Resource Management using SLURM

The 7th International Conference on Linux Clusters
University of Oklahoma
May 1, 2006

Morris Jette (jette1@llnl.gov)
Lawrence Livermore National Laboratory

http://www.llnl.gov/linux/slurm

UCRL-PRES-219562

Disclaimer

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livremore National Laboratory under Contract No. W-7405-Eng-48.

Overview

- The role of a resource manager
- Design issues for resource management on large-scale clusters
- SLURM architecture
- SLURM commands and their use
- SLURM configuration
- Demonstration of SLURM build, installation, configuration and use
- Special topics

Role of Resource Manager (RM)

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

Role of Resource Manager

- Allocate resources within a cluster to jobs
  - Nodes
    - Processors
    - Memory
    - Disk space
  - Interconnect/switch resources
    - Switch windows

Can necessitate extensive hardware knowledge and interactions (e.g. establish switch wiring and boot nodes on BlueGene)

- Launch and otherwise manage jobs
- Typically light-weight and well suited for highly parallel computers and jobs
- Examples: SLURM, Torque, Quadrics RMS

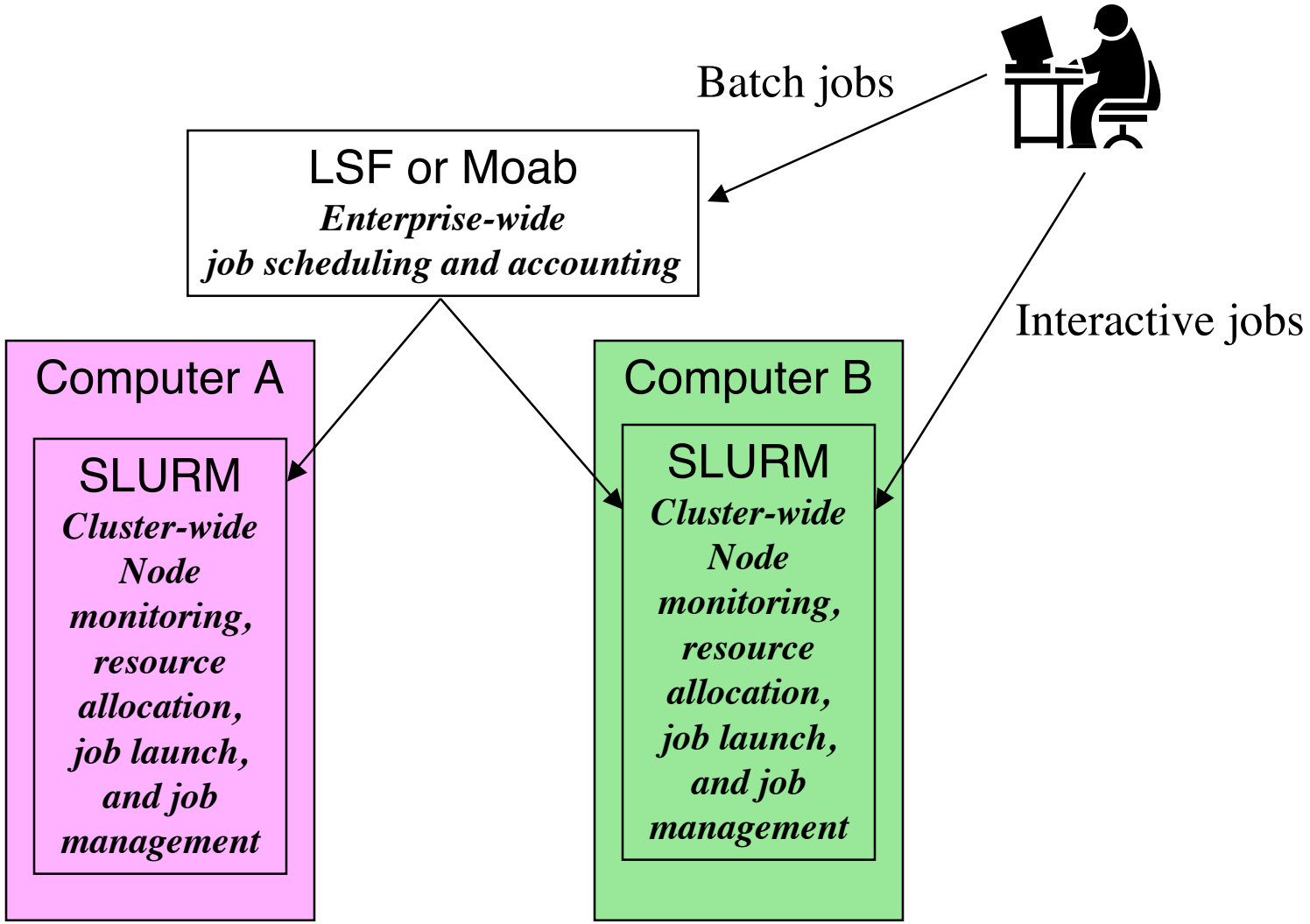
Role of Meta-Scheduler
(Resource Manager at a Higher Level)

- Allocate resources on one or more computers to jobs
  - Nodes
    - Processors
    - Memory
    - Disk space

Typically lacks extensive hardware knowledge or interactions

- Launch and otherwise manage jobs
- Typically heavy-weight (many daemons)
- Typically complex to configure and administer
- Examples: LSF, Moab

A Typical Configuration



Are both schedulers needed?

