Basic Configuration and Usage

Rod Schultz
email: rod.schultz@bull.com

Architect of an Open World

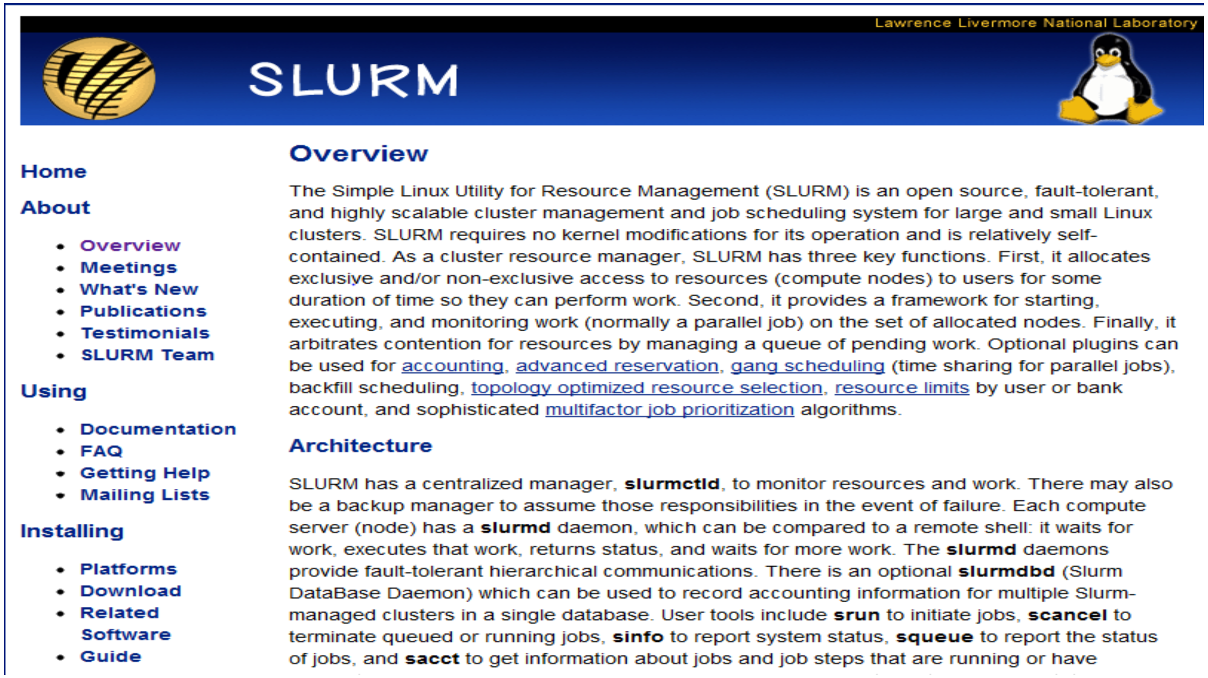
Outline

- Introduction
- Commands & Running Jobs
- Configuration
- Scheduling
- Accounting
- Advanced Topics

Simple Linux Utility for Resource Management

Documentation

- SchedMD.com, (computing.llnl.gov/linux/slurm/)
- <install_loc>/share/doc/<release>/overview.html (..man_index.html)



Home

About

- Overview
- Meetings
- What's New
- Publications
- Testimonials
- SLURM Team

Using

- Documentation
- FAQ
- Getting Help
- Mailing Lists

Installing

- Platforms
- Download
- Related Software
- Guide

Overview

The Simple Linux Utility for Resource Management (SLURM) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. SLURM requires no kernel modifications for its operation and is relatively self-contained. As a cluster resource manager, SLURM has three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work. Optional plugins can be used for accounting, advanced reservation, gang scheduling (time sharing for parallel jobs), backfill scheduling, topology optimized resource selection, resource limits by user or bank account, and sophisticated multifactor job prioritization algorithms.

Architecture

SLURM has a centralized manager, **slurmctld**, to monitor resources and work. There may also be a backup manager to assume those responsibilities in the event of failure. Each compute server (node) has a **slurmd** daemon, which can be compared to a remote shell: it waits for work, executes that work, returns status, and waits for more work. The **slurmd** daemons provide fault-tolerant hierarchical communications. There is an optional **slurmdbd** (Slurm DataBase Daemon) which can be used to record accounting information for multiple Slurm-managed clusters in a single database. User tools include **srun** to initiate jobs, **scancel** to terminate queued or running jobs, **sinfo** to report system status, **squeue** to report the status of jobs, and **sacct** to get information about jobs and job steps that are running or have

SLURM Principles

Architecture Design:

