

# SC3260 / SC5260

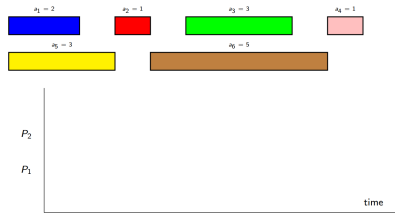
## Batch Scheduler - part 2 -

Lecture by: Ana Gainaru

# Why do we need batch schedulers?

The scheduler can be used to balance all the metrics

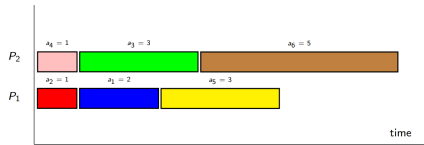
- ▶ User fairness
- ▶ System utilization
- ▶ Application response time



## Longest job first



## Shortest job first



VANDERBILT  
UNIVERSITY

## Currently used: FCFS scheduling policy with backfilling

Let's look at some examples to better understand the difference.

### ScheduleFlow

- ▶ Scheduler simulator developed by our group
- ▶ <https://github.com/anagainaru/ScheduleFlow>



VANDERBILT  
UNIVERSITY

## Where did we left off?

- ① SLURM and how to use it
- ② A few promising directions for the future
  - ▶ Gang scheduling
  - ▶ Task based scheduler (work stealing)



## Simple Linux Utility for Resource Management

- ▶ Development started in 2002 @ Lawrence Livermore National Lab as a resource manager for Linux clusters
  - ▶ Sophisticated scheduling plugins added in 2008
- ▶ About 550,000 lines of C code today
- ▶ Supports Linux and limited support for other Unix variants
- ▶ Used on many of the world's largest computers
- ▶ Active global user community

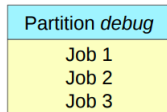


- ▶ Highly scalable
  - ▶ Managing 3.1 million core Tianhe-2
  - ▶ Tested to much larger systems using emulation
- ▶ Open source GPLv2, available on Github: <https://github.com/SchedMD/>
- ▶ Fault-tolerant (no single point of failure)
- ▶ Dynamically linked objects loaded at run time based upon configuration file and/or user options
- ▶ Multiple plugins in 30 classes currently available
  - ▶ Network topology: 3D-torus, tree, etc
  - ▶ MPI: OpenMPI, PMI2, PMIx
  - ▶ Process tracking: cgroup, linuxproc, pgid, ipmi, etc.

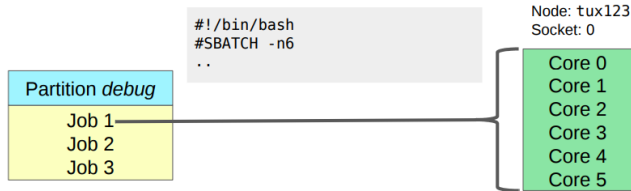


- Users submit jobs to partitions (queue)

Priority ordered queue of jobs



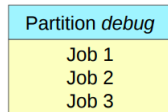
- Jobs are allocated resources



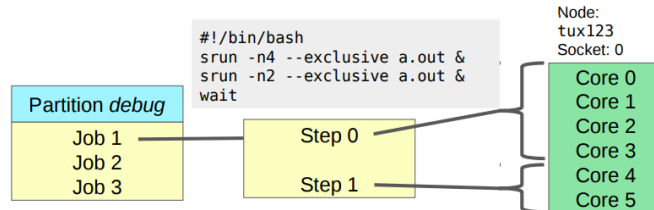
VANDERBILT  
UNIVERSITY

- Users submit jobs to partitions (queue)

Priority ordered queue of jobs

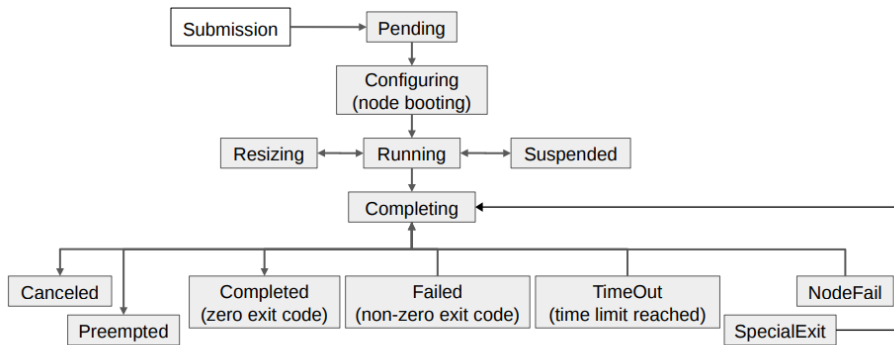


- Jobs spawn steps, which are allocated resources from within the job's allocation





