SLURM Status Report
Supercomputing 2008
November 2008

Morris Jette (jette1@llnl.gov)

Danny Auble (auble1@llnl.gov)

S&T Principal Directorate - Computation Directorate

Lawrence Livermore National Laboratory

LLNL-PRES-408510

Agenda

- Recent enhancements to version 1.3
- Plans for version 1.4
- Questions and comments

Recent enhancements to version 1.3

- Many accounting enhancements
- Some enhancements to sched/backfill for support of more job constraints
- Improved reliability for job re-queue
- *Salloc* command no longer requires specification of command to execute. User's default shell or a program defined in *SallocDefaultCommand* configuration parameter (if specified) is started
- For Moab: User's resource limits and default environment variables loaded when a batch job is initiated rather than at submit time (less delay for Moab)

Accounting

- Development is ongoing in SLURM version 1.3
- Central database can collect information from multiple clusters
  - Different Munge authentication keys can be used for communications within each cluster and between clusters for better security
- Information managed by "*association*": cluster, user, bank account and (optionally) SLURM partition/queue
- SLURM optionally enforces job use of valid bank account
  - Default bank available by user
  - Bank account structure is hierarchical

Moab use of SLURM database

- The database can also serve as repository for Moab configuration information (limits and fair-share)
  - Similar functionality to Gold, but 100x faster and secure (communications are authenticated)
  - We have modified Moab's identity mapping program to upload configuration from the database whenever Moab's configuration file (moab.cfg) is read
  - We reboot the Moab daemon daily to get revised configuration information
  - The identity mapping program we use is based upon work by Cluster Resources. It will be available through CRI soon

Current LLNL workload scheduling architecture

sacctmgr
sacct
sreport
(admin tools)

User and account limits and shares

moab.cfg

Job and step
details

SLURM
(cluster 1)

SLURM
(cluster 2)