- Perhaps, but that depends upon the system architecture and workload
- If workload prioritization is not critical, a separate batch system may not be needed
  - Simple queuing (FIFO or conservative backfill scheduling)
  - No movement of jobs between computers
- Depending upon the computer architecture and workload, a separate resource manager may not be needed
  - Ethernet or InfiniBand
  - Serial or moderately parallel jobs
- Some schedulers operate at both levels, but not very well: LoadLeveler and LSF

Large-Scale RM Issues

- Highly parallellized components
  - Lots of threads
  - Separate read and write locks on various data types
- Minimize communications
  - Avoid proliferation of daemons
- Highly optimized algorithms
  - Use bitmaps in scheduling logic
- Fault-tolerance
  - No single point of failure
- Eliminate “system noise”
  - Quiescent daemon on allocated nodes
  - Any use is synchronized across the cluster
- Compact representation of data
  - “linux[0-1023]” instead of “linux0,linux1, linux2,…”

Introducing SLURM

- SLURM (Simple Linux Utility for Resource Management) has become a very popular resource manager
- It is production quality and used on many of the largest computers in the world
- It was developed primarily for Linux clusters, but also supports BlueGene and IBM SP systems with the Federation switch
- Developers include LLNL, HP, Linux NetworX, PathScale, North Dakota State University, Indiana University and others

Key SLURM Features

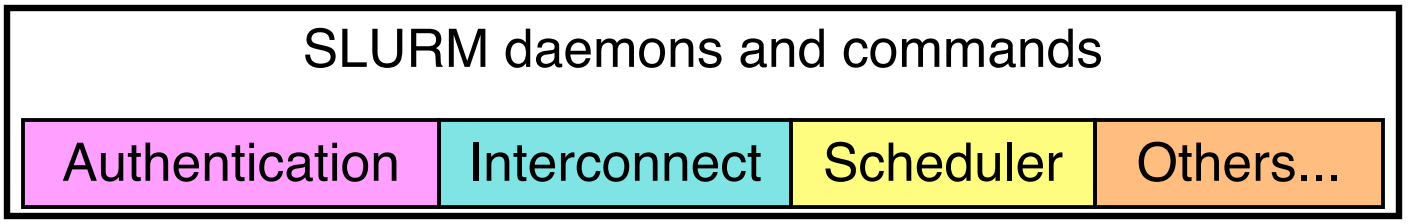
- Simple (relatively)
  - Scheduling complexity external to SLURM
- Open source: GPL
- Portable (see next slide)
- Fault-tolerant
  - For SLURM daemons and its jobs
- Secure
  - Authentication plugin
- System administrator friendly
  - Simple configuration file, supports heterogeneous clusters
- Scalable to the largest computers (16k nodes, 128k processors)

SLURM Portability

- No kernel modifications
- C-language
- *Autoconf* configuration engine
- Provides skeleton of functionality with general-purpose plugin mechanism. A highly flexible building block approach to configuration

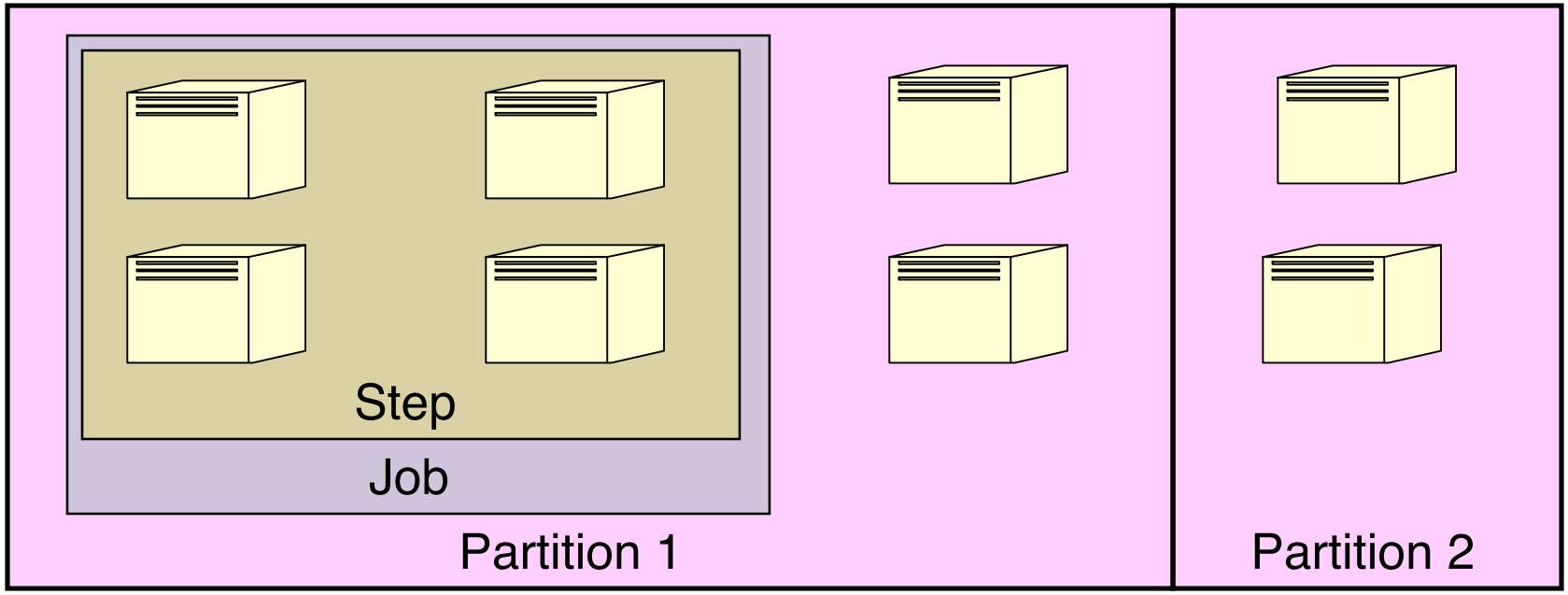
Plugins

- Dynamically linked objects loaded at run time per configuration file
- 30 different plugins of 10 different varieties
  - Authentication
    - Authd, Munge or none
  - Interconnect
    - Quadrics Elan3/4, IBM Federation, BlueGene or none (for Infiniband, Myrinet and Ethernet)
  - Scheduler
    - Maui, FIFO or backfill
  - Accounting, Logging, MPI type, etc.