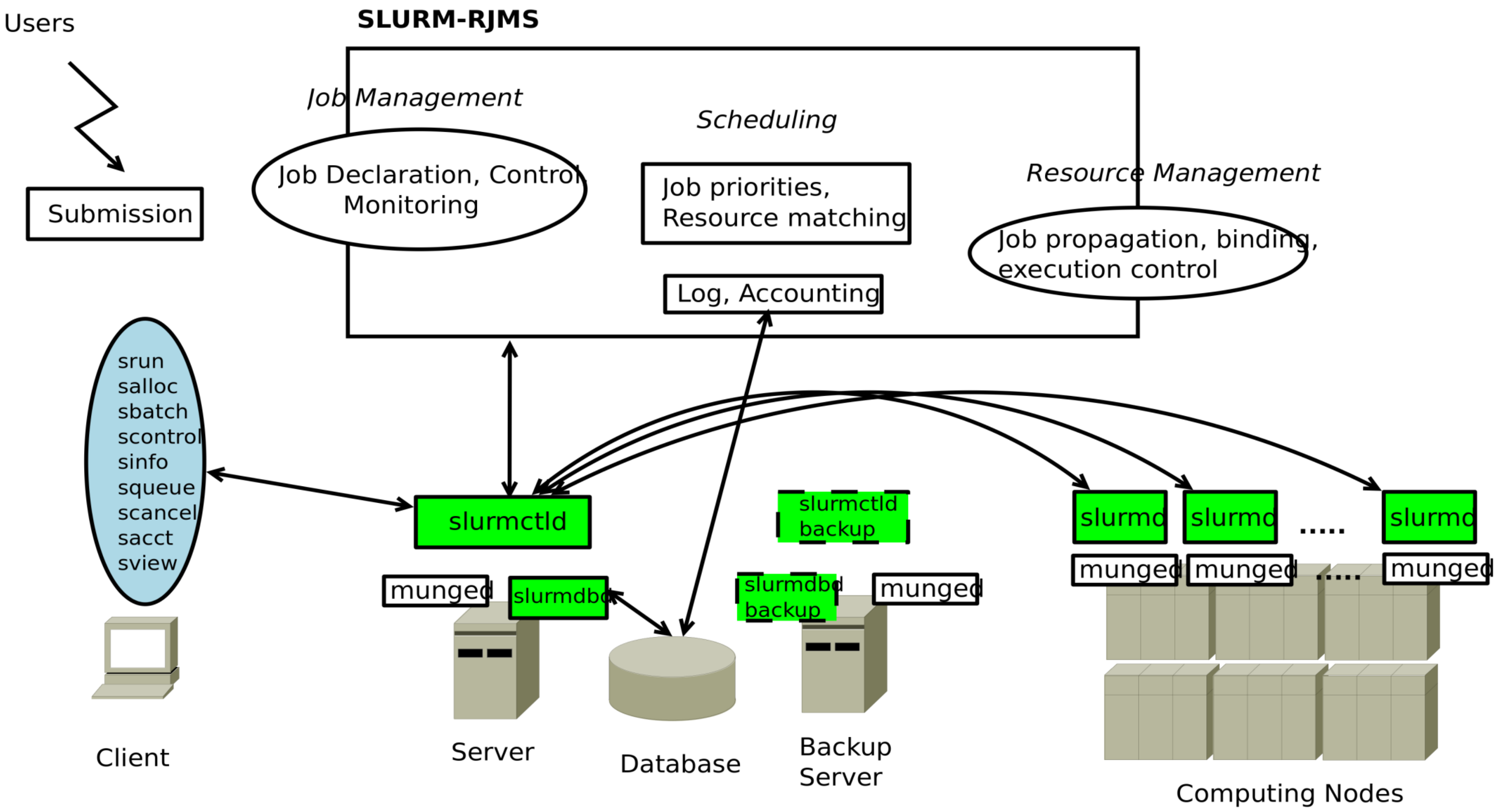
- One central controller daemon (**slurmctld)** on a management node
- A daemon on each computing node (**slurmd)**
- One central daemon for the accounting database (**slurmdbd)**
- SLURM may be aware of network topology and use it in node selection.
- IO nodes are not managed by SLURM

SLURM Principles ...

Principal Concepts:

- A general purpose **plug-in mechanism** (provides different behavior for features such as scheduling policies, process tracking, etc)
- **Partitions** represent group of nodes with specific characteristics (similar resources, priority, job limits, access controls, etc)
- One **queue** of pending work
- **Job steps** which are sets of tasks within a job

SLURM Architecture



Basic CPU Management Steps

SLURM uses four basic steps to manage CPU resources for a job/step:

1) Selection of Nodes

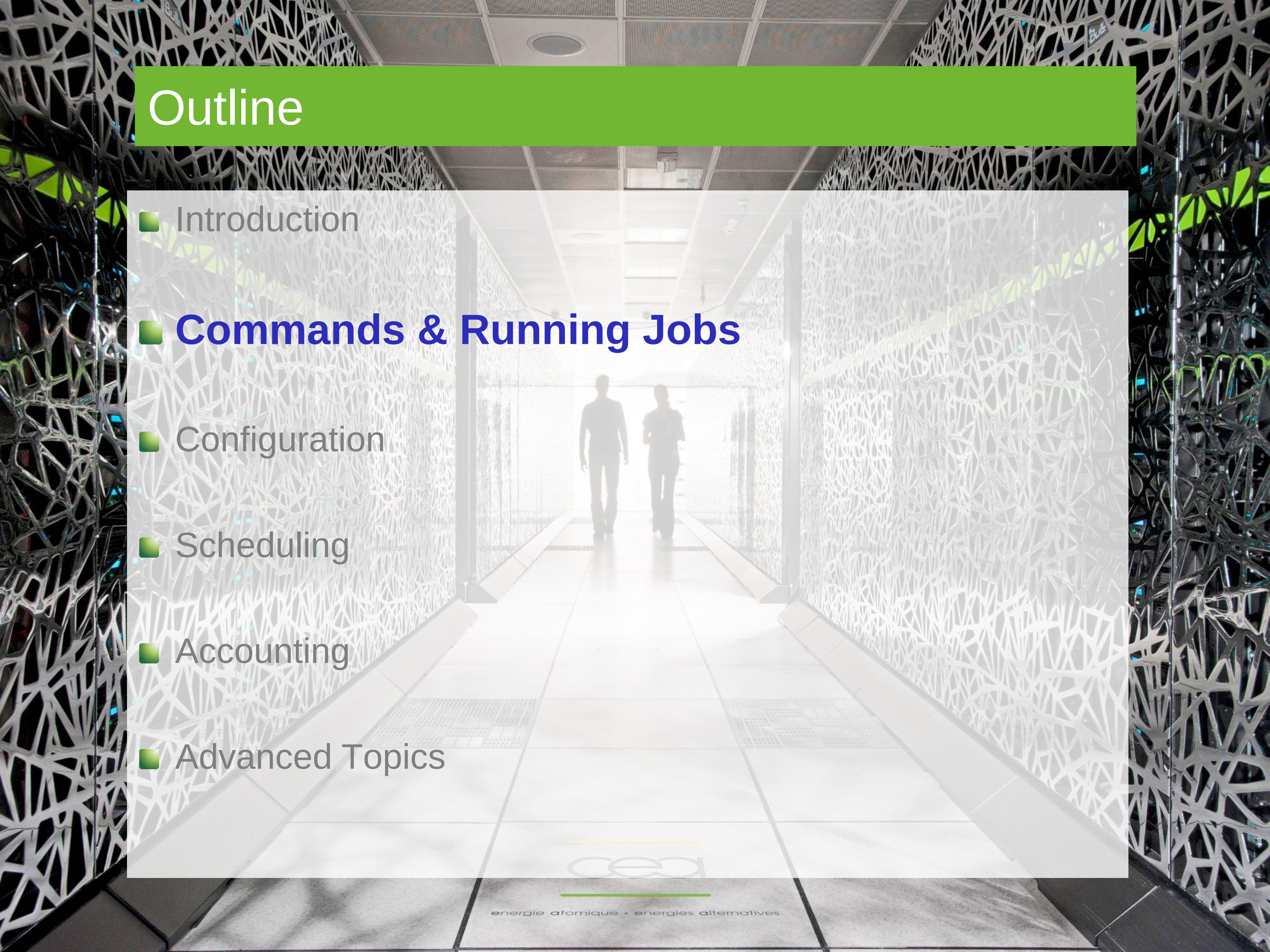
2) Allocation of CPUs from Selected Nodes

