviced each account in the hierarchy is and try to balance access over time
- The example user is expected to get more than 2% of resources and his peers will get less than 18% if/when they become active
- http://slurm.schedmd.com/priority_multifactor.html

Fairshare – Ticket Based

- “Tickets” are distributed to all active accounts and users in the hierarchy in proportion to their shares
- The example user is expected to get the full account's 20% of resources/tickets
- The ticket based system works best when there are many inactive users
- Configure **PriorityFlags=TICKET_BASED**
- http://slurm.schedmd.com/priority_multifactor2.html

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

Slurm Commands: Accounting

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

sprio Command

- Reports components of a job's priority
- Reports on all pending jobs by default, but can filter by job ID or user ID

```
> sprio
 JOBID  PRIORITY  AGE  FAIRSHARE  JOBSIZE  PARTITION   QOS
 65539      62664    0          51664      1000          10000        0
 65540      62663    0          51663      1000          10000        0
 65541      62662    0          51662      1000          10000        0

Values normalized to one
> sprio --norm
 JOBID   PRIORITY       AGE  FAIRSHARE  JOBSIZE  PARTITION       QOS
 65539 0.00001459 0.0007180  0.5166470  1.0000000  1.0000000  0.0000000
 65540 0.00001459 0.0007180  0.5166370  1.0000000  1.0000000  0.0000000
 65541 0.00001458 0.0007180  0.5166270  1.0000000  1.0000000  0.0000000
```

sacct Command

- Reports accounting information for individual jobs and steps
- Many filtering and output format options
- Uses accounting database (which may not exist depending upon Slurm configuration)

Report accounting information for user “joseph”
> sacct -u joseph

Report accounting information for partition “debug”
> sacct -p debug

sreport Command

- Reports accounting information aggregated by user, bank account, etc. not for individual jobs
- Many filtering and output format options
- Uses accounting database (which may not exist depending upon Slurm configuration)

Report utilization of cluster "tux" by allocated, idle, down, reserved, etc.
> sreport cluster utilization cluster=tux start=01/01/14 end=02/01/14

Report utilization of cluster "tux" by bank account and user
> sreport cluster AccountUtilizationByUser cluster=tux start=01/01/14 end=02/01/14

sacctmgr Command

- Use to check user, bank and QOS limits
- Access to data may be limited

```
> sacctmgr show association where user=jette
 Cluster   Account     User  Partition  Share   GrpJobs GrpNodes ...
--------- _---------- --------- ----------- --------- ------------ -----------
    cray       admin      jette                 100           30       128
    hp         admin      jette                 100                    64
    ibm        admin      jette                 100           10       256

> sacctmgr show association where account=admin cluster=cray
 Cluster   Account     User  Partition  Share   GrpJobs GrpNodes ...
---------- --------- ----------- --------- ------------ -----------
    cray       admin      jette                 100           30       128
    cray       admin     danny                  200           30       128
    cray       admin       phil                  50           30       128
```

Outline

- Nodes and Partitions
- Daemons
- Job Launch
- Logging and log file
- Scheduling
- Admin Commands
- Infrastructure

Commonly Used Infrastructure

- See: http://slurm.schedmd.com/download.html
- **Munge** (Munge Uid “n” Gid Emporium) provides authentication, digital signature and encryption mechanism
  - Requires “key” on each node
  - Need the “munge-dev” package with headers and library
- **NPT** (Network Time Protocol) provides uniform time across the cluster as needed by Munge
- **MySQL** or **MariaDB** database for accounting and configuration information
  - Need the “client-dev” packages with headers and library
- **hwloc** Portable Hardware Locality library and tools to report on-node topology (NUMA, Sockets, Cores, Threads, Cache, etc.)

Thank you!

- Q & A