High Scalability Resource Management with SLURM
Supercomputing 2008
November 2008

Morris Jette (jette1@llnl.gov)

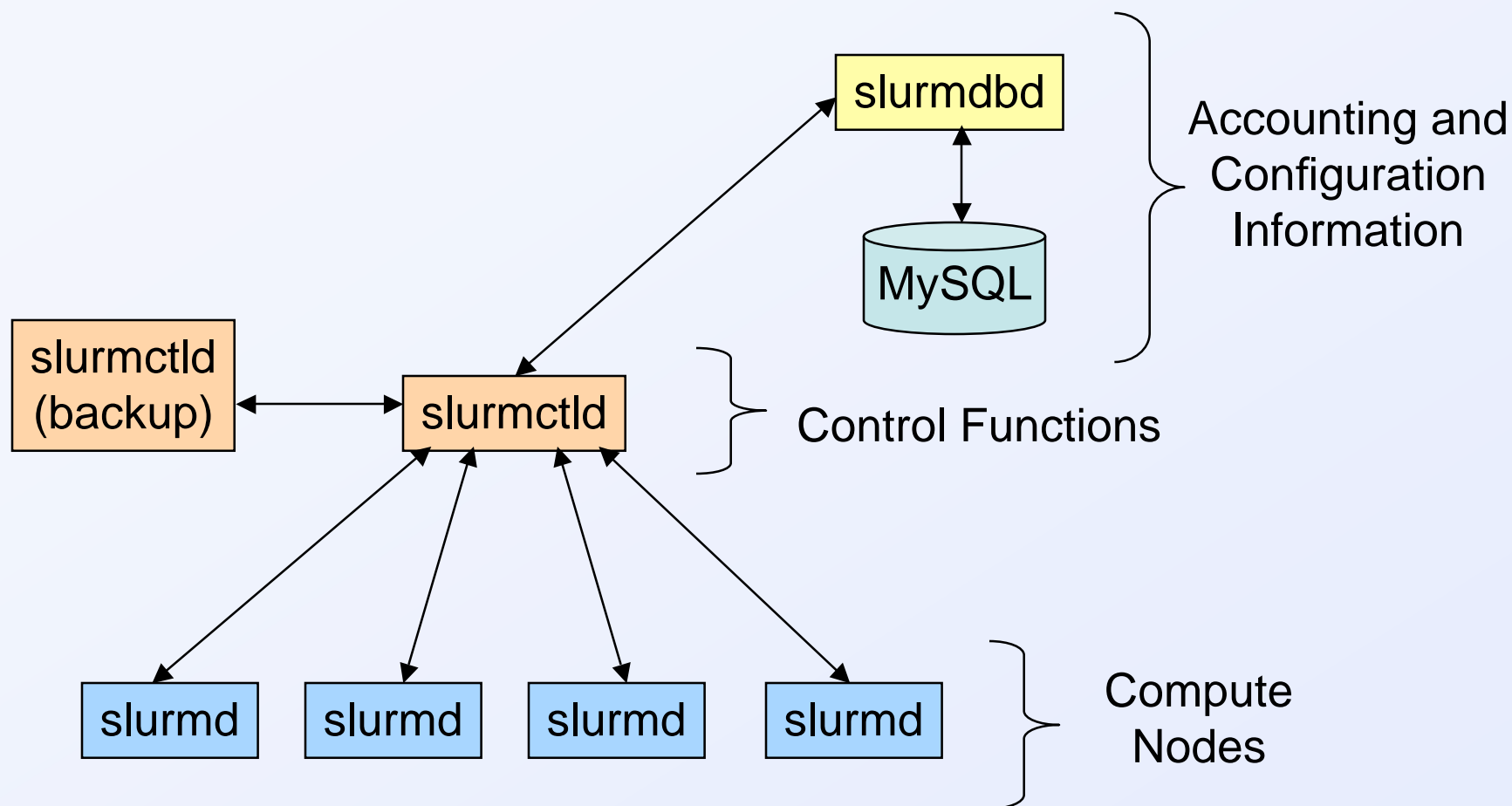**S&T Principal Directorate - Computation Directorate**

What is SLURM

- **S**imple **L**inux **U**tility for **R**esource **M**anagement

- Performs resource management within a single cluster
- Typically used with an external scheduler (e.g. Moab or LSF)

- Arbitrates requests by managing queues of pending work
- Allocates access to computer nodes and their interconnect
- Launches parallel jobs and manages them (I/O, signals, time limits, etc.)