3) Distribution of Tasks to Selected Nodes

4) Optional Distribution and **Binding** of Tasks to Allocated CPUs within a Node (Task Affinity)

- SLURM provides a rich set of configuration and command line options to control each step
- Many options influence more than one step
- Interactions between options can be complex
- Users are constrained by Administrator's configuration choices

Outline



- Introduction
- **Commands & Running Jobs**
- Configuration
- Scheduling
- Accounting
- Advanced Topics

User & Admin Commands

sinfo display characteristics of partitions

squeue display jobs and their state

scancel cancel a job or set of jobs.

scontrol display and changes characteristics of jobs, nodes, partitions.

sstat show status of running jobs.

sview graphical view of cluster. Display and change characteristics of jobs, nodes, partitions.

Examples of info commands

```
> sinfo

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
all*         up   infinite      4   idle trek[0-3]
P2           up   infinite      4   idle trek[0-3]
P3           up   infinite      4   idle trek[0-3]
```

```
> scontrol show node trek0
NodeName=trek3 Arch=x86_64 CoresPerSocket=4
   CPUAlloc=0 CPUErr=0 CPUTot=16 Features=HyperThread
   Gres=(null)
   NodeAddr=trek0 NodeHostName=trek0
   OS=Linux RealMemory=1 Sockets=2
   State=IDLE ThreadsPerCore=2 TmpDisk=0 Weight=1
   BootTime=2011-06-30T11:04:22 SlurmdStartTime=2011-07-12T06:23:43
   Reason=(null)
```

User Commands

srun allocate resources (number of nodes, tasks, partition, constraints, etc.) launch a job that will execute on each allocated cpu.

salloc allocate resources (nodes, tasks, partition, etc.), either run a command or start a shell. Request launch srun from shell. (interactive commands within one allocation)

sbatch allocate resources (nodes, tasks, partition, etc.) Launch a script containing sruns for series of steps.

- Similar set of command line options.
- Request number of nodes, tasks, cpus, constraints, user info, dependencies, and lots more.

Sample srun

```
>srun -l –p P2 –N2 –tasks-per-node=2 –exclusive hostname
```

-l	prepend task number to output (debug)
-p P2	use Partition P2
-N2	use 2 nodes
--tasks-per-node	launch 2 tasks on each node
--exclusive	do not share the nodes
hostname	command to run.

```
0: trek0
1: trek0
2: trek1
3: trek1
```

Admin Commands

sacctmgr setup accounts, specify limitations on users and groups. (more on this later)

sreport display information from accounting database on jobs, users, clusters.

sview graphical view of cluster. Display and change characteristics of jobs, nodes, partitions. (admin has more privilege.)

srun & info command example

```
>srun -p P2 -N2 -n4 sleep 120 &
>srun -p P3 sleep 120 &
>srun -w trek0 sleep 120 &
>srun sleep 1
srun: job 108 queued and waiting for resources
```

```
>sinfo

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
all*         up   infinite      3  alloc trek[0-2]
all*         up   infinite      1   idle trek3
P2           up   infinite      3  alloc trek[0-2]
P2           up   infinite      1   idle trek3
P3           up   infinite      3  alloc trek[0-2]
P3           up   infinite      1   idle trek3
```