SLURM Entities

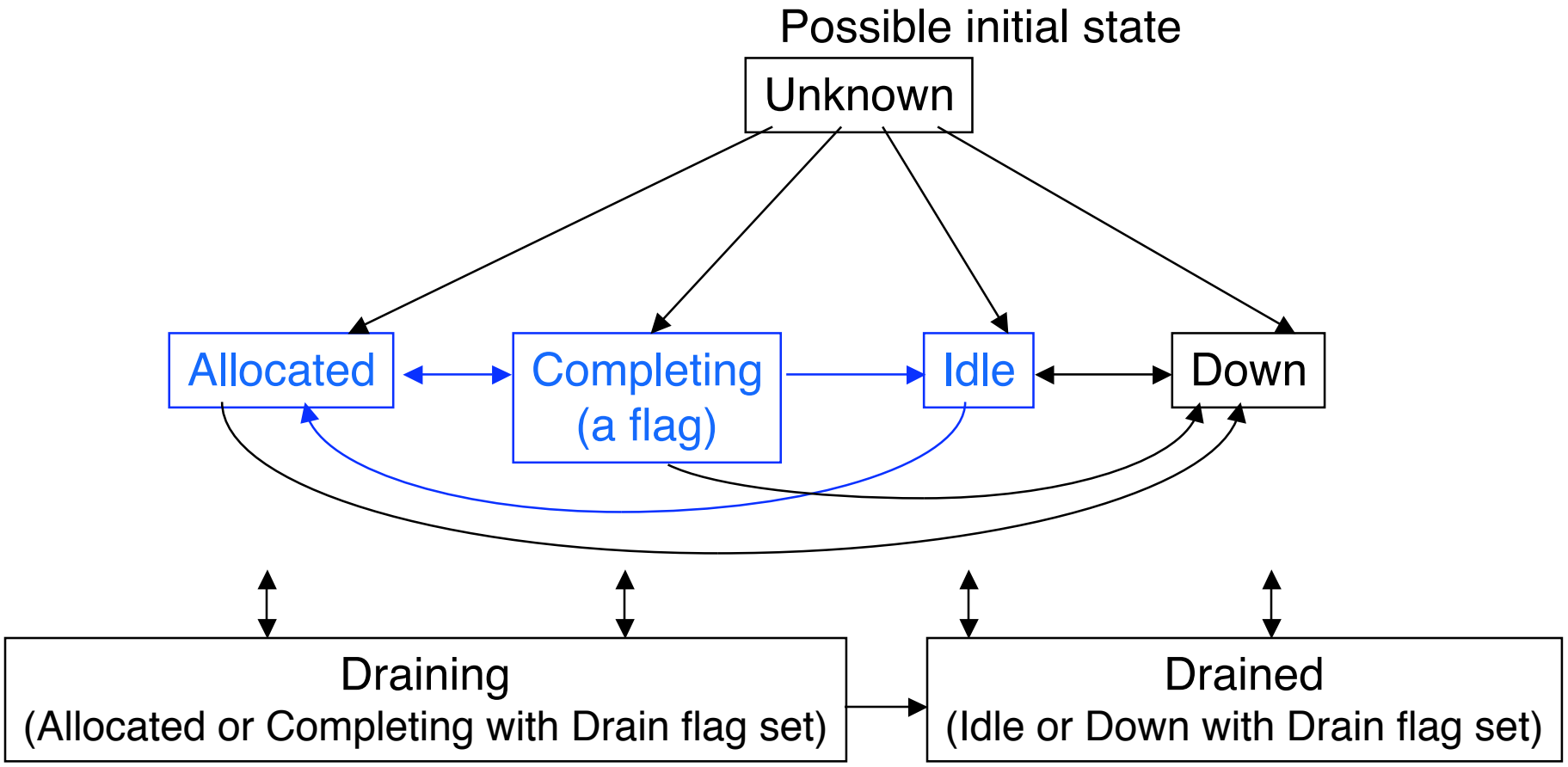
- Nodes: Individual computers
- Partitions: Job queues
- Jobs: Resource allocations
- Job steps: Set of (typically parallel) tasks



Nodes

- Configuration parameters
  - Processor count
  - Real memory size
  - Temporary disk space
  - Features (arbitrary string, e.g. OS version)
  - Scheduling weight (preference to allocate, use smaller number for less capable nodes to use them first unless users specifically request more capable nodes, e.g. more memory)
- Can allocate entire nodes to jobs or individual processors on each node
  - Nodes can also be shared (over-subscribed)

