pawsey

Supercomputing User Training

Module 8: Job Scheduling Overview

Pawsey Training Series

Supercomputing User Training

1. Supercomputing Introduction
2. Logging In
3. Filesystems Overview
4. Moving Data In and Out
5. Using Software Modules
6. Using Software Containers
7. Accounting Model Overview
8. Job Scheduling Overview
9. Running Jobs
10. Testing Job Runs
11. Managing Project Data

Outcomes for this Module

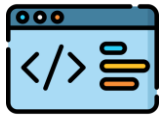
- Describe what a job scheduler is and why it is useful
- Query scheduler partitions
- Query job queues

✓ Prerequisite knowledge:

- ✓ **Bash shell basics**
- ✓ **User Training 02: Logging In**
- ✓ **User Training 07: Accounting Model Overview**

Watch for These Signs!

Definition of new concepts



Hands-on coding (demo)

Best practices

Exercises and solutions

Warnings (bad practices)

Links to user documentation

Icons made by Freepik from www.flaticon.com

Pawsey

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

The University of Western Australia

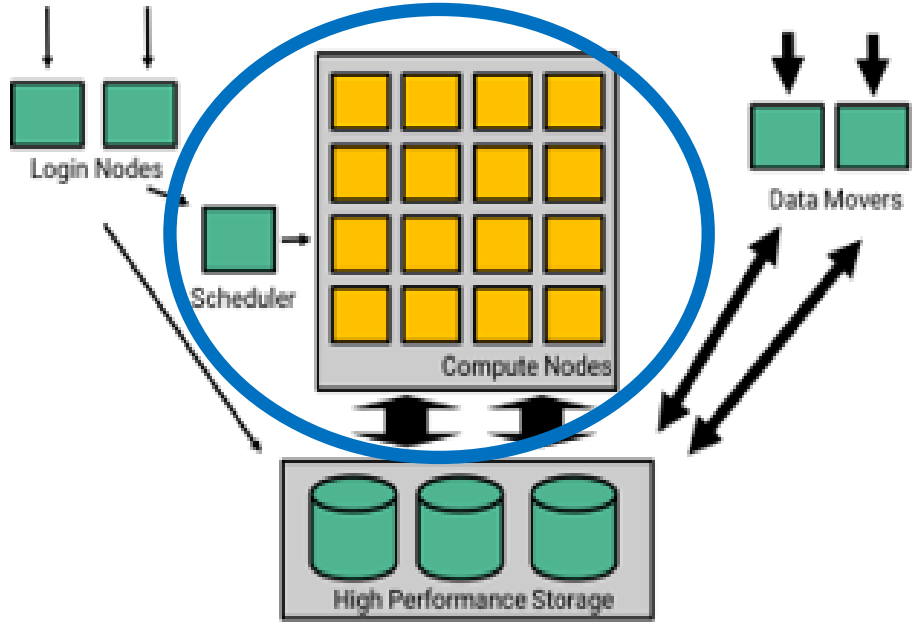
The Slurm Scheduler at Pawsey

What is a Job Scheduler?

Job
A unit of computational work to be executed on a supercomputer.

Job Scheduler
A program that manages queuing and execution of jobs by multiple users on a shared computer.

