SLURM Version 2.2: Features and Release Plans

Supercomputing 2010

16 November 2010

Morris Jette (jette1@llnl.gov)

Danny Auble (auble1@llnl.gov)

Donald Lipari (lipari1@llnl.gov)

S&T Principal Directorate - Computation Directorate

LLNL-PRES-461787

Agenda

- Major enhancements in version 2.2
- Release schedule for version 2.2
- SLURM plans for 2011 and beyond

What Changes in SLURM Version 2.2

A lot.
There are major changes in many areas.

Commands can Operate Between Clusters

- The client and server do not need to be running the same version of SLURM
- *SlurmDBD* required and must have the latest minor version (*slurmctld* v2.2.# requires *SlurmDBD* v2.2.#)
- The client and server do not need to be running on the same architecture (e.g. BlueGene and Cray or traditional Linux cluster)
- Use the –clusters=<name> *or* –M <name> option on SLURM command line or SLURM_CLUSTERS environment variable. Default value is the current cluster.

Commands can Operate Between Clusters (continued)

- Batch job will be sent to the one cluster with the earliest expected start time from the list of clusters specified. It will not migrate after job submission
- New *sbatch* option *–export* or *SBATCH_EXPORT* environment variables control what environment variables get propagated
- There is currently NO spooling of files between clusters. Global file systems required for input files

Limit and QOS (Quality of Service) Enhancements

- *MaxCPUs*: Maximum number of CPUs any one job in this association can use
- *GrpCPUs*: Maximum number of CPUs all jobs in this association can use
- Default QOS per association
- QOS flags added to override partition limits
- Default account by cluster
- Fair-share usage threshold to hold pending jobs

Many *sview* Enhancements

- Default configuration (preferences) saved in ~/.slurm/sviewrc file
- Switch between clusters viewed
- Select multiple jobs, partitions, etc.
- View database configuration
- Add and remove visible tabs
- Better highlighting of selected rows

Added Support for User Hold of Jobs

- Submit job using *sbatch* or *srun –hold* or *–H* option
- Hold and release using *scontrol* command
  - *scontrol hold <jobid>*
  - *scontrol release <jobid>*
- User can not release jobs held by system administrator
- Job *Reason* reported by *squeue* and *scontrol*
  - *JobHeldUser* if held by user
  - *JobHeldAdmin* if held by system administrator

Enhanced Permissions for Operators, etc.

- Enhanced permissions for operators, administrators, and account coordinators (as configured in the database) not running as root
  - Cancel or requeue any user's job
  - Create, delete, or modify partitions
  - Create, delete, or modify reservation

Added Derived Exit Code and String

- Gives the user a means to annotate completed jobs in the database
- Derived exit code is determined by control daemon as the highest exit code of all the job steps
- Derived exit code field appears in SLURM's job record and is sent to the database
- Database job record contains the derived exit code and a derived exit string initially set to NULL
- User can modify either of these fields after job completes. E.g.,
  - Can add an explanation of why a job failed
  - Can indicate when a successful job produced bad results

Management of Generic Resources (GRES)

- Generic resources can be defined on a per-node basis and consumed by jobs and job steps
- Generic resources can be associated with specific device files and (later) access controlled using *cgroups*
- The *gres/gpu* plugin currently controls access using an environment variable *CUDA_VISIBLE_DEVICES*

Generic Resource Configuration and Use (example)

```
# slurm.conf (excerpt)
GresTypes=gpu
NodeName=linux[0-15] Sockets=4 CoresPerSocket=2 Gres=gpu:4
```

```
# gres.conf (from compute node)
Name=gpu File=/dev/nvidia[0-3]
```

```
# Launch batch job on one node with 4 CPUs and 2 GPUs
$ sbatch –N1 –n4  -–gres=gpu:2 my.script
```

```
# Environment variable set for the batch job
CUDA_VISIBLE_DEVICES=0,1
```

Jobs can Specify a Time Limit Range

- The *–time* or *–t* option specifies the maximum time limit
- A new option *–time-min* specifies the minimum acceptable job run time, default is same as *–time*
- Job will receive its maximum time limit unless reducing the time permits backfill scheduling to start it earlier
- The job's time limit does not change after starting execution (needed for jobs to calculate remaining time consistently)

Running Jobs can Decrease in Size

- *scontrol* option to decrease a job's size by specifying a new node count or specific nodes to use
  - *scontrol update JobId=<id> NumNodes=<count>*
  - *scontrol update JobId=<id> NodeList=<names>*

- *scontrol* generates a script to be executed to reset job's environment variables

```
#bin/sh
# Do parallel work
srun my.work
# Release all but one node
scontrol update jobid=$SLURM_JOBID NumNodes=1
. slurm_job_${SLURM_JOBID}_resize.sh
srun my.post.processing
```