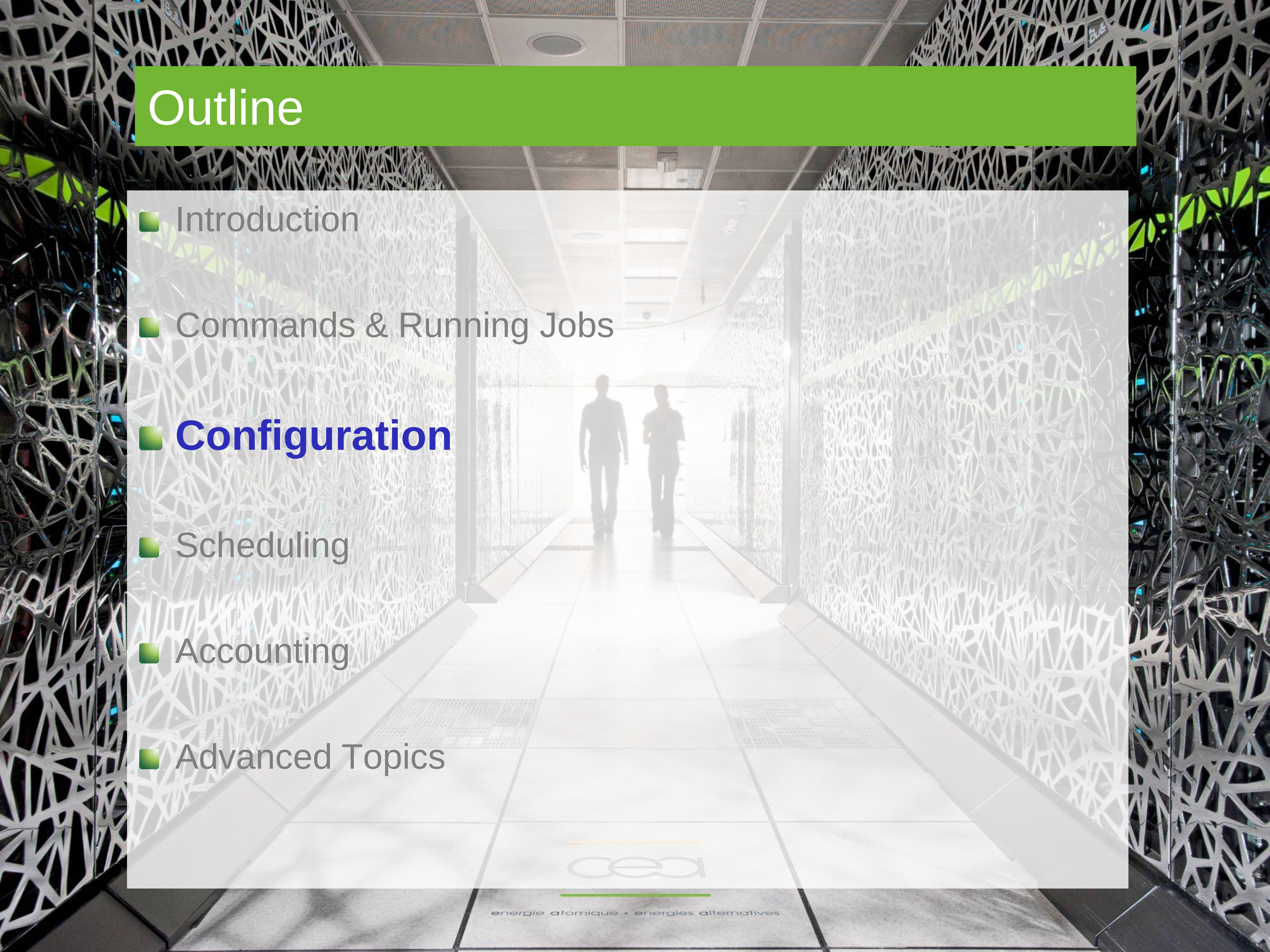
```
>squeue

JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
    106         P2     sleep    slurm   R        0:01      2 trek[1-2]
    107         P3     sleep    slurm   R        0:01      1 trek1
    108        all     sleep    slurm  PD        0:00      1 (Resources)
    105        all     sleep    slurm   R        0:02      1 trek0
```

More info commands ...

```
> scontrol show job 108

JobId=108 Name=sleep
   UserId=slurm(200) GroupId=slurm(200)
   Priority=4294901733 Account=slurm QOS=normal
   JobState=PENDING Reason=Resources Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=UNLIMITED TimeMin=N/A
   SubmitTime=2011-07-12T09:15:39 EligibleTime=2011-07-12T09:15:39
   StartTime=2012-07-11T09:15:38 EndTime=Unknown
   PreemptTime=NO_VAL SuspendTime=None SecsPreSuspend=0
   Partition=all AllocNode:Sid=sulu:8023
   ReqNodeList=trek0 ExcNodeList=(null)
   NodeList=(null)
   NumNodes=1 NumCPUs=1 CPUs/Task=1 ReqS:C:T=*:*:*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/bin/sleep
   WorkDir=/app/slurm/rbs/_Scripts
```



Outline

- Introduction
- Commands & Running Jobs
- **Configuration**
- Scheduling
- Accounting
- Advanced Topics

Configuration

slurm.conf

- Management policies
- Scheduling policies
- Allocation policies
- Node definition
- Partition definition
- Present on controller and all compute nodes

slurmdbd.conf

- Type of persistent storage (DB)
- Location of storage
- Admin choices

topology.conf

- Switch hierarchy

Others:

- plugstack.conf, gres.conf, cgroup.conf, ...

Configuration (slurm.conf)

Management Policies

- Location of controllers, backups, logs, state info
- Authentication
- Cryptographic tool
- Checkpoint
- Accounting
- Logging
- Prolog / epilog scripts
- Process tracking

Configuration (slurm.conf) …

```
# Sample config for SLURM Users Group
# Management Policies
ClusterName=rod
ControlMachine=sulu
SlurmUser=slurm
SlurmctldPort=7012
SlurmdPort=7013
AuthType=auth/munge
CryptoType=crypto/munge

# Location of logs and state info
StateSaveLocation=/app/slurm/rbs/tmp_slurm/rbs-slurm/tmp
SlurmdSpoolDir=/app/slurm/rbs/tmp_slurm/rbs-slurm/tmp/slurmd.%n.spool
SlurmctldPidFile=/app/slurm/rbs/tmp_slurm/rbs-slurm/var/run/slurmctld.pid
SlurmdPidFile=/app/slurm/rbs/tmp_slurm/rbs-slurm/var/run/slurmd.%n.pid
SlurmctldLogFile=/app/slurm/rbs/tmp_slurm/rbs-slurm/slurmctld.log
SlurmdLogFile=/app/slurm/rbs/tmp_slurm/rbs-slurm/slurmd.%n.log.%h

# Accounting
AccountingStorageType=accounting_storage/slurmdbd
AccountingStorageEnforce=limits
AccountingStorageLoc=slurm3_db
AccountingStoragePort=8513
AccountingStorageHost=sulu
```

Configuration (slurm.conf) …

Scheduling policies

- Priority
- Preemption
- Backfill

```
# Scheduling Policies
SchedulerType=sched/builtin
FastSchedule=1
PreemptType=preempt/partition_prio
PreemptMode=GANG,SUSPEND
```

Configuration (slurm.conf)

Allocation policies

- Entire nodes or 'consumable resources'
- Task Affinity (lock task on CPU)
- Topology (minimum number of switches)

```
# Allocaton Policies
SelectType=select/cons_res
SelectTypeParameters=CR_Core
TaskPlugin=task/cgroup
```

Configuration (slurm.conf)

Node definition

- Characteristics (sockets, cores, threads, memory, features)
- Network addresses

```
# Node Definitions
NodeName=DEFAULT Sockets=2 CoresPerSocket=4 ThreadsPerCore=1
NodeName=trek[0-31]
NodeName=trek[32-63] Sockets=2 CoresPerSocket=4 ThreadsPerCore=2 Feature=HyperThread
```