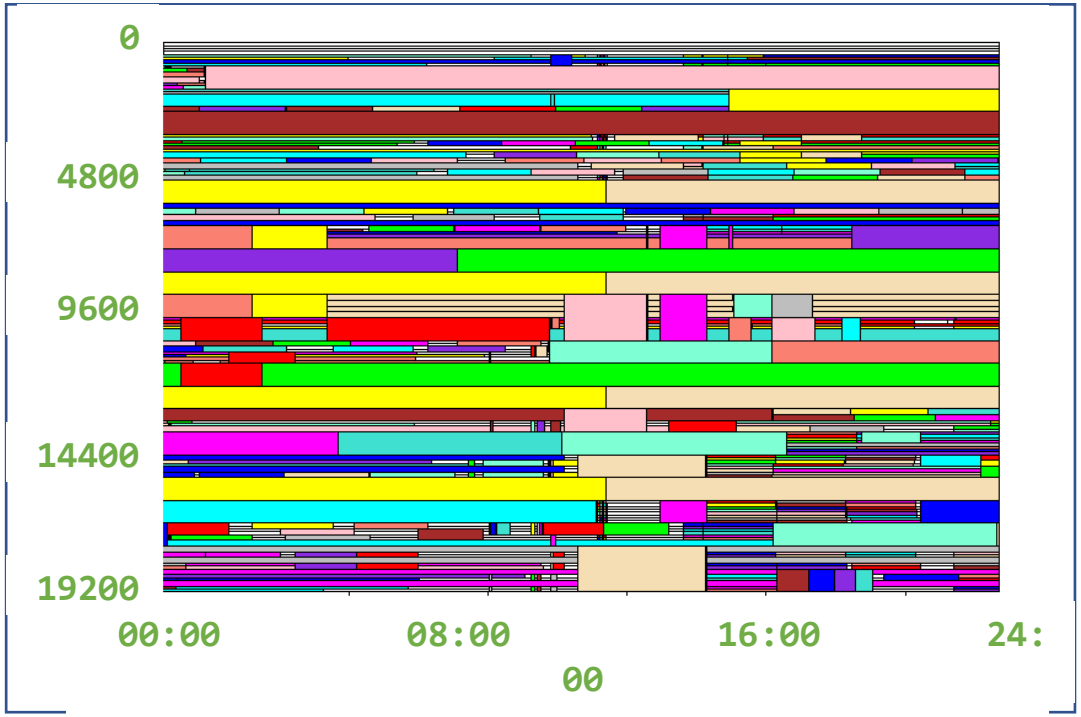
- At Pawsey we use the “Slurm” scheduler by SchedMD
- The login nodes are used to submit jobs for execution to the scheduler
- The scheduler executes jobs on the compute nodes



Do not run any computationally intensive programs on the login nodes
Be mindful. Login nodes are designed for lightweight tasks, and shared across 100s of users.

More details @
- Slurm Documentation (external link)

How is a Job Scheduler Useful?

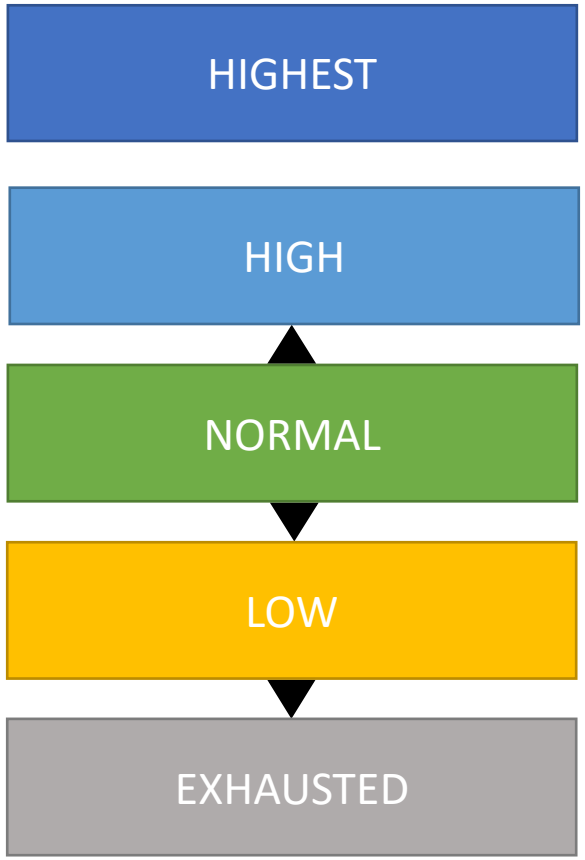


- Imagine a calendar planner with
  - hours of the day
  - compute resources instead of week days
  - each coloured bar represents a job
- Users submit jobs with different durations and different compute requirements
- The scheduler queues job requests over time and available resources
  - to maximise efficiency in using the shared supercomputer
  - taking into account compute allocations (via job “priorities”)

Plan and organise your computational work with the scheduler in mind
Maximise the amount of your jobs in the scheduler queue. Do not just wait for jobs to start. Maximise automation, too.