- Developed by LLNL with help from HP, Bull, Linux NetworX, and others

Design objectives

- High scalability
    - Thousands of nodes

- Reliable
    - Avoid single point of failure

- Simple to administer

- Open source (GPL)

- Extensible
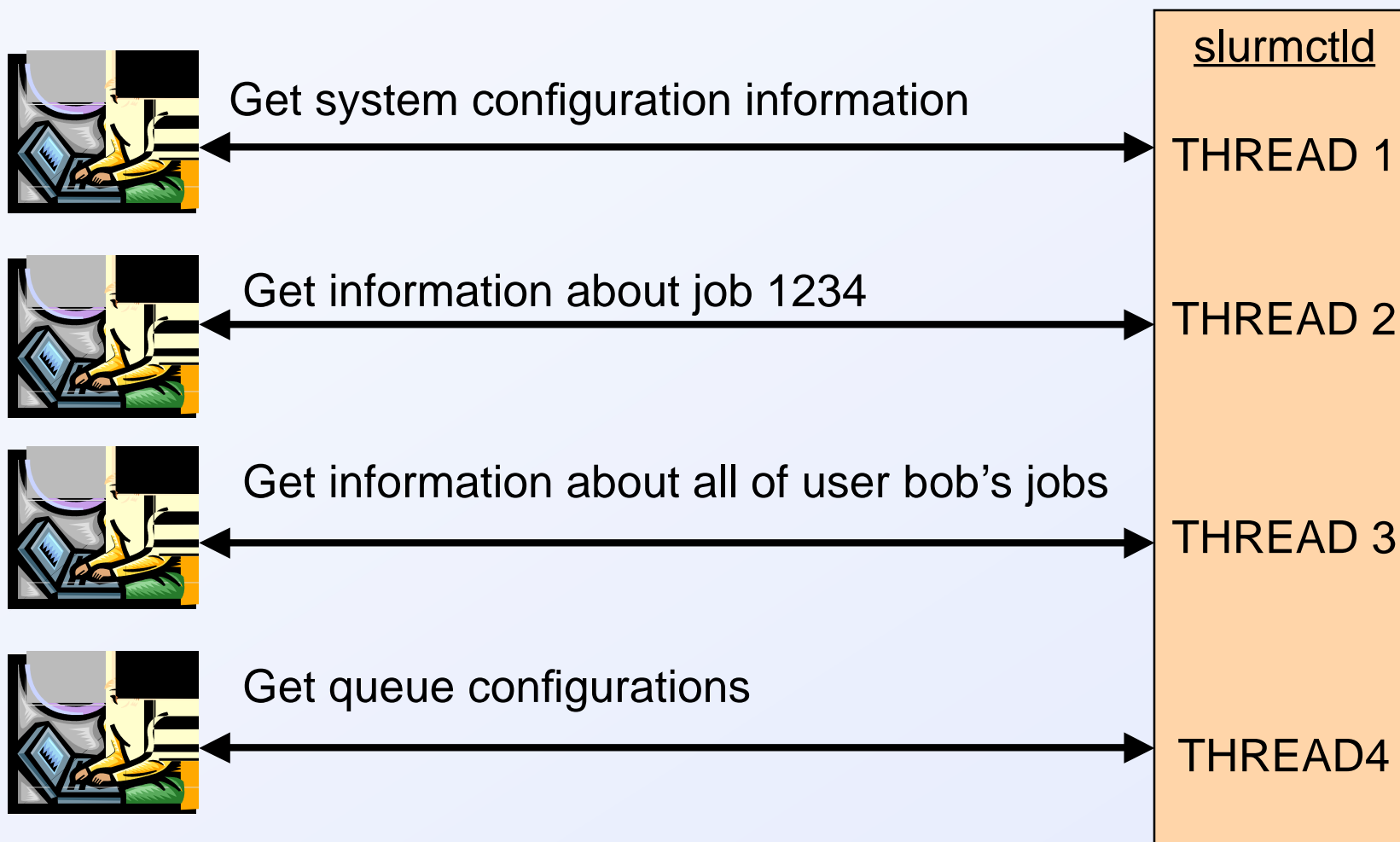   - Very flexible plugin mechanism