Node States



scontrol update NodeName=X state=[drain | resume] Reason=X

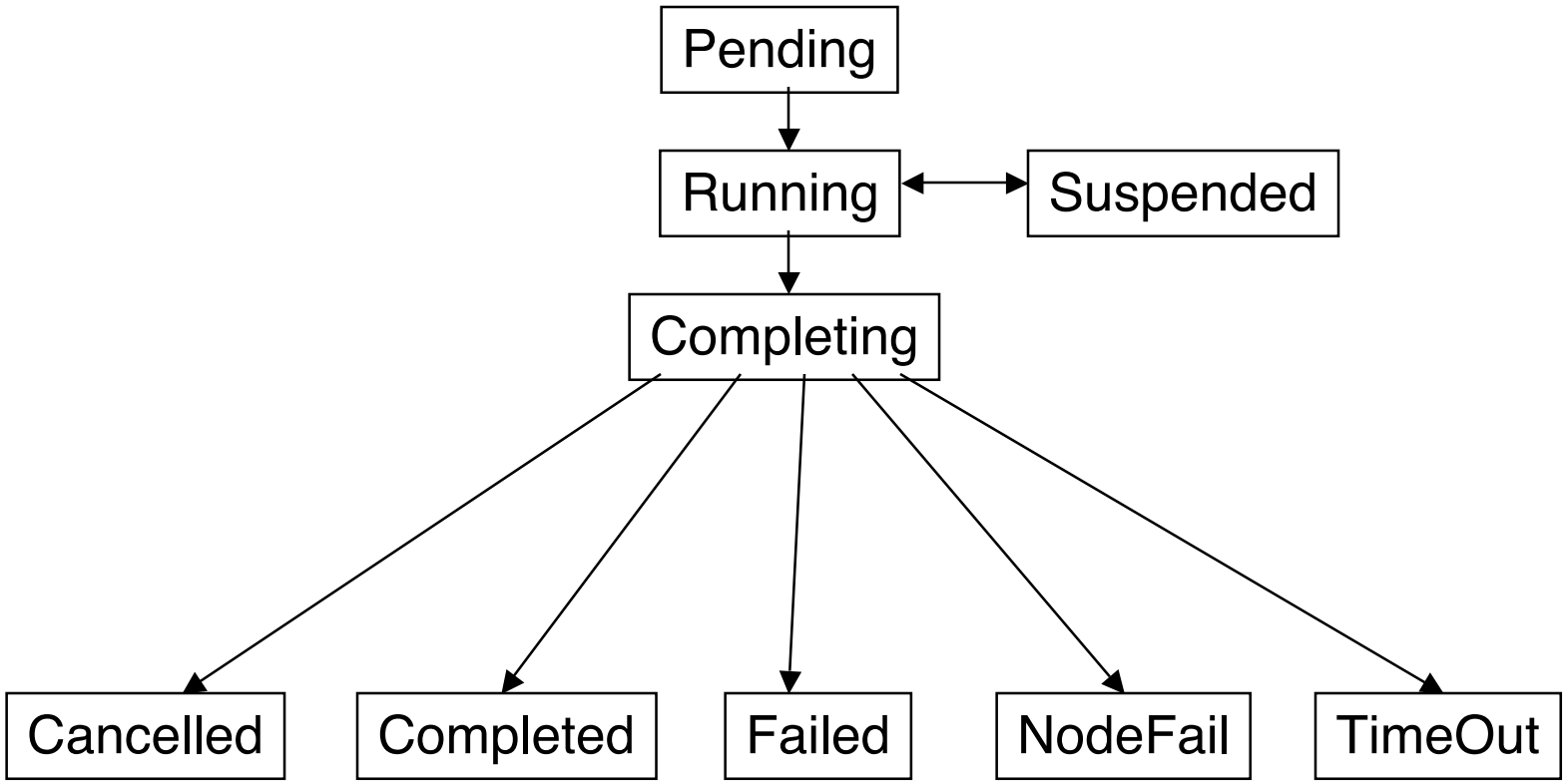
Partitions

- General purpose job queue
- Nodes can be in more than one partition (new in version 1.0)
- Configuration parameters
  - Default (where jobs run by default)
  - Unix groups allowed to use
  - Maximum time for job allocation
  - Minimum and maximum node count for job allocation
  - Shared (permit or force more than one job per node)
  - Hidden (by default, not seen by users lacking access)
  - RootOnly (only user root can create the allocation, prevents direct use by users, enforces batch system queuing)
  - State (UP or DOWN)

Jobs

- Resource allocation: specific processors and memory or entire nodes allocated to a user for some time period
- Can be interactive (executed in real-time) or batch (script queued for later execution)
- Many constraints available for user request
- Identified by ID number

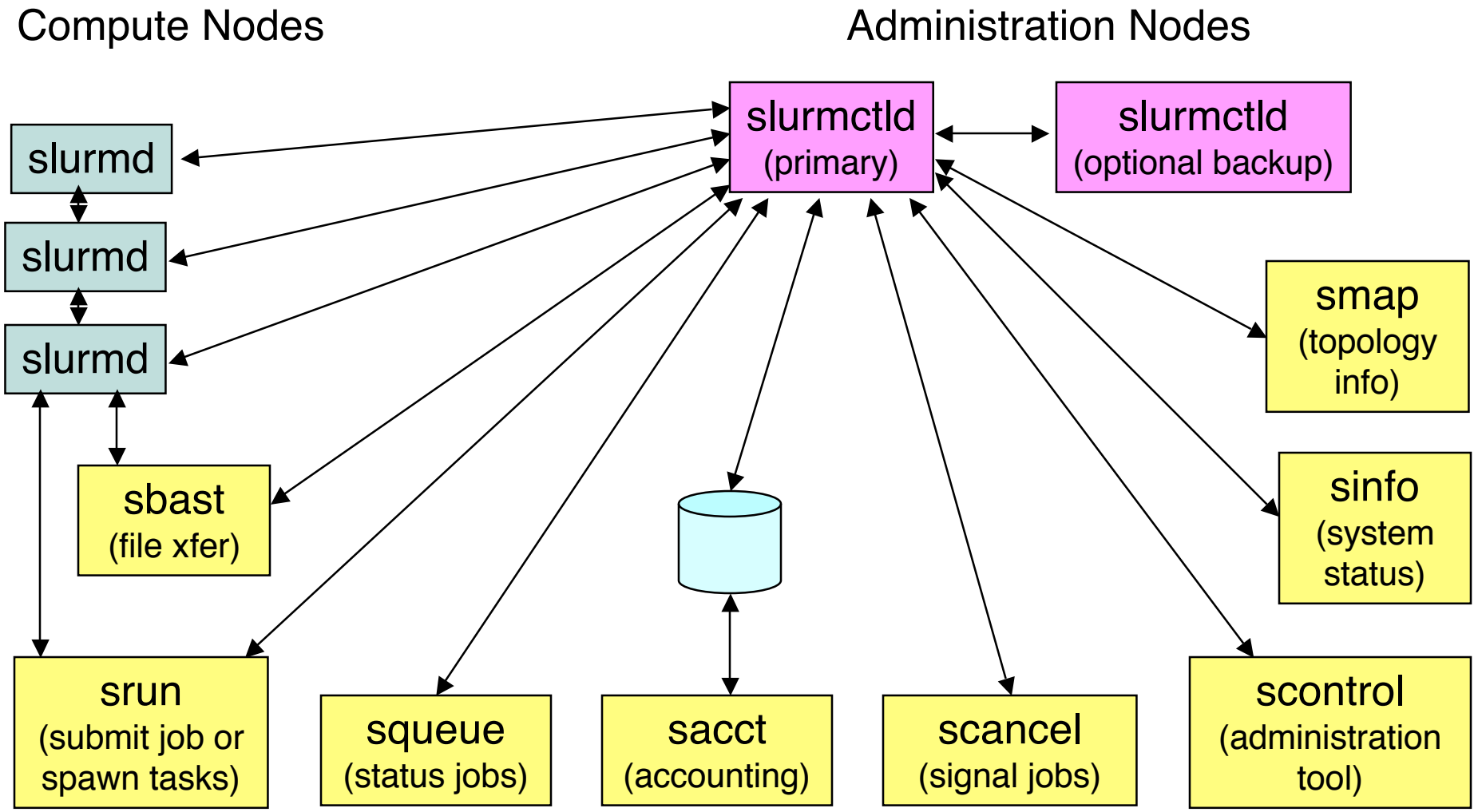
Job States


Pending
Running
Suspended
Completing
Cancelled
Completed
Failed
NodeFail
TimeOut

Job Steps

- A set of tasks launched at the same time and sharing a common communication mechanism (e.g. switch windows configured for the tasks)
- Allocated resources within the job's allocation
- Multiple job steps can executed concurrently or sequentially on unique or overlapping resources
- Identified by ID number: <jobid>.<stepid>