Major Improvements for High Throughput Computing

- MySQL database restructured for 50 to 75% speedup
- Multiple job record send to *SlurmDBD* in single RPC

- General improvements in scheduling algorithms

- Additional *SchedulerParameters* for tuning
  - *Default_queue_depth* (default job count for scheduling, default is 100, previously no limit)
  - *Interval* (for *sched/backfill*, in seconds)
  - *Max_job_bf* (for *sched/backfill*, job count)

- *MinJobAge* parameter can now purge jobs more quickly

Additional Partition States

State	Queue new jobs	Run queued jobs
Up	Yes	Yes
Down	Yes	No
Drain (new)	No	Yes
Inactive (new)	No	No

An *Alternate* partition parameter has also been added. Jobs submitted to a partition in *Drain* or *Inactive* state will automatically be transferred to the *Alternate* partition (if any).

Added Job Submit Plugin

- Called by *slurmctld* daemon for each job submit or job modification call
- Can be used to customize environment by site or user
- Sample use:
  - Set default job partition (queue) based upon job characteristics
- Plugin has C or LUA (script) interface

Job Preemption More Configurable

- The mechanism used to preempt jobs can be configured on a per partition or per QOS (Quality Of Service)

- Sample configuration:
  - Jobs in standby QOS get requeued
  - Jobs in normal QOS get suspended and resumed

Many *DebugFlags* Added

- Generates detailed logging for specific sub-systems
  - *Backfill*: Backfill scheduling
  - *CPU_Bind*: CPU binding details for job and steps
  - *Gang*: Gang scheduling
  - *GRES*: Generic Resources
  - *Priority*: Job priority calculation
  - *Reservation*: Advanced reservations
  - *Steps*: Resource allocation for job steps
  - *Triggers*: Event triggers
  - And many more

Reduced Fragmentation with Consumable Resources Plugin

- Old logic would identify nodes to use then evenly distribute tasks
- New logic packs allocation onto nodes (subject to job specifications). Idle resources normally located on one node

Reduced Fragmentation with Consumable Resources Plugin

Example: Allocate 10 tasks on two node, each with 8 CPUS
New logic leaves unused resources all on one node

Node 0	Node 1
Task 0	Task 1
Task 2	Task 3
Task 4	Task 5
Task 6	Task 7
Task 8	Task 9
Unused	Unused
Unused	Unused
Unused	Unused

Node 0	Node 1
Task 0	Task 1
Task 2	Task 3
Task 4	Unused
Task 5	Unused
Task 6	Unused
Task 7	Unused
Task 8	Unused
Task 9	Unused