Job Priorities and Quality of Service Levels

- The scheduler assign job priorities based on:
  - Usage relative to allocation (priority decreases as the allocation is used up)
  - Waiting time in queue (priority increases with time)
  - Size of job (priority increases with size)
  - Quality-of-Service level



Reserved for critical work.
Motivated request via Helpdesk.

Selected by users.
Up to 10% of allocation.
Good for urgent jobs.

0% to 100% of allocation.
Default.

100% to 150% of allocation.

Over 150% of allocation.
Can no longer run jobs.

Slurm Quality-of-Service Level
A modifier to the priorities assigned to submitted jobs by the scheduler.

NOTE: Quality-of-service levels reset each quarter.

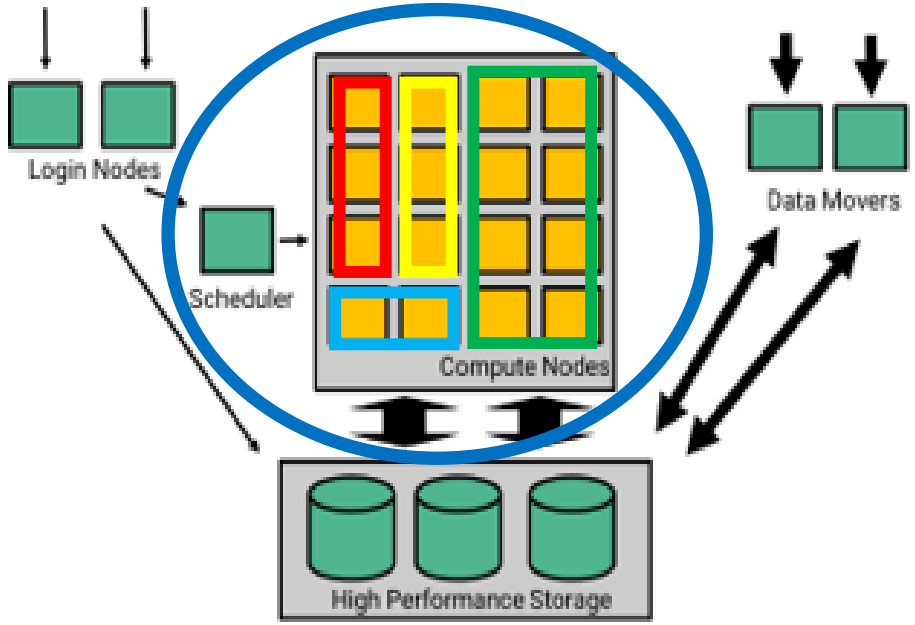
Some Slurm jargon: Clusters and Partitions

Slurm Cluster

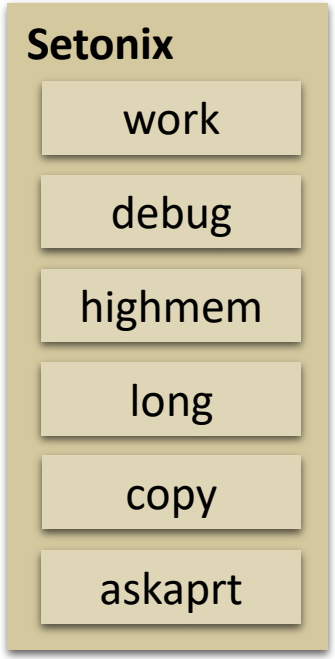
A set of compute resources all managed by the same Slurm daemon (program instance).
At Pawsey, Slurm clusters map to Pawsey supercomputers: Setonix, Topaz, Garrawarla, ...
A Slurm cluster is subdivided in Slurm partitions.

Slurm Partition

A subset of resources with an associated queue of jobs.
Jobs submitted to the scheduler are always assigned to a specific Slurm partition.