Configuration (slurm.conf)

Partition definition

- Set of nodes
- Sharing
- Priority/preemption

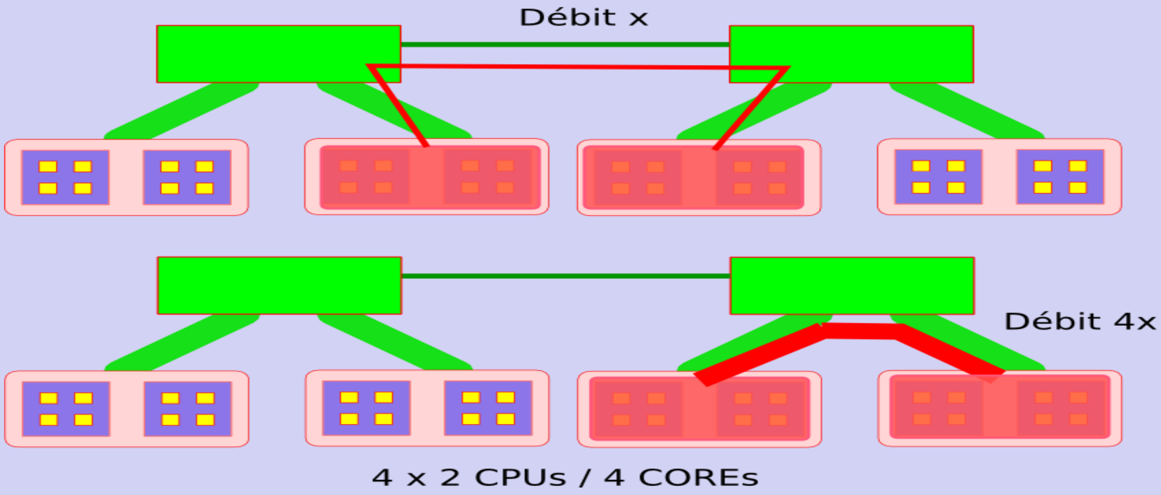
```
# Partition Definitions
PartitionName=all Nodes=trek[0-63] Shared=NO Default=YES
PartitionName=P2 Nodes=trek[0-63] Shared=NO Priority=2 PreemptMode=CANCEL
PartitionName=P3 Nodes=trek[0-63] Shared=NO Priority=3 PreemptMode=REQUEUE
PartitionName=P4 Nodes=trek[0-63] Priority=1000 AllowGroups=vip
PartitionName=MxThrd Nodes=trek[32-63] Shared=NO
```

Why use multiple partitions

- Provide different capabilities for different groups of users.
- Provides multiple queue for priority (with different preemption behavior)
- Provide subsets of the cluster.
- Group machines with same features (hyperthreading)
- Provide sharing.

Network Topology Aware Placement

- topology/tree SLURM Topology aware plugin. **Best-Fit** selection of resources
- In fat-tree hierarchical topology: Bisection Bandwidth Constraints need to be taken into account



```
#slurm.conf file
TopologyPlugin=topology/tree
```

Configuration (topology.conf)

topology.conf file needs to exist on all computing nodes
for network topology architecture description

```
# topology.conf file
SwitchName=Top Switches=IS1,IS2

SwitchName=IS1 Switches=TS1,TS2
SwitchName=IS2 Switches=TS3,TS4

SwitchName=TS1 nodes=knmi[1-18]
SwitchName=TS2 nodes=knmi[19-37]
SwitchName=TS3 nodes=knmi[38-56]
SwitchName=TS4 nodes=knmi[57-75]
....
```

Outline

- Introduction
- Commands & Running Jobs
- Configuration
- **Scheduling**
- Accounting
- Advanced Topics

Scheduling Policies

Scheduler Type

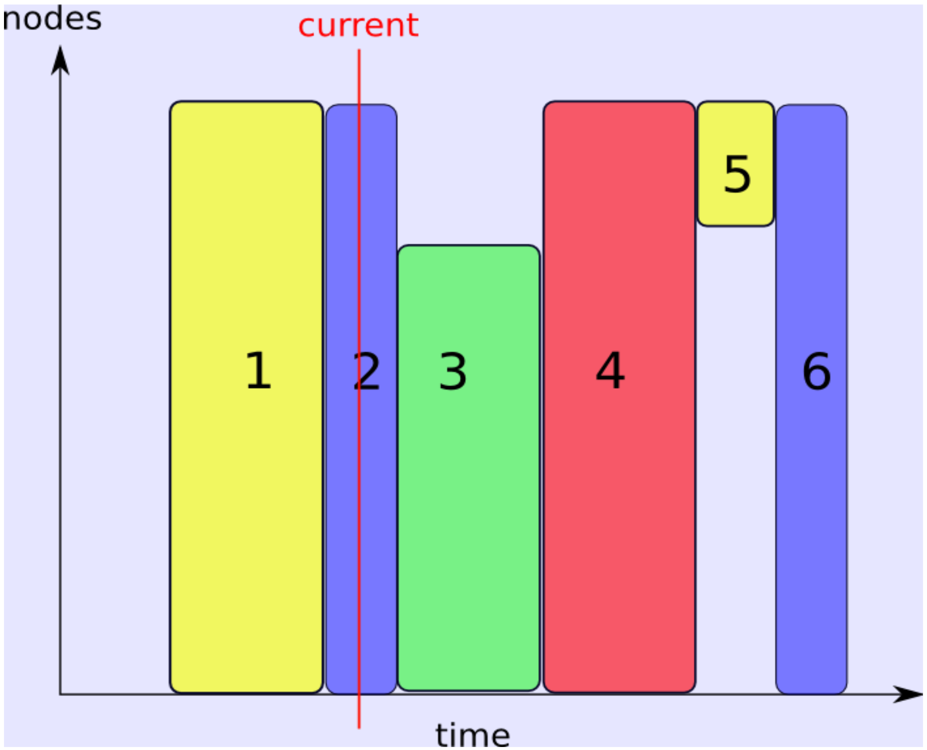
Sched/builtin Default FIFO

Sched/backfill schedule jobs as long as they don't delay a waiting job that is higher in the queue.

