High Performance Computing

ICRAR/CASS Radio School

Oct 2, 2018

The Pawsey Supercomputing Centre is an unincorporated joint venture between

CSIRO, Curtin University, ECU Edith Cowan University, Murdoch University, The University of Western Australia

and proudly funded by

NCRIS National Research Infrastructure for Australia, Government of Western Australia

PAWSEY supercomputing centre

Overview

- Intro to Pawsey Supercomputing Centre
- Architecture of a supercomputer
- Basics of parallel computing
- Filesystems
- Software environment (modules)
- Job submission
- Project accounting

Pawsey Supercomputing Centre

PAWSEY
supercomputing centre

National Reach

ECU Edith Cowan University

Curtin University

The University of Western Australia

Murdoch University

PAWSEY supercomputing centre

ICRAR

Department of Mines and Petroleum

Government of Western Australia
Department of Biodiversity, Conservation and Attractions

Landgate

The University of Adelaide

CSIRO

DHI

The University Of Queensland
AUSTRALIA

Australian National University

UNSW

NCI

The University of Sydney

Swinburne University of Technology

RMIT University

The University of Melbourne

Monash University

PAWSEY supercomputing centre