Moab
Workload
Manager

SlurmDBD

Association info

MySQL

User account and limit info
Job and step accounting info

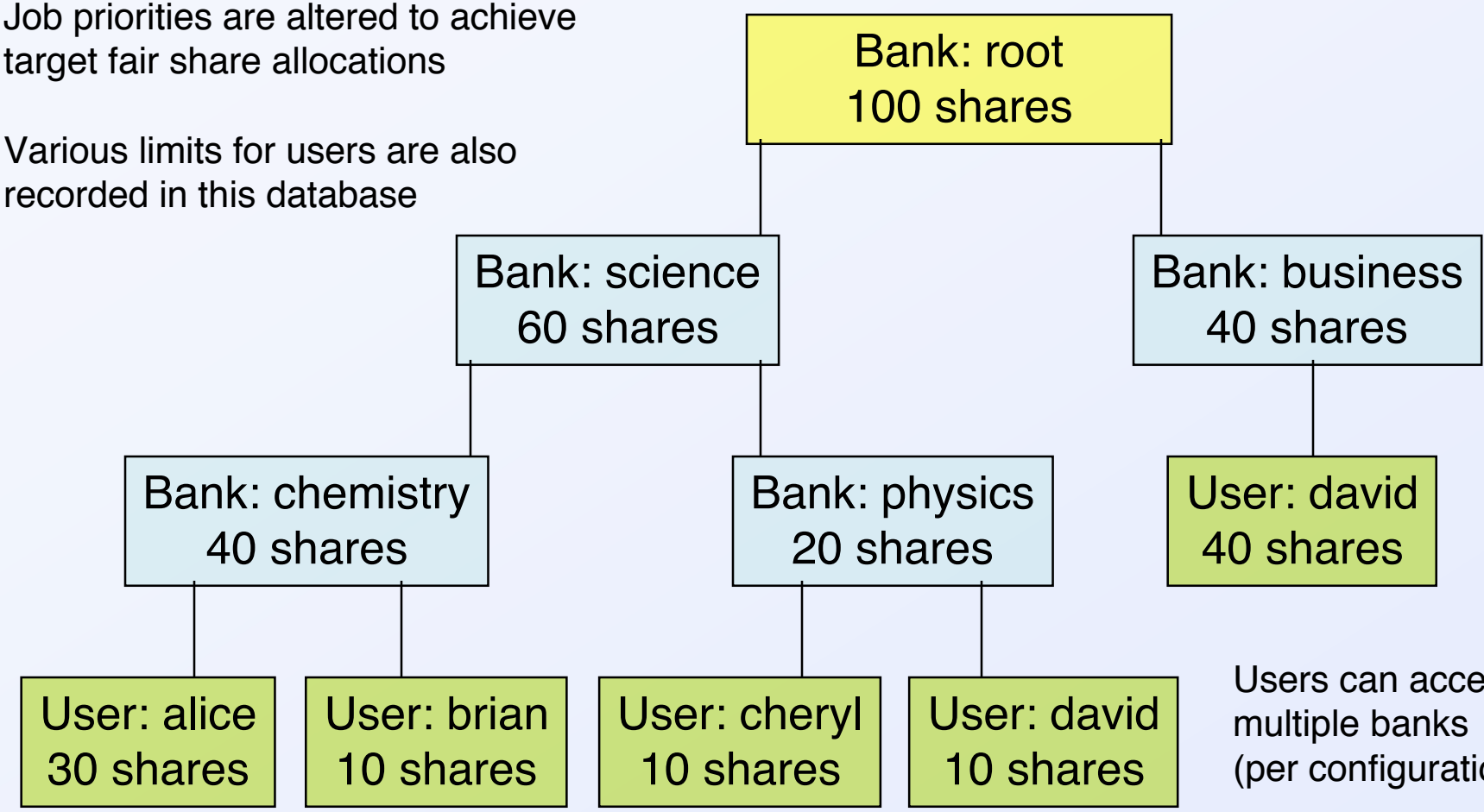
SLURM account info updated in real time

Moab is restarted daily to upload current limits and usage info

Hierarchical banks example

Job priorities are altered to achieve target fair share allocations

Various limits for users are also recorded in this database



Users can access multiple banks (per configuration)

Accounting (continued)

- New tool, *sacctmgr*, is used to manage the database
  - Management of specific portions of the bank hierarchy can be delegated to select individuals
    - They can control users access, fair-share and various limits
- New tool, *sreport*, is used to generate accounting reports by cluster, bank account, user, etc.
- The *sacct* tool can be used generate accounting reports by job ID or step ID
- Access to database is configurable
- All tools can be used from any computer to view or manage all computers at the site

- Lots more information at *https://computing.llnl.gov/linux/slurm/accounting.html*
- Also see the man pages for each command

Plans for Version 1.4 - 2nd Quarter 2009

- Improved multi-core support
  - Required for gang scheduling
  - New configuration options control default and maximum memory per allocated CPU (or node) for jobs
- Preemption of jobs based upon queue priority
- Power down idle nodes
- Topology aware scheduling
- Boot different operating systems by job
- *CryptoType=munge* new default for digital signatures on messages
  - OpenSSL not used in default configuration

Multi-core support in SLURM version 1.3

- In SLURM version 1.3, the *slurmctld* (control daemon) only keeps track of the count of processors allocated on each node
  - The *slurmd* selects specific processors for tasks at launch time
  - *slurmctld* is not aware of task binding at the socket/core level
  - Gang scheduling or other preemptive scheduling is not possible with task affinity and more than one job per node since the *slurmctld* lacks task binding details

Multi-core support in SLURM version 1.4

- In SLURM version 1.4, the *slurmctld* (control daemon) allocates specific sockets/cores to each job and step
  - A bitmap is used to communicate task binding details
  - The *slurmd* launches tasks on the select resources
  - Gang scheduling becomes a possibility with more than one job per node

Example: Job A can time slice with Job C, but not Job B

	Core 0	Core 1	Core 2	Core 3
Time 1	Job A	Job A	Job B	Job B
Time 2	Job C	Job C	Job B	Job B

Memory allocation

- Configuration parameters can control default and maximum memory a job can be allocated on a per node or per processor basis
  - For example, on an 8-processor node, limit a job to 1/8th of memory for each processor it is allocated
- Critical to avoid oversubscribing memory when more than one job is running per node
  - Multiple jobs could run when allocating individual processors to jobs
  - Multiple jobs can are also run through gang scheduling or other preemptive scheduling
- Enforcement through periodical checking status of processes (part of accounting mechanism)
- Use of Linux containers for memory management expected in a future release

Preemption of jobs by partition (queue) priority

- SLURM can be configured to preempt jobs in lower priority queues for jobs in higher priority queues

- For example, you might configure a queue "*expedite*" and another queue called "*batch*" with different priorities, access controls, limits, etc. but having access to the same nodes. When there is a pending job in the *expedite* queue, it could preempt jobs in the *batch* queue to start immediately. The preempted jobs would be resumed once the job from the *expedite* queue completes.