Slurm Partitions in Setonix Phase 1

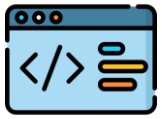


Specifications
(unless otherwise stated)

- 128 cores per node
- 230 GB RAM per node, which is 1.8 GB RAM per node
- 24 hours maximum job wall time

- `work` partition **(316 nodes)**
  - Use for regular compute jobs
  - Default partition
- `debug` partition **(8 nodes)**
  - Use for compiling, benchmarking, testing, and development
  - Maximum wall time is 1 hour
- `highmem` partition **(8 nodes)**
  - Use for jobs requiring more than 230 GB RAM memory per node
  - 980 GB available memory per node, which is 7.65 GB per core
- `long` partition **(8 nodes)**
  - Use for jobs requiring wall times between 24 and 96 hours
- `copy` partition **(8 nodes)**
  - Use for all data transfers, in, out and within Pawsey
  - 32 cores per node, 89 GB RAM per node, which is 2.8 GB per core
- `askaprt` partition **(180 nodes)**
  - Reserved for the operational workflow of the ASKAP radio telescope

Querying Slurm Partitions



DEMO on Setonix: `sinfo` – let's do this together

Key information in the `sinfo` output:

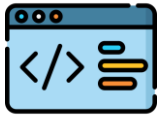
- PARTITION: list of partitions, multiple entries per partition classified by status
- STATE: status of entries of nodes within each partition
  - idle
  - alloc: allocated
  - mix: mixed use (fraction of nodes are allocated)
  - resv: reserved use (e.g. for user training)
  - down: not available
  - maint: maintenance
  - drain: temporarily unavailable

More details @
- Job Scheduling

OUTPUTS: Querying Slurm Partitions

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
work*        up 1-00:00:00      2  down$ nid[001070-001071]
work*        up 1-00:00:00      5  down* nid[001192,001205,001212-001213,001267]
work*        up 1-00:00:00     11   down nid[001100-001103,001135,001145,001214-001215,001264-001266]
work*        up 1-00:00:00      1    mix nid001206
work*        up 1-00:00:00    297   idle nid[001008-001069,001072-001099,001104-001134,001136-
001144,001146-001191,001193-001204,001207-001211,001216-001263,001268-001323]
long         up 4-00:00:00      8   idle nid[001316-001323]
copy         up 2-00:00:00      7  down* dm[02-08]
askaprt      up 1-00:00:00      3  maint nid[001488-001489,001491]
askaprt      up 1-00:00:00      1  down$ nid001490
askaprt      up 1-00:00:00    176   idle nid[001324-001487,001492-001503]
debug        up    1:00:00      3  maint nid[001004-001005,001007]
debug        up    1:00:00      1  down$ nid001006
debug        up    1:00:00      4   idle nid[001000-001003]
highmem      up 1-00:00:00      8   idle nid[001504-001511]
```

Querying Job Queues



DEMO on Setonix: `squeue`

- Use `squeue` to query the job queue in a Slurm cluster
- To make the output more readable, pipe it to Linux commands such as `head`, `less`, `wc`
- Useful `squeue` options (will use them in the Exercise)
  - Query a specific partition with `--partition` (or `-p`) *`partition-name`*
  - Query only your jobs with `--me`
- More useful `squeue` options (see user documentation)
  - Query a specific project account with `--account` (or `-A`) *`project-name`*
  - Query a specific user with `--user` (or `-u`) *`username`*
  - Change output format with `--format`, or the `SQUEUE_FORMAT` shell variable

More details @
- Job Scheduling

OUTPUTS: Querying Job Queues