# Job States



Copyright 2018 SchedMD LLC  
[www.schedmd.com](http://www.schedmd.com)  
SLUG Sept. 25&26th, 2018



VANDERBILT  
UNIVERSITY

# SLURM implementation

SLURM can be configured by system administrators in different formats

Frequently using a reservation-based configuration that assigns reservations on job arrival, job finish and at priority change.

- ▶ Can use one or multiple priority queues
- ▶ FCFS policy of ordering the queue
- ▶ Job priority is increased when waiting long time in the queue
- ▶ EASY Backfilling
- ▶ Node hours are required, job is killed in case of under-estimation
- ▶ Supports interactive sessions (`salloc`)
- ▶ Supports non-interactive sessions (`srun`, `sbatch`)



VANDERBILT  
UNIVERSITY

Documentation for SLURM available at <https://slurm.schedmd.com>

**What you used for the homeworks should be enough for most projects**

<https://hpc-carpentry.github.io/hpc-intro/13-scheduler/index.html>



VANDERBILT  
UNIVERSITY

Documentation for SLURM available at <https://slurm.schedmd.com>

**What you used for the homeworks should be enough for most projects**

<https://hpc-carpentry.github.io/hpc-intro/13-scheduler/index.html>

## Next

- ▶ Gang scheduling
- ▶ Task based scheduler (work stealing)



VANDERBILT  
UNIVERSITY

# Gang Scheduling: Basis

- ▶ All processes belonging to a job run at the same time (the term **gang** denotes all processors within a job).
- ▶ Each process runs alone on each processor.
- ▶ BUT: there is rapid **coordinated** context switching.
- ▶ It is possible to **suspend/preempt** jobs arbitrarily

May allow more flexibility to improve some metrics



VANDERBILT  
UNIVERSITY

# Gang Scheduling: Basis

- ▶ All processes belonging to a job run at the same time (the term **gang** denotes all processors within a job).
- ▶ Each process runs alone on each processor.
- ▶ BUT: there is rapid **coordinated** context switching.
- ▶ It is possible to **suspend/preempt** jobs arbitrarily

## May allow more flexibility to improve some metrics

- ▶ If processing times are not known in advance (or grossly erroneous), preemption can help short jobs that would be "stuck" behind a long job.
- ▶ Should improve machine utilization



VANDERBILT  
UNIVERSITY

# Gang Scheduling: Example

- ▶ A 128 node cluster.
- ▶ A running 64-node job.
- ▶ A 32-node job and a 128-node job are queued.

Should the 32-node job be started?

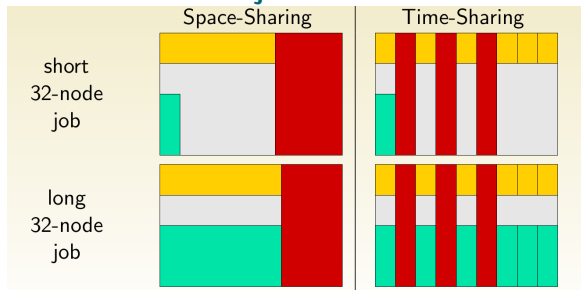


VANDERBILT  
UNIVERSITY

# Gang Scheduling: Example

- ▶ A 128 node cluster.
- ▶ A running 64-node job.
- ▶ A 32-node job and a 128-node job are queued.

Should the 32-node job be started?



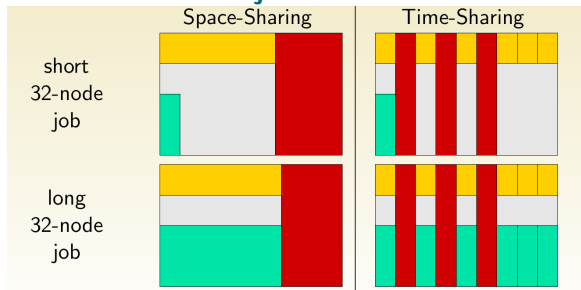
VANDERBILT  
UNIVERSITY



# Gang Scheduling: Example

- ▶ A 128 node cluster.
- ▶ A running 64-node job.
- ▶ A 32-node job and a 128-node job are queued.