Daemons and Commands



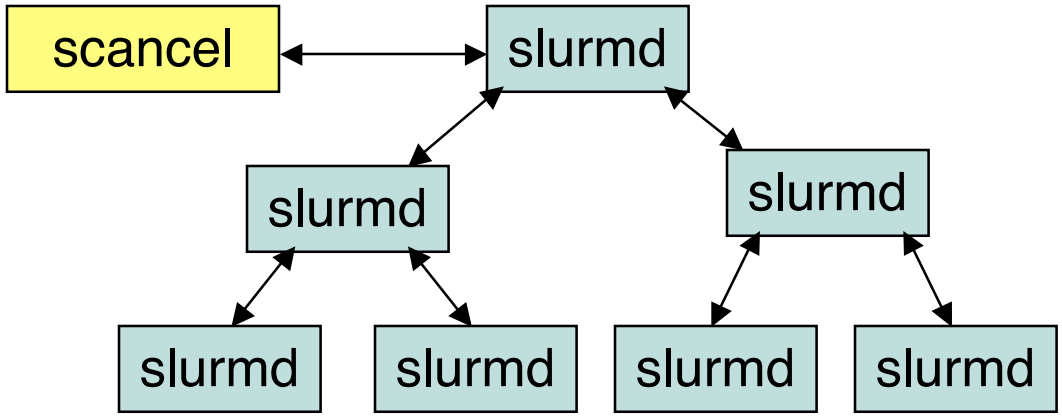
slurmctld
(SLURM Control Daemon)

- Orchestrates SLURM activities across the cluster
- Primary components
  - Node Manager: Monitors node state
  - Partition Manager: Groups nodes into partitions with various configuration parameters and allocates nodes to jobs
  - Job Manager: Accepts user job requests and places pending jobs into priority-ordered queue. Uses the partition manager to allocate resources to the jobs and then launch them.
- Optional backup slurmctld (must share file system for state information)

slurmd

(SLURM Compute Node Daemon)

- Monitors state of a single node
- Manages user jobs and job steps within that node
- Very light-weight
- Supports hierarchical communications with configurable fanout (new in version 1.1)



slurmctld and slurmd options

- -c Clear previous state, purges all job records for slurmctld
- -D Run in the foreground, logs are written to stdout
- -v Verbose error messages, repeat for extra verbose logging

```
> slurmctld -Dcvvvv   (a typical debug mode)
> slurmd -Dcvvvv      (a typical debug mode)
```

slurmstepd

(SLURM daemon to shepherd a job step)

- Spawned by slurmd on job step initiation
- Manages a job step and process its I/O
- One slurmstepd per job step
- Only persist while the job step is active

Commands – General Information

- Man pages are available for all commands
- “--help” option reports brief description of all options
- “--usage” option lists the options
- Can be run on any node in the cluster
- Any failure results in non-zero exit code
- SLURM APIs make new tools easy to develop
  - The APIs are all well documented

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”)
- Time formats are days-hours:minutes:seconds
- 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

scontrol

- Designed for system administrator use
  - Reports all available fields for all entities
  - Simple fixed output formats
  - Limited filtering
  - No sorting options
- Options can be abbreviated and are case insensitive
  - “scontrol sho conf” == “scontrol show configuration” == “scontrol SHOW CONFIG”
- Many fields can be modified interactively by user root
- Just enter “scontrol” command name to run in interactive mode

scontrol Display Example

- *scontrol show <entity> [id]*
  - Entity: configuration, job, node or partition
  - ID: a specific entity identifier, displays all by default

```
> scontrol show partition
PartitionName=debug TotalNodes=2 TotalCPUs=2048 RootOnly=NO
  Default=YES Shared=FORCE State=UP MaxTime=120 Hidden=NO
  MinNodes=1 MaxNodes=2 AllowGroups=ALL
  Nodes=linux[000-001] NodeIndicies=0,1,-1
```

Bitmap indicies, for mapping to node tables without string comparison.
Collection of comma separated min,max pairs with -1 terminator.
(e.g. linux[0-4,6-8,15] -> 0,4,6,8,15,15-1 assuming zero origin)

scontrol Update Example

- *The output generated with the "show" command can be used as input to the "update" command*
  - *Cut and paste relevant fields between commands*
  - *NOTE: Not all fields can be changed this way, some require changing the configuration file and reconfiguring SLURM*
- *When draining a node, you must specify a reason, the user id and time will be automatically appended*
- *Enclose node expressions with quotes*

```
> scontrol update PartitionName=debug MaxTime=60
> scontrol update NodeName="mcr[000-001]" State=drain \
Reason="Power supply failing"
```

scontrol More Examples


```
> scontrol reconfig    (re-read configuration file)
> scontrol shutdown    (shutdown SLURM daemons)

> scontrol suspend <jobid>
> scontrol resume <jobid>
```

scancel

- Cancel a running or pending job or job step
- Can send an arbitrary signal to all processes on all nodes associated with a job or job step
- Has filtering options (state, user, partition)
- Has interactive (verify) mode

```
> scancel 12.56   (cancel job step 12.56)
> scancel 13      (cancel job 13 and all of its steps)
> scancel --user=don --state=pending (cancel don's
                                      pending jobs)
```

sacct

- Reports accounting information for jobs and job steps
- Many filtering and output format options
- Uses job accounting file as input
- Accounting may be disabled (configuration option)

```
> sacct -u phil   (get accounting information for user "phil")
> sacct -p debug (get information for jobs in partition "debug")
```