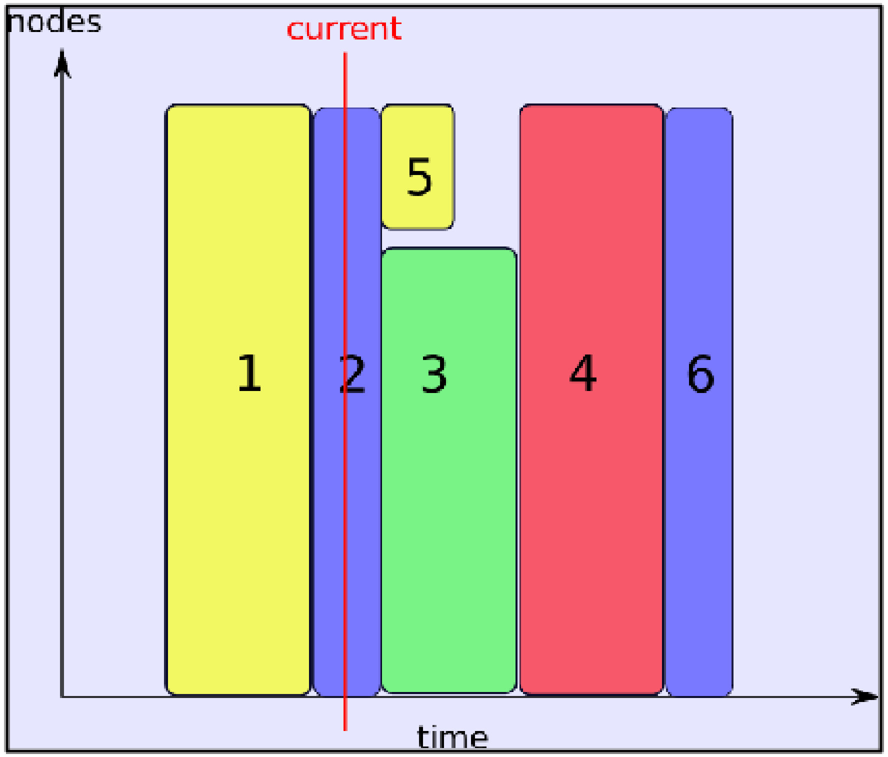
- Increases utilization of the cluster.
- Requires declaration of max execution time of jobs.
  - --time on 'srun',
  - DefaultTime or MaxTime on Partition
  - MaxWall from accounting association

Backfill Theory

Holes can be filled if previous jobs order is not changed



FIFO Scheduler



Backfill Scheduler

Backfill Example

```
srun -j C1 -N4 sleep 10
srun -j C2 -N1 -time=4 sleep 60
srun -j C3 -N4 -time=1 sleep 10
srun -j C4 -N2 -time=2 sleep 30
srun -j C5 -N3 -time=1 sleep 10
srun -j C6 -N1 -time=1 sleep 15
```

With Backfill
C1 Terminates
C2 Starts
C3 Pending, not enough nodes
C4 Backfills, limit less than C2
C5 Pending, can't backfill as not enough nodes
C6 Backfills, limit less than C2
C4 Terminates
C6 Terminates
C5 now backfills
C2 terminates
C3 waits for C5 to terminate.
C5's termnation still before C2's expected termination.

Note: it is important to have accurate estimated times.

	FIFO Scheduler											
Node												
0	C1	C2	C2	C2	C2	C2	C2	C3	C4	C4	C4	C5
1	C1							C3	C4	C4	C4	C5
2	C1							C3				C5
3	C1							C3				C6
	0:10	0:20	0:30	0:40	0:50	1:00	1:10	1:20	1:30	1:40	1:50	2:00
	<---- Time ----->											

	Backfill Scheduler							
Node								
0	C1	C2	C2	C2	C2	C2	C2	C3
1	C1	C4	C4	C4			C5	C3
2	C1	C4	C4	C4			C5	C3
3	C1	C6	C6	C6	C6		C5	C3
	0:10	0:20	0:30	0:40	0:50	1:00	1:10	1:20
	<----- Time ----->							

Preemption Policies

Preempt Types

None
Partition_prio priority defined on partition definition.
Qos quality of service defined in accounting database.

Example of **Partition_prio**

```
PartitionName=all Nodes=trek[0-63] Shared=NO Default=YES
PartitionName=P2 Nodes=trek[0-63] Shared=NO Priority=2 PreemptMode=CANCEL
PartitionName=P3 Nodes=trek[0-63] Shared=NO Priority=3 PreemptMode=REQUEUE
PartitionName=P4 Nodes=trek[0-63] Priority=1000 AllowGroups=vip
```

Define QOS

```
sacctmgr add qos meremortal
sacctmgr add qos vip Preempt=meremortal PreemptMode=cancel
```

Include QOS in association definition

```
sacctmgr add user Rod DefaultAccount=math qos=vip,normal DefaultQOS=normal
```

Preemption Policies

Preempt Modes

Off

Cancel preempted job is cancelled.

Checkpoint preempted job is checkpointed if possible, or cancelled.

Gang enables time slicing of jobs on the same resource.

Requeue job is requeued and restarted at the beginning (only for sbatch).

Suspend job is suspended until the higher priority job ends (requires Gang).

Preemption Example

Naming Conventions,

Partition name

1st Character is Preemmpt mode (Requeue, Cancel, Suspend, None)
2nd Character is priority.