Power down idle nodes

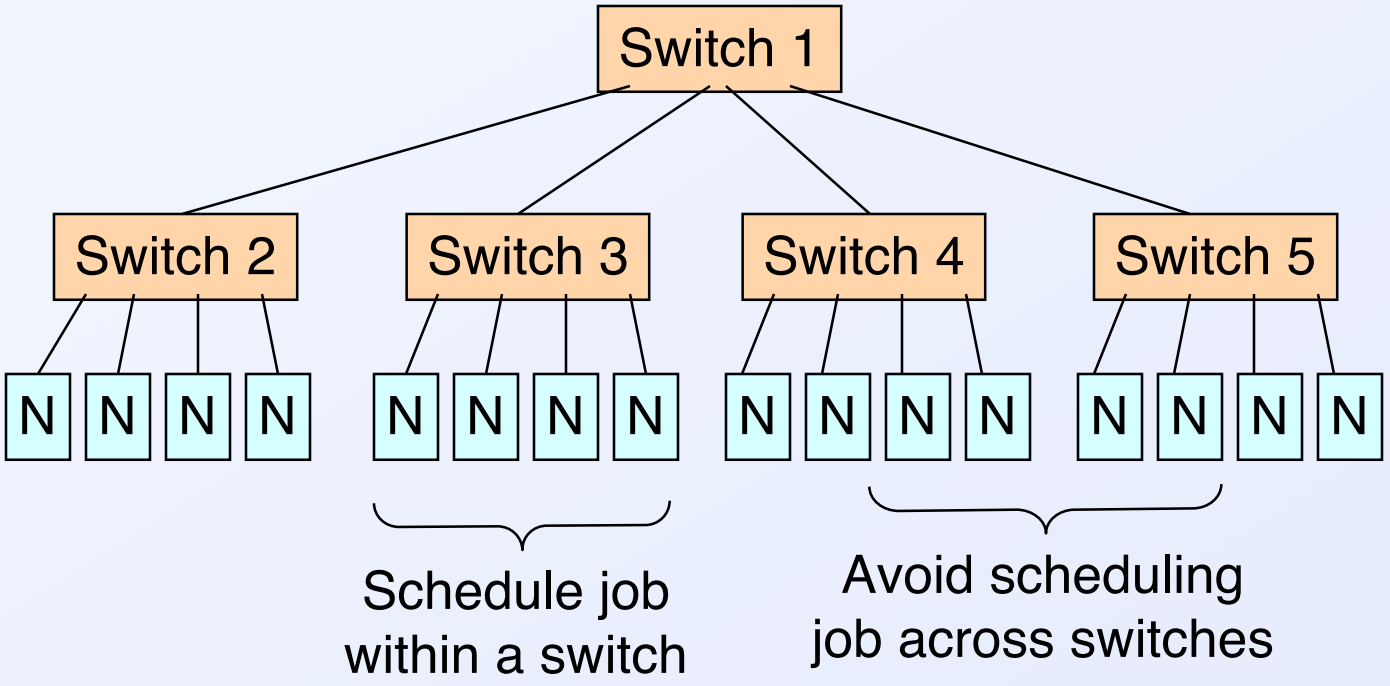
- Version 1.3 had support for decreasing power consumption on idle nodes using a variety of configuration options (timers and programs), but the node could not be powered down (*slurmd* must be kept running)

- Version 1.4 lets you power down the idle nodes and power them back up when jobs are allocated to them
  - No effort will be made to contact these nodes and new jobs will not be initiated on those nodes until the *slurmd* daemon responds
  - Node state will be “IDLE+POWER”

Topology aware scheduling

- Allocate resources to jobs so as to minimize network contention, especially for Infiniband switches

Example: Allocate resources to 4-node job



Boot different operating system by job

- *SlurmctldProlog* configuration parameter added
  - Executed by *slurmctld* before a job begins execution

- Suggested mode of operation
  - Job can specify the operating system to be used with a job constraint
  - The job will be scheduled on nodes with the specified feature
  - *SlurmctldProlog* will be responsible for booting the node with the specified constraint
  - Job's *Comment* field could be used to specify other information such as storage requirements, etc.

Your turn

- Questions?
- Comments?
- Feature requests?

For more information about SLURM

- Information: https://computing.llnl.gov/linux/slurm/
- Downloads: http://sourceforge.net/projects/slurm/
- Email: slurm-dev@lists.llnl.gov

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.