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
- Launch and otherwise manage jobs

Requires 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, QoS: by queue, user, group, etc.
Examples

Resource Managers | Schedulers
---|---
ALPS (Cray) | Maui
Torque | Moab
LoadLeveler (IBM) | Slurm
Slurm | LSF
LSF | PBS Pro
PBS Pro | Slurm started as a resource manager (the “rm” in “Slurm”) and added scheduling logic later

Many span both roles
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:
  - Simple Linux Utility for Resource Management
- 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?
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.

<table>
<thead>
<tr>
<th>Slurm Kernel (65% of code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication Plugin</td>
</tr>
<tr>
<td>MUNGE</td>
</tr>
</tbody>
</table>
Plugin Design

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

```

<table>
<thead>
<tr>
<th>slurmctld</th>
<th>slurmdbd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write job completion accounting record</td>
<td>Write job completion accounting record</td>
</tr>
<tr>
<td>accounting_storage/slurmdbd</td>
<td>accounting_storage/mysql</td>
</tr>
<tr>
<td>Send to SlurmDBD</td>
<td>Write to MySQL database</td>
</tr>
</tbody>
</table>

```
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

Copyright 2018 SchedMD LLC
www.schedmd.com
SLUG Sept. 25&26th, 2018
Slurm Entities Example

- Users submit jobs to partitions (queue):

Priority ordered queue of jobs

<table>
<thead>
<tr>
<th>Partition</th>
<th>debug</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job 1</td>
<td></td>
</tr>
<tr>
<td>Job 2</td>
<td></td>
</tr>
<tr>
<td>Job 3</td>
<td></td>
</tr>
</tbody>
</table>
Jobs are allocated resources:

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

Partition *debug*

- Job 1
- Job 2
- Job 3

Node: tux123
- Socket: 0
- Core 0
- Core 1
- Core 2
- Core 3
- Core 4
- Core 5
Slurm Entities Example

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

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

Partition **debug**

<table>
<thead>
<tr>
<th>Job 1</th>
<th>Job 2</th>
<th>Job 3</th>
</tr>
</thead>
</table>

Step 0

Step 1

Node: tux123
Socket: 0

- Core 0
- Core 1
- Core 2
- Core 3
- Core 4
- Core 5
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

- Submission
- Pending
- Configuring (node booting)
- Resizing
- Running
- Suspended
- Completing
  - Canceled (zero exit code)
  - Preempted
  - Completed (non-zero exit code)
  - Failed (non-zero exit code)
  - TimeOut (time limit reached)
  - NodeFail
  - SpecialExit
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

Slurm user tools

Slurmctld (master)

Slurmctld (backup)

Slurmdbd (master)

Slurmdbd (backup)

MySQL

Accounting and configuration records

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

Slurm user tools

Slurm (cluster 1)

Accounting data

Slurm (cluster N)

Jobs & status

Slurm administration tools

Slurmdbd

MySQL

User account and limit info

Job and step accounting info

User and account limits and preferences
Configuration

- General conf
- Plugin activation
- Scheduler parameters
- Node definition
- Partition definition
- Slurmdbd params.
- Archive/Purge
- Storage options
Configuration

- 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).
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
sr

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.

<table>
<thead>
<tr>
<th>Some common reasons why jobs are pending:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
</tr>
<tr>
<td>Resources</td>
</tr>
<tr>
<td>Dependency</td>
</tr>
<tr>
<td>Reservation</td>
</tr>
<tr>
<td>AssociationJobLimit</td>
</tr>
<tr>
<td>AssociationResourceLimit</td>
</tr>
<tr>
<td>AssociationTimeLimit</td>
</tr>
<tr>
<td>QOSJobLimit</td>
</tr>
<tr>
<td>QOSResourceLimit</td>
</tr>
<tr>
<td>QOSTimeLimit</td>
</tr>
<tr>
<td>Resources being reserved for higher priority job</td>
</tr>
<tr>
<td>Required resources are in use</td>
</tr>
<tr>
<td>Job dependencies not yet satisfied</td>
</tr>
<tr>
<td>Waiting for advanced reservation</td>
</tr>
<tr>
<td>User or account job limit reached</td>
</tr>
<tr>
<td>User or account resource limit reached</td>
</tr>
<tr>
<td>User or account time limit reached</td>
</tr>
<tr>
<td>Quality Of Service (QOS) job limit reached</td>
</tr>
<tr>
<td>Quality Of Service (QOS) resource limit reached</td>
</tr>
<tr>
<td>Quality Of Service (QOS) time limit reached</td>
</tr>
</tbody>
</table>
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

Root 100%

- Division A 33.3%
  - Group Alpha 50%
    - Pam 20%
    - Ted 30%
- Division B 33.3%
  - Group Beta 30%
    - Pat 25%
    - Bob 25%
- Division C 33.3%
  - Group Gamma 20%
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?