squeue

- Reports status of jobs and/or job steps
- Almost complete control of filtering, sorting and output format is available

```
> squeue -u bob -t all (report jobs user "bob" in any state)
JOBID PARTITION  NAME USER ST TIME NODES NODELIST(REASON)
   56     debug a.out bob  CD 12:30     1 bgl000

> squeue -s -b debug   (report steps in partition "debug")
STEPID PARTITION  NAME USER TIME NODELIST
123.45     debug sleep  don 0:10 bgl001

> squeue -i60          (report job status every 60 seconds)
```

sinfo

- Reports status of nodes or partitions
  - Partition-oriented format is the default
- Almost complete control of filtering, sorting and output format is available

```
> sinfo --Node    (report status in node-oriented form)
NODELIST   NODES PARTITION STATE
linux[0-10]   11      batch alloc
linux[11-15]   5      debug  idle

> sinfo -p debug (report status of nodes in debug partition)
PARTITION AVAIL TIMELIMIT NODES NODELIST
debug        UP   1:00:00     5 linux[11-15]

> sinfo -i60      (reports status every 60 seconds)
```

smap

- Reports status of jobs, nodes and partitions using a graphical format
- Critical for BlueGene computer, shows 3-D node layout (e.g. job packing, like Tetris)
- Displays: job, partitions, bglblocks (BlueGene only)
  - Also establishes initial BlueGene configuration
- Command-line or graphical (curses) output formats

smap


```
   A A B B      ID JOBID PARTITION BG_BLOCK USER  ST TIME BP_LIST
  A A B B        A  1234     debug RMP0     donna  R 1:30 bgl[000x133]
 A A B B         B  1235     debug RMP1     danny  R 0:10 bgl[220x333]
A A B B          C  1240     debug RMP2     chris  R 9:10 bgl[200x311]
                 D  1241     debug RMP3       bob  R 0:49 bgl[202x213]
   A A B B
  A A B B
 A A B B
A A B B

   A A C C
  A A C C
 A A D D          Y
A A D D           |
                  |
   A A C C        0---X
  A A C C        /
 A A D D        /
A A D D        Z
```

sbcast (new in version 1.1)

- Copy a file to local disk on allocated nodes
  - Execute after a resource allocation has taken place
- Can be faster than using a single file system mounted on multiple nodes

```
> sbcast my_file /tmp/my_file

> sbcast --force my_data /tmp/my_data (overwrite old file)

> sbcast --preserve a.out /tmp/a.out   (preserve timestamps)
```

srun

- Used to create a job
  - Interactive mode (creates job allocation and runs a job step with a single command)
  - Allocate mode (allocates resources then spawns shell which can initiate one or more job steps using srun)
  - Batch mode (submit script for later execution)
- Can attach to previously allocated job
- Spawns job steps
- Dozens of options to control resource allocation
  - Count of nodes or processors
  - Specific nodes to use or avoid
  - Node features (processor count, memory size, etc.)

Different Executables by Task (new in version 1.1)

- 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 program, and arguments (“%t” mapped to task ID, “%o” mapped to offset within task ID range)

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

> srun -N5 --multi-prog master.conf
```

srun Interactive Examples


```
> srun -N2 --label hostname
0: linux0
1: linux1

> srun -n4 -N1 --oversubscribe --label hostname
0: linux0
1: linux0
2: linux0
3: linux0
```

srun Interactive Examples

- Standard Input, Output, and Error are forwarded to the controlling srun command

```
> srun -N2 bash
hostname
0: linux0
1: linux1
date
0: Thu Feb 16 13:33:35 PST 2006
1: Thu Feb 16 13:33:35 PST 2006
exit  (terminate "bash" command,
       which ends the job)
```

srun Allocate Example


```
> srun -N2 --allocate  (NOTE: the resource allocation
$ echo $SLURM_JOBID     is performed, no tasks spawned)
1234

$ squeue -j $SLURM_JOBID
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
 1234     debug        moe  R 0:02     2 linux[0-1]

$ srun hostname  (run one or more job steps)
linux0
linux1

$ exit (terminate shell and release the allocation)
```

srun Batch Example

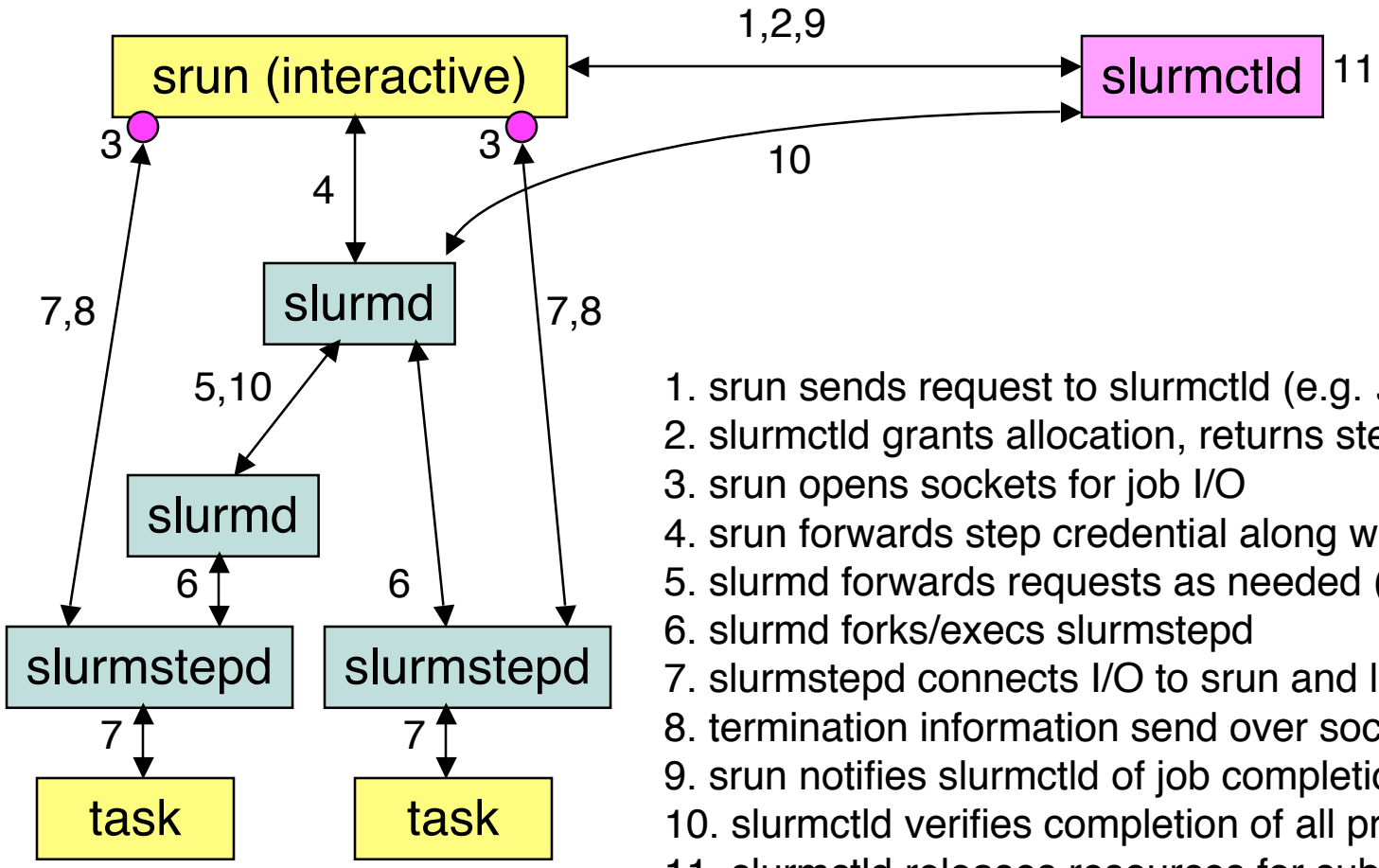