```
$ squeue |head
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          156594+0   askaprt   imager   vor010  R    3:56:57      4 nid[001497-001500]
          156594+1   askaprt   imager   vor010  R    3:56:57     12 nid[001324-001335]
            157339   askaprt interact   pelahi  R      12:40      2 nid[001485,001487]
            123346      copy scrkeep1  dadams1 PD       0:00      1 (BeginTime)
            123545      copy scrkeep1   awoods PD       0:00      1 (BeginTime)
            123347      copy scrkeep2  dadams1 PD       0:00      1 (BeginTime)
            123546      copy scrkeep2   awoods PD       0:00      1 (BeginTime)
            156734      copy gcp-chai   wil240 PD       0:00      1 (Dependency)
            156912      copy manta_D0 nhurleyw PD       0:00      1 (ReqNodeNotAvail)
```

Key information on jobs in the `squeue` output

- Cluster PARTITION
- Submitting USER
- Elapsed TIME
- Number of requested NODES
- NODELIST (if running), REASON (if not running)

OUTPUTS: Querying Job Queues

```
$ squeue |head
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
         156594+0   askaprt   imager   vor010  R    3:56:57      4 nid[001497-001500]
         156594+1   askaprt   imager   vor010  R    3:56:57     12 nid[001324-001335]
           157339   askaprt interact   pelahi  R      12:40      2 nid[001485,001487]
           123346      copy scrkeep1  dadams1 PD       0:00      1 (BeginTime)
           123545      copy scrkeep1   awoods PD       0:00      1 (BeginTime)
           123347      copy scrkeep2  dadams1 PD       0:00      1 (BeginTime)
           123546      copy scrkeep2   awoods PD       0:00      1 (BeginTime)
           156734      copy gcp-chai   wil240 PD       0:00      1 (Dependency)
           156912      copy manta_D0 nhurleyw PD       0:00      1 (ReqNodeNotAvail)
```

- Status (ST) of the job
  - R: Running
  - PD: Pending
  - CG: Completing
  - F: Failed

EXERCISE: Querying Job Queues

1. How many jobs are currently queued on Setonix? (tip: pipe the relevant output to `wc -l`)
2. List only jobs that are queued in the `work` partition (tip: use the option `-p partition-name`)
3. How many jobs are currently queued in this partition?
4. List only jobs that are queued in the `copy` partition

OUTPUTS: Querying Job Queues

```
$ squeue | wc -l
no. of lines

$ squeue -p work
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
..

$ squeue -p work | wc -l
no. of lines

$ squeue -p copy
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
..
```

EXERCISE: Querying Job Queues

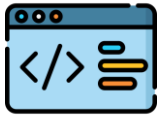
5. List only jobs that are queued by you (tip: use the option `--me`)
6. Now go into your `$MYSCRATCH` directory
7. Download the Github repository with exercise scripts for this training series
   - `git clone https://github.com/PawseySC/supercomputing-user-training`
8. Go into the directory: `supercomputing-user-training/job-scheduling-overview`
9. Submit the batch script `first_job.sh` using: `sbatch first_job.sh`
   - This is a spoiler from the next User Training (09 Running Jobs)
10. List again the jobs that are queued by you

OUTPUTS: Querying Job Queues

```
$ squeue --me
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)

$ cd $MYSCRATCH
$ git clone https://github.com/PawseySC/supercomputing-user-training
..
$ cd supercomputing-user-training/job-scheduling-overview

$ sbatch first_job.sh
Submitted batch job 157423

$ squeue --me
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
157423      work first_jo username  R       0:01      1 nid001008
```

Summary

- Terms we learnt
  - Job
  - Job Scheduler
  - Slurm cluster
  - Slurm partition
  - Slurm Quality-of-Service Level



- Tasks we learnt
  - Query scheduler partitions and job queues: `sinfo` and `squeue`

- Do not run any computationally intensive programs on the login nodes

- Plan and organise your computational work with the scheduler in mind

Pawsey

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

The University of Western Australia

Getting Help

Getting Help

https://support.pawsey.org.au

Pawsey has extensive User Support Documentation.

Areas covered include:

- System user guides
- Knowledge Base
- Pawsey-supported software list
- Maintenance logs
- Policies and terms of use

For further assistance, contact the help desk, via User Support Portal.

Help us to help you by providing details, such as:

- Which resource
- Error messages
- Location of files
- SLURM job id
- Your username if having login issues
- Never tell us (or anyone) your password!

Become a Pawsey Friend and receive our Newsletter:
https://pawsey.org.au/pawsey-friends/

Q & A Session