Queue or Run Time Added to E-Mail Notifications

SLURM Job_id=123 Name=my_job Began, Queued time 01:23:45

SLURM Job_id=123 Name=my_job Ended, Run time 1-00:15:20

Time format:
[days-]hours:minutes:seconds

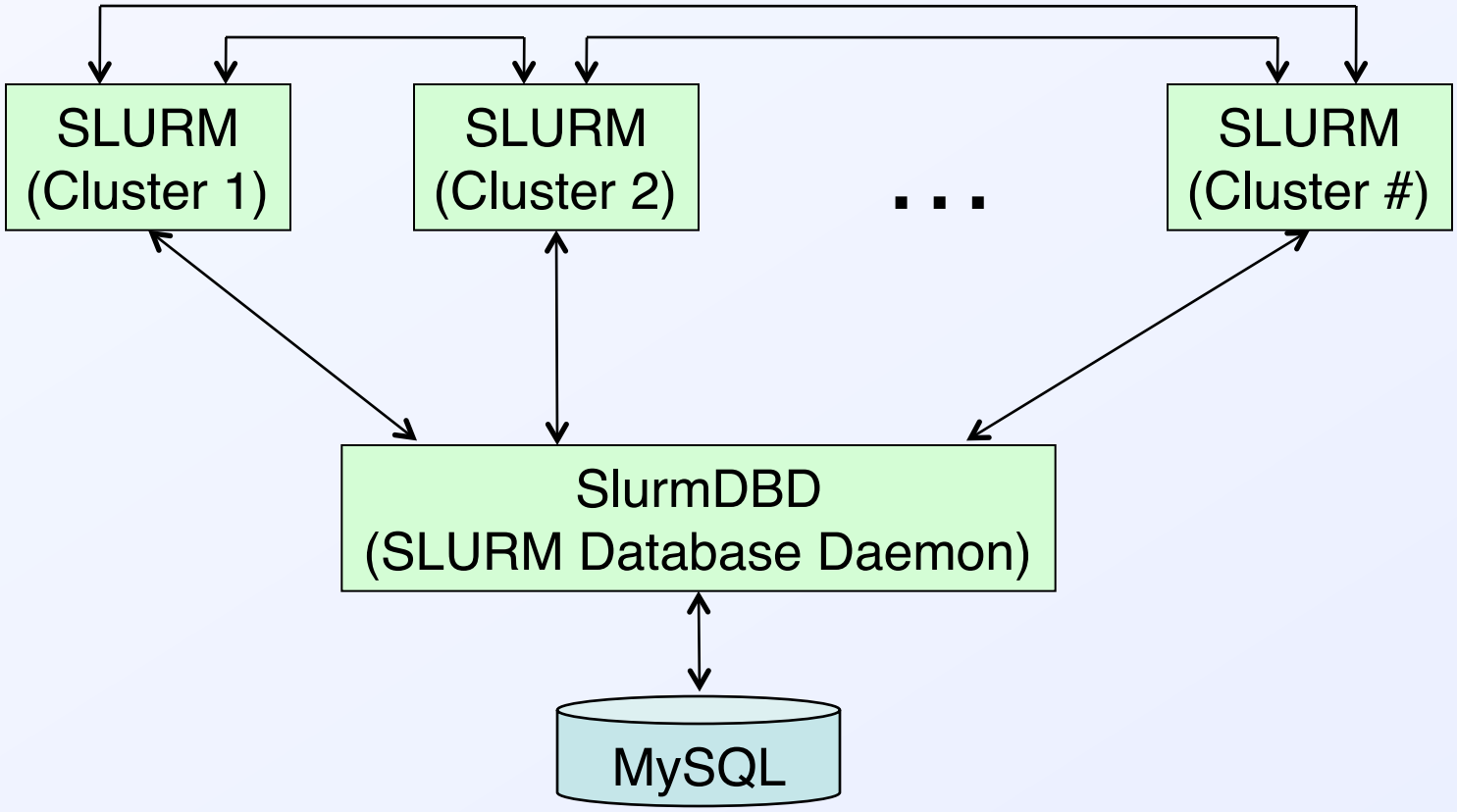
TotalView support to attach to subset of tasks

- Better scalability than attaching to all tasks
- Disable with –*disable-partial-attach* option to *configure* (build) script

Other Enhancements

- State preserved when upgrading from version 2.1 (including running jobs)
- Jobs can specify multiple partitions (queues) and use the first available
- Test added for circular job dependencies
- Perl APIs available for SlurmDBD communications
- Additional event triggers (by Bull)
  - Triggers for state changes in database, *SlurmDBD*, and *Slurmctld*
- Some scalability enhancements, mostly by NUDT for Tianhe-1A (3,072 nodes, 14,336 processors, 2.5 Pflop)

SLURM Job Scheduling, Typical Version 2.2 Configuration



Release Schedule for Version 2.2

- Development stopped in early November
- Spending November and December testing
  - There is a fairly stable version available now
  - http://sourceforge.net/projects/slurm/files/under_development/
- Release in December or when very stable

Tentative Plans for Version 2.3

- Release SLURM version 2.3 about May 2011
- Support for Linux *cgroups* (job containers, by Bull)
  - Integrate with *PAM*
- Some infrastructure for BlueGene/Q systems (by LLNL)
- Support for Cray XE and XT systems (by CSCS)?

SLURM Plans for 2011 at LLNL

- Focus at LLNL in 2011 on port to BlueGene/Q
  - 20 Pflops, 5-D torus interconnect
  - Completely new interface for managing network, booting nodes, etc.
- Add support for multiple slurmd daemons on systems with “front-end” architecture (e.g. BlueGene)
  - Better scalability and fault-tolerance
- Better support for generic resources (e.g. GPUs)
  - Use Linux cgroups to bind tasks to resources

Areas of Interest, 2011 and Beyond

- Better fault tolerance for user applications (e.g. hot-spare nodes)
- Replace *mpirun* with *srun* on BlueGene systems
  - Uniform interface across architectures
- Faster task launch
  - In user space, without *slurmctld* daemon
- Support for running jobs to grow in size

Areas of Interest, 2011 and Beyond (continued)

- Advanced resource reservation enhancements
  - Topology aware resource reservation
  - Better integration with gang scheduling
  - Query to identify where and when reservations can be created
  - Floating reservations (start early if possible)
- Integrate license management with FlexLM
- Better checkpoint/restart integration for fault tolerance

Areas of Interest, 2011 and Beyond (continued)

- Better enterprise-wide job scheduling
  - Job migration for workload changes
  - Cross-cluster file spooling

Questions and Comments?

Disclaimer

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.