```
> echo tst.sh
#!/bin/sh
#SLURM -N4
srun hostname

> srun -b tst.sh
srun: jobid 4567 submitted

… later …

> cat slurm.4567.out
linux0
linux1
linux2
linux3
```

MPI Support

- Many different versions of MPI are supported: MPICH2, OpenMPI, HP-MPI, LAM/MPI, Quadrics MPI, ChaMPIon, BlueGene MPI
- Srun is used to directly launch tasks for Quadrics MPI and MPICH2: *srun –N16 a.out*
- Others use mpirun, typically within a previously created SLURM job allocation

```
> srun -N2 --allocate
$ mpirun a.out
$ exit (release the allocation)
```

Job Execution Sequence



1. srun sends request to slurmctld (e.g. *srun –N2 hostname*)
2. slurmctld grants allocation, returns step credential
3. srun opens sockets for job I/O
4. srun forwards step credential along with task info to slurmd
5. slurmd forwards requests as needed (with fanout)
6. slurmd forks/execs slurmstepd
7. slurmstepd connects I/O to srun and launches tasks
8. termination information send over sockets to srun
9. srun notifies slurmctld of job completion
10. slurmctld verifies completion of all processes
11. slurmctld releases resources for subsequent jobs

Build and Install, RPMs

- Download a tar-ball
  - ftp://ftp.llnl.gov/pub/linux/slurm or
  - http://www.sourceforge.net (also has RPMs)
- Build and install the relevant RPMs
  - rpmbuild --ta slurm-1.1.0.tar.bz2
  - rpm --install <the rpm files>
- NOTE: Some RPMs are infrastructure specific:
  - slurm-auth-authd*rpm authd authentication plugin
  - slurm-auth-munge*rpm munge authentication plugin
  - slurm-auth-none*rpm “none” authentication plugin
  - slurm-bluegene*rpm BlueGene specific plugins and tools
  - slurm-switch-elan*rpm Quadrics Elan switch plugin
  - slurm-switch-federation*rpm IBM Federation switch plugin

Build and Install without RPMs

- ./configure <options>
  - --enable-debug — additional debugging
  - --prefix=<dir> — installation location
  - --sysconfdir=<dir> — configuration file location
- make
- make install

Configuration

- Most configuration parameters have usable defaults
- You will at least need to identify the nodes in your cluster and their grouping into a partition

- Web-based tool included, good for simple setups
  - doc/html/configurator.html
- Sample configuration file with extensive in-line comments included
  - etc/slurm.conf.example
- For more information: *man slurm.conf*

Configuration Options

- Specify some authentication mechanism
  - AuthType=auth/munge is recommended
  - AuthType=auth/none is OK for our testing
- Node naming
  - NodeName: Name SLURM uses for the node
  - NodeAddr (optional): Name or IP address for communications
  - NodeHostname (optional): what “hostname -s” returns on the node
  - “localhost” works for a stand-alone system
- SelectType controls node selection plugin
  - SelectType=select/cons_res allocates individual processors
  - SelectType=select/linear allocates entire nodes to jobs
  - SelectType=select/bluegene for BlueGene computer only

Configuration Options

- The actual resources that each node registers with will be used for scheduling if FastSchedule=0
- Otherwise the resources configured in slurm.conf will be used as a basis for scheduling (faster)

- You must also explicitly define the partitions and their nodes

Scheduling Options

- SchedType=sched/builtin First-In First-Out
- SchedType=sched/backfill Conservative backfill
- SchedType=sched/maui External Maui Scheduler

- Gang scheduling: Time slice resources for parallel jobs
  - Use scontrol suspend/resume (control via script)
  - Can dramatically improve system utilization and performance
  - Example:
    - scontrol update PartitionName=batch state=down
    - scontrol suspend <jobid> (for all running jobs in partition “batch”)
    - scontrol update PartitionName=full state=up
    - scontrol resume <jobid> (for any suspended job in partition “full”)

Sample Configuration
(excerpt)