Job name

1st Character is 'B', 2nd is submit order,
3rd is priority, 4th is Preempt mode of partition

```
PartitionName=R1 Nodes=trek[0-2] Priority=1 PreemptMode=REQUEUE
PartitionName=C1 Nodes=trek[0-2] Priority=1 PreemptMode=CANCEL
PartitionName=S1 Nodes=trek[0-2] Priority=1 PreemptMode=SUSPEND
PartitionName=S2 Nodes=trek[0-2] Priority=2 PreemptMode=SUSPEND
PartitionName=R3 Nodes=trek[0-2] Priority=3 PreemptMode=REQUEUE
PartitionName=N4 Nodes=trek[0-2] Priority=4
```

```
sbatch -J B11R -N1 --time=02:00 -P R1 echodate.bash 30
srun   -J B21C -N1 --time=02:00 -P C1 sleep 85
srun   -J B31S -N2 --time=01:00 -P S1 sleep 10
srun   -J B41S -N1 --time=01:00 -P S1 sleep 30
srun   -J B52S -N3 --time=01:00 -P S2 sleep 20
sbatch -J B63R -N2 --time=02:00 -P R3 echodate.bash 60
srun   -J B74N -N3 -P N4 sleep 5
```

B31S is queue for resource
B41S backfills

Running Jobs

Node								
0	B11R	B11R	B52S	B63R	B74N	B52S	B63R	B31S
1		B21C	B52S	B63R	B74N	B52S	B63R	B31S
2		B41S	B52S	B41S	B74N	B52S	B41S	B11R

<---- Time ----->

Suspended Jobs

B41S	B52S	B52S	B41S
	B52S	B52S	
	B52S	B52S	
		B41S	

Queued Jobs

B31S	B31S	B31S	B63R	B63R	B31S
B31S	B31S	B31S	B63R	B63R	B31S
	B11R	B11R	B31S	B31S	B11R
			B31S	B31S	
			B11R	B11R	

Allocation Policies

Select Types

Linear entire nodes are allocated, regardless of the number of tasks (cpus) required.

Cons_res cpus and memory as a consumable resource. Individual resources on a node may be allocated (not shared) to different jobs. Options to treat CPUs, Cores, Sockets, and memory as individual resources that can be independently allocated. Useful for nodes with several sockets and several cores per socket.

Bluegene for three-dimensional BlueGene systems

Allocation (Task Assignment) Policies

Task Plugin controls assignment (binding) of tasks to CPUs

None All tasks on a node can use all cpus on the node.

Cgroup cgroup subsystem is used to contain job to allocated CPUs. Portable Hardware Locality (hwloc) library used to bind tasks to CPUs.

Affinity Bind tasks with one of the following

Cpusets use cpuset subsystem to contain cpus assigned to tasks.
Sched use sched_setaffinity to bind tasks to cpus.

In addition, a *binding unit* may also be specified. It can be one of
Sockets, Cores, Threads, None

Both the are specified on the TaskPluginParam statement.

More on Partitions

Shared Option

Controls the ability of the partition to execute more than one job on a resource (node, socket, core)

EXCLUSIVE allocates entire node (overrides cons_res ability to allocate cores and sockets to multiple jobs)

NO sharing of any resource.

YES all resources can be shared, unless user specifies –exclusive on srun | salloc | sbatch

FORCE all resources can be shared and user cannot override. (Generally only recommended for BlueGene, although FORCE:1 means that users cannot use –exclusive, but resources allocated to a job will not be shared.)

Outline

- Introduction
- Commands & Running Jobs
- Configuration
- Scheduling
- **Accounting**
- Advanced Topics

Accounting

SLURM Accounting **Records Resource** usage by users and enables controlling their access (Limit Enforcement) to resources.

Limit Enforcement mechanisms

- Fairshare
- Quality of Service (QOS)
- Time and count limits for users and groups

More on this later.

For full functionality, the accounting daemon, **slurmdbd** must be running and using the **MySQL** database.

See the **accounting.html** page for more detail.

Accounting ...

Configuration options associated with resource accounting

AccountingStorageType controls how information is recorded (MySQL with SlurmDBD is best)
AccountingStorageEnforce enables Limits Enforcement.
JobAccntGatherType controls the mechanism used to gather data. (OS Dependent)
JobCompType controls how job completion information is recorded.

Commands

sacctmgr is used to create account and modify account settings.
sacct reports resource usage for running or terminated jobs.
sstat reports on running jobs, including imbalance between tasks.
sreport generates reports based on jobs executed in a time interval.

Sacctmgr

Used to define clusters, accounts, users, etc in the database.

Account Options

- **Clusters** to which the Account has access
- **Name**, **Description** and **Organization**.
- **Parent** is the name of an account for which this account is a child.

User Options

- **Account(s)** to which the user belongs.
- **AdminLevel** is accounting privileges (for sacctmgr). None, Operator, Admin
- **Cluster** limits clusters on which accounts user can be added to.
- **DefaultAccount** is the account for the user if an account is not specified on srun
- **QOS** quality of services user can use
- Other limits and much more.

Accounting Associations

An Association is a combination of a Cluster, a User, and an Account.

- An accounting database may be used by multiple **Clusters**.
- **Account** is a slurm entity like 'science' or 'math'.
- **User** is a Linux user like 'Rod' or 'Nancy'

Use **–account** srun/salloc/sbatch option to specify the Account

With associations, a user may have different privileges on different clusters.

A user may also be able to use different accounts, with different privileges.

Limit enforcement control apply to associations

Accounting Association Example

Add a cluster to the database (matches ClusterName from slurm.conf)

```
sacctmgr add cluster snowflake
```

Add an account

```
sacctmgr add account math Cluster=snowflake Description="math students" Organization="Bull"
```

Add let a user use the account, and place limits on him

```
sacctmgr add user Rod DefaultAccount=math qos=vip,normal DefaultQOS=normal
```

Accounting – Limits Enforcement

If a user has a limit set SLURM will read in those, if not we will refer to the account associated with the job. If the account doesn't have the limit set we will refer to the cluster's limits. If the cluster doesn't have the limit set no limit will be enforced.

Some (but not all limits are)

Fairshare= Integer value used for determining priority. Essentially this is the amount of claim this association and it's children have to the above system.