Should the 32-node job be started?



More uniform slowdown, better resource usage.



VANDERBILT  
UNIVERSITY

- ▶ Overhead for context switching (trade-off between overhead and fine grain)
- ▶ Overhead for coordinating context switching across multiple processors



- ▶ Overhead for context switching (trade-off between overhead and fine grain)
- ▶ Overhead for coordinating context switching across multiple processors
- ▶ Reduced cache efficiency(Frequent cache flushing)
- ▶ RAM Pressure (more jobs must fit in memory, swapping to disk causes unacceptable overhead)



- ▶ Overhead for context switching (trade-off between overhead and fine grain)
- ▶ Overhead for coordinating context switching across multiple processors
- ▶ Reduced cache efficiency(Frequent cache flushing)
- ▶ RAM Pressure (more jobs must fit in memory, swapping to disk causes unacceptable overhead)

Typically not used in production HPC systems (batch scheduling is preferred)

Some implementations (MOSIX, Kerighed)



VANDERBILT  
UNIVERSITY

# Gang Scheduling conclusion

- ▶ Nice idea, might work in the future.

For now it is not implemented in any of the major HPC systems



VANDERBILT  
UNIVERSITY

- ▶ A task-graph  $G$  needs to be executed on  $p$  processors.
- ▶ Non-clairvoyant setting : the structure of  $G$  and/or the execution times of its constitutive tasks are discovered online



VANDERBILT  
UNIVERSITY

# Sharing vs Stealing

## Batch scheduling

- ▶ Centralized scheduling
- ▶ A single list stores all ready tasks
- ▶ All processors retrieve work from that list
- ▶ Advantage(s)
  - ▶ Global view and knowledge
- ▶ Drawback(s)
  - ▶ Does not scale (contentions, etc.)

## Work stealing

- ▶ Distributed scheduling
- ▶ Each processor owns a list of 'its' ready tasks
- ▶ Advantage(s)
  - ▶ No contention problem
  - ▶ Scalable solution
- ▶ Drawback(s)
  - ▶ Processors with empty lists do not know where to retrieve work from



VANDERBILT  
UNIVERSITY

## Global round-robin

- ▶ A global variable holds the identity of the next processor to steal from
- ▶ Variable incremented after each steal (successful or not)
- ▶ Advantage : eventual progress
- ▶ Drawback : centralized solution...

## Local round-robin

- ▶ Each processor has its own variable indicating the next processor it should try to steal from
- ▶ Variable incremented after each steal (successful or not)
- ▶ Advantage : eventual progress ; solution is scalable
- ▶ Drawback : all stealing processors may attempt to steal from the same processor; arbitrary notion of "distance" between processors





# For now we're stuck with batch scheduling

Why don't we like Batch Scheduling?



VANDERBILT  
UNIVERSITY

# For now we're stuck with batch scheduling

**Why don't we like Batch Scheduling?** Because queue waiting times are difficult to predict

- ▶ depends on the status of the queue
- ▶ depends on the scheduling algorithm used
- ▶ depends on all sorts of configuration parameters set by system administrator
- ▶ depends on future job completions!
- ▶ etc.

There is more and more demand for reservation support



VANDERBILT  
UNIVERSITY

**Batch schedulers are complex pieces of software that are used in practice**

- ▶ A lot of experience on how they work and how to use them
- ▶ But ultimately everybody knows they are an imperfect solution
- ▶ Many view the lack of theoretical foundations as a big problem

**You need to know about them since every cluster uses them**



VANDERBILT  
UNIVERSITY

### Book on the theory of scheduling

D.B. Shmoys, J. Wein, and D.P. Williamson. *Scheduling parallel machines on-line* Symposium on Foundations of Computer Science, 0:131-140, 1991.

### SLURM slides from

[https://slurm.schedmd.com/SLUG18/slurm\\_overview.pdf](https://slurm.schedmd.com/SLUG18/slurm_overview.pdf)

### List of SLURM commands

<https://hpc-carpentry.github.io/hpc-intro/13-scheduler/index.html>

Figures from today's slides courtesy of Arnaud Legrand and Guillaume Pallez

[http://people.bordeaux.inria.fr/gaupy/ressources/teachings/2019/algo\\_hpc/](http://people.bordeaux.inria.fr/gaupy/ressources/teachings/2019/algo_hpc/)



VANDERBILT  
UNIVERSITY