```
# Sample SLURM configuration (excerpt)
ControlMachine=linux0
BackupController=linux1
#
AuthType=“auth/authd”
PluginDir=/usr/lib/slurm
SlurmctldPort=7002
SlurmdPort=7003
SlurmUser=slurm
#
NodeName=DEFAULT Procs=2 TmpDisk=64000
NodeName=linux[2-1000]      RealMemory=16000 Weight=16
NodeName=linux[1001-1016] RealMemory=32000 Weight=32
#
PartitionName=debug Nodes=linux[2-33]      MaxTime=30
PartitonName=batch   Nodes=linux[34-1016] MaxTime=Infinite
```

Pluggable Authentication Module (PAM)

- Can be used to prevent users from logging into node that isn't allocated to them
- Distinct package (not in the SLURM tar-ball or RPM), but can be downloaded from the same FTP server
- Can also be used by SLURM to establish node-specific limits for a user's tasks when spawned (new in version 1.1)

Test Suite

- SLURM has an extensive test suite: about 160 tests that execute roughly 1,000 jobs and 10,000 job steps

- First build, install, configure and initiate SLURM
- Change directory to “testsuite/expect”
- Copy “globals.example” to “globals” and modify pathnames as needed (likely the variable “slurm_dir”)
- Run individual tests as desired (see README for their descriptions) or run the full suite by executing the script “regression”

Demonstration Build and Install (Step 0)

- The SLURM CD contains
  - SLURM tar-ball
  - Sample configuration files
    - Single host
    - Emulated cluster
    - Emulated BlueGene system
  - SLURM web pages
  - SLURM PAM tar-ball
  - Tutorial

Demonstration Build and Install (Step 1)

- *cd /tmp*
- *mkdir slurm* (used as install directory)
- mkdir slurm/etc (used for configuration info)
- *cp <cd_location>/slurm-1.1.0.tar.bz2 .*
- *bunzip2 slurm-1.1.0.tar.bz2*
- *tar -xf slurm-1.1.0.tar*
- *cd slurm-1.1.0*

Demonstration Build and Install (Step 2)

- *./configure --enable-debug --prefix=/tmp/slurm \ --sysconfdir=/tmp/slurm/etc*

- Trick to emulate a cluster:
  - *echo "#define HAVE_FRONT_END 1" >>config.h*
- Trick to emulate BlueGene:
  - *echo "#define HAVE_FRONT_END 1" >>config.h*
  - *echo "#define HAVE_BG 1" >>config.h*

- *make*

- *make install*

Demonstration Build and Install (Step 3)

- *cp <cd_location>/configs/* /tmp/slurm/etc*
  - Substitute slurm.conf.bluegene or slurm.conf.cluster for slurm.conf as needed
  - Set SlurmUser to your user name
- *cd /tmp/slurm*
- *xterm &*
- *xterm &*
- *xterm &*

Demonstration Build and Install (Step 4)

- New window 1:
  - sbin/slurmctld -Dcvv
- New window 2:
  - *sbin/slurmd -Dcvv*
- *New window 2:*
  - *cd bin*
  - *./sinfo*
  - *./srun hostname*
  - *./srun -N1 -n10 -O hostname*
  - *./srun -N1 -A*
    - *./squeue*

Demonstration Build and Install (Extras)

- Try running configurator.html (don't need Linux machine)
- Try running test suite
  - *cd /tmp/slurm-1.1.0/testsuite/expect*
  - *cp globals.example globals*
  - Edit globals: set slurm_dir to "/tmp/slurm"
  - *./test1.1*
  - *./regression >qa.out*

Blue Gene Support

- Additional configuration file, *bluegene.conf*, controls configuration of bglblocks
  - Build using smap tool
  - bglblocks may overlap or be created as needed (new in v1.1)
- SlurmProlog and SlurmEpilog must use included slurm_prolog and slurm_epilog to synchronize job with bglblock state
- SLURM internally treats each midplane as a node
  - Names specify end points in rectangular prism: bgl[000x133]
- Major change in version 1.1:
  - User tools all use c-node count
  - Bluegene plugin maps between the two systems

IBM SP / AIX Support

- SLURM builds on AIX and has a plugin for the Federation switch
- Job step launch is performed through POE for compatibility with IBM tools (it's slower than SLURM job step launch, but much faster than LoadLeveler)
- SLURM has a library that emulates LoadLeveler for POE's use. Unfortunately that is based upon IBM confidential information and is not available to others at this time

Large Cluster Support (>1k nodes)

- Virtually all SLURM components have been validated to 16k nodes
- Allocate whole nodes to jobs rather than individual processors: *SelectType=select/linear*
- Set slurmd ping interval large: *SlurmdTimeout=120* or disable completely: *SlurmdTimeout=0*
- Avoid enabling job accounting: *JobAcctType=jobacct/none*
  - If needed, configure long sampling interval: *Frequency*=120
  - Job completion logs may also prove useful

Mailing lists

- For communications with developers (sometimes high-volume): *slurm-dev@lists.llnl.gov*
- For announcements about new releases (low-volume): *slurm-announce@lists.llnl.gov*
- To subscribe send e-mail to majordomo@lists.llnl.gov with the body of the message containing the word "subscribe" followed by the list name and your e-mail address (if not the sender). For example: *subscribe slurm-dev bob@yahoo.com*

Development

- Contact slurm-dev@lists.llnl.gov to coordinate efforts and avoid “re-inventing the wheel”
- If practical, perform your development work in the form of a plugin so that it can be well isolated. There is documentation only for all of the plugin interfaces
- If possible, use the “C” programming language and follow Linux Kernel coding style for consistency (see http://www.llnl.gov/linux/slurm/coding_style.pdf) , which is basically Kernigan and Richie coding style with 8-character indentations

Testimonials

- *"SLURM is the coolest invention since Unix."* – Dennis Gurgul, Partners Health Care
- *"[SLURM] reduces job launch times from tens of minutes with LoadLeveler to a few seconds using SLURM. This effectively provides us with millions of dollars worth of additional compute resources without additional cost."* - Dona Crawford, LLNL
- "I would rank SLURM as the best of the three open source batching systems available, by a rather large margin." Bryan O'Sullivan, PathScale
- *"SLURM is a great product that I'd recommend to anyone setting up a cluster…"* - Josh Lothian, Oak Ridge National Laboratory

For More Information

- Information: http://www.llnl.gov/linux/slurm
- Downloads: ftp://ftp.llnl.gov/pub/linux/slurm
- Email: jette1@llnl.gov