GrpCPUMins= A hard limit of cpu minutes to be used by jobs running from this association and its children. If this limit is reached all jobs running in this group will be killed, and no new jobs will be allowed to run. (GrpCPUs, GrpJobs, GrpNodes, GrpSubmitJobs, GrpWall)

MaxCPUMinsPerJob= A limit of cpu minutes to be used by jobs running from this association. If this limit is reached the job will be killed. (MaxCPUsPerJob, MaxJobs, MaxNodesPerJob, MaxSubmitJobs, MaxWallDurationPerJob)

QOS (quality of service) comma separated list of QOS's this association is able to run.

Multifactor Priority Plugin

By default, SLURM assigns job priority on a First In, First Out (FIFO) basis. (`PriorityType=priority/basic` in the slurm.conf file.)

SLURM now has a Multi-factor Job Priority plugin.

`(PriorityType=priority/multifactor)`

This plugin provides a very versatile facility for ordering the queue of jobs waiting to be scheduled.

It requires the accounting database as previously described.

Multifactor Factors

Age the length of time a job has been waiting in the queue, eligible to be scheduled

Fair-share the difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed

Job size the number of nodes a job is allocated

Partition a factor associated with each node partition

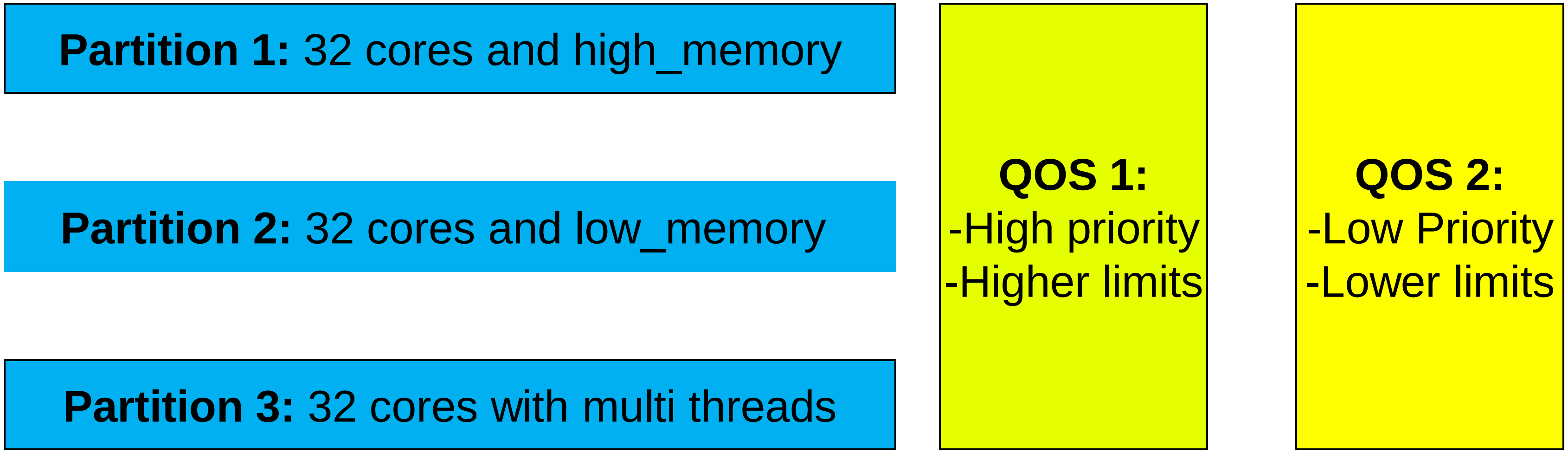
QOS a factor associated with each Quality Of Service

Additionally, a weight can be assigned to each of the above factors. This provides the ability to enact a policy that blends a combination of any of the above factors in any portion desired. For example, a site could configure fair-share to be the dominant factor (say 70%), set the job size and the age factors to each contribute 15%, and set the partition and QOS influences to zero.

See **priority_multifactor.html** and **qos.html** for more detail

Partitions and Multifactor (with QOS)

- **Partitions** and **Multifactor Priority** are used in SLURM to group nodes and jobs characteristics
- The use of Partitions and Multifactor Priority entities in SLURM is orthogonal:
  - Partitions for grouping resources characteristics
  - QOS factor for grouping limitations and priorities



Partition 1: 32 cores and high_memory

Partition 2: 32 cores and low_memory

Partition 3: 32 cores with multi threads

QOS 1:
-High priority
-Higher limits

QOS 2:
-Low Priority
-Lower limits

Partitions and QOS Configuration

Partitions Configuration:
In slurm.conf file

```
# Partition Definitions
PartitionName=all Nodes=trek[0-95] Shared=NO Default=YES
PartitionName=HiMem Nodes=trek[0-31] Shared=NO
PartitionName=LoMem Nodes=trek[32-63] Shared=NO
PartitionName=MxThrd Nodes=trek[64-95] Shared=NO
```

QOS Configuration:
In Database

```
>sacctmgr add qos name=lowprio priority=10 PreemptMode=Cancel GrpCPUs=10 MaxWall=60 MaxJobs=20
>sacctmgr add qos name=hiprio priority=100 Preempt=lowprio GrpCPUs=40 MaxWall=120 MaxJobs=50
>sacctmgr list qos
      Name   Priority Preempt PreemptMode     GrpCPUs  MaxJobs  MaxWall
---------- ---------- ---------- ---------- ----------- --------------------------------------- ---------- -------- ----------- -------
   lowprio      10                  cancel            10           20         60
     hiprio     100      lowprio                      40           50        120
```

Running Jobs

To get resource characteristics select partition

To get nodes with hyperthreads

```
srun -p MxThrd …
```

To get priority use appropriate QOS

To get high priority

```
srun –qos=hiprio --account=vip
```

Outline

- Introduction
- Commands & Running Jobs
- Configuration
- Accounting
- Scheduling
- **Advanced Topics**

Site Functionality for SLURM

Site Optional Scripts

Prolog (before an event) and Epilog (after an event)

- Before and after a job on the controller (slurmctld)
- Before and after a job an a compute node
- Before and after each task on a compute node.
- Before and after srun (on the client machine)

Spank plugin

- ‘c’ code in a shared library.
- Don’t need to modify slurm source.
- Called at specific life cycle events.
- API to get job characteristics.

bullx

instruments for innovation

powered by Bull