SLURM architecture overview

slurmdbd

MySQL

Accounting and Configuration Information

slurmctld (backup)

slurmctld

Control Functions

slurmd   slurmd   slurmd   slurmd

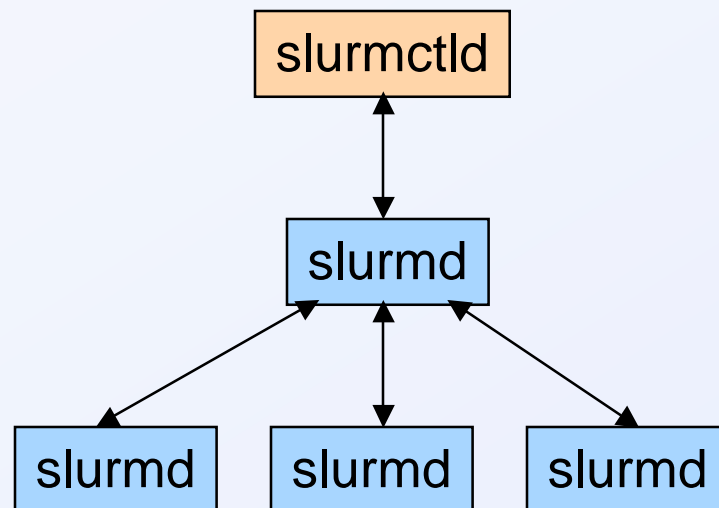Compute Nodes

How we achieve high scalability

- High parallelism

  - Dozens of active threads are common in the daemons

  - Independent read and write locks on various data structures

  - Offload as much work as possible from slurmctld (SLURM Control Daemon)

Example of parallel slurmctld operations

Get system configuration information

Get information about job 1234

Get information about all of user bob's jobs

Get queue configurations

slurmctld

THREAD 1

THREAD 2

THREAD 3

THREAD4

Hierarchical communication

- Slurmd daemons (one per compute node) have hierarchical communications with fault-tolerance and configurable fanout

```
         ┌──────────┐
         │ slurmctld │
         └──────────┘
              ↕
          ┌────────┐
          │ slurmd │
          └────────┘
          ↙   ↕   ↘
   ┌────────┐ ┌────────┐ ┌────────┐
   │ slurmd │ │ slurmd │ │ slurmd │
   └────────┘ └────────┘ └────────┘
```

Most of the communications overhead is pushed down to slurmd daemons.

Synchronizes system noise to minimize impact upon applications.

Slurmd provides fault-tolerance and combines results into <u>one</u> response message, dramatically reducing slurmctld's overhead.

Non-killable processes

- Non-killable processes (typically hung on I/O to global file system) are not uncommon so we try to minimize their impact

- SLURM supports configurable timeout and program to execute when non-killable processes are found so that system administrators can respond quickly to problems

- SLURM will release a resource for reuse by another job once all processes associated with the previous job on that node complete. There is no need to wait for all processes on all nodes to complete

Highly efficient algorithms

- Bitmap operations used for much of the scheduling work

- RPCs use simple binary information rather than XML (which is more flexible, but slower)

Nodes in selected partition AND   111111111111110000
Nodes with selected features AND   111111100000000000
Nodes with available resources   001110000011100001
      Select from these nodes-->   001110000000000000

Hostlist expressions used in configuration file

- Configuration file size is relatively independent of cluster size

```
# slurm.conf
# plugins, timers, log files, etc.
#
NodeName=tux[0-1023] SocketsPerNode=4 CoresPerSocket=4
#
PartitionName=debug Nodes=tux[0-15] MaxTime=30:00
PartitionName=batch Nodes=tux[16-1023] MaxTime=1-00:00:00
```

Hostlist expressions used in most commands

```
$ squeue
PARTITION AVAIL  TIMELIMIT NODES STATE NODELIST
debug           up      30:00        16   idle   tux[0-15]
batch           up  1-00:00:00       32   alloc  tux[16-47]
batch           up  1-00:00:00      976   idle   tux[48-1023]

$ sinfo
JOBID PARTITION NAME USER ST TIME NODES NODELIST
 1234  batch         a.out    bob    R  10:14        16  tux[16-31]
 1238  batch         my.sh  alice    R   8:57        16  tux[32-47]
```

Results

- SLURM is running on about 35% of the Top500 systems

- Total execution time (resource allocation, launch, I/O processing, resource deallocation)

  - 32 nodes   0.1 second

  - 256 nodes 1.0 seconds   } Same system

  - 1k nodes   3.7 seconds

  - 2k nodes 19.5 seconds   } Two different systems

  - 4k nodes 56.6 seconds   with different configurations

- Virtual machine with 64k nodes has been emulated

Special note on task launch

- Some vendors supply proprietary task launch mechanisms (e.g. IBM BlueGene *mpirun*)

- For compatibility with existing vendor tools and/or infrastructure (rather than for performance reasons), the vendor supplied task launch mechanism can be used with SLURM performing the resource management
  - Current model on IBM BlueGene systems

For more information about SLURM

- Information: https://computing.llnl.gov/linux/slurm/

- Downloads: http://sourceforge.net/projects/slurm/

- Email: jette1@llnl.gov

- SLURM BOF in Hilton (directly across the street)
  - Thursday 20 November 3PM to 5PM
  - Room: Salon D

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accurTm(e) Tj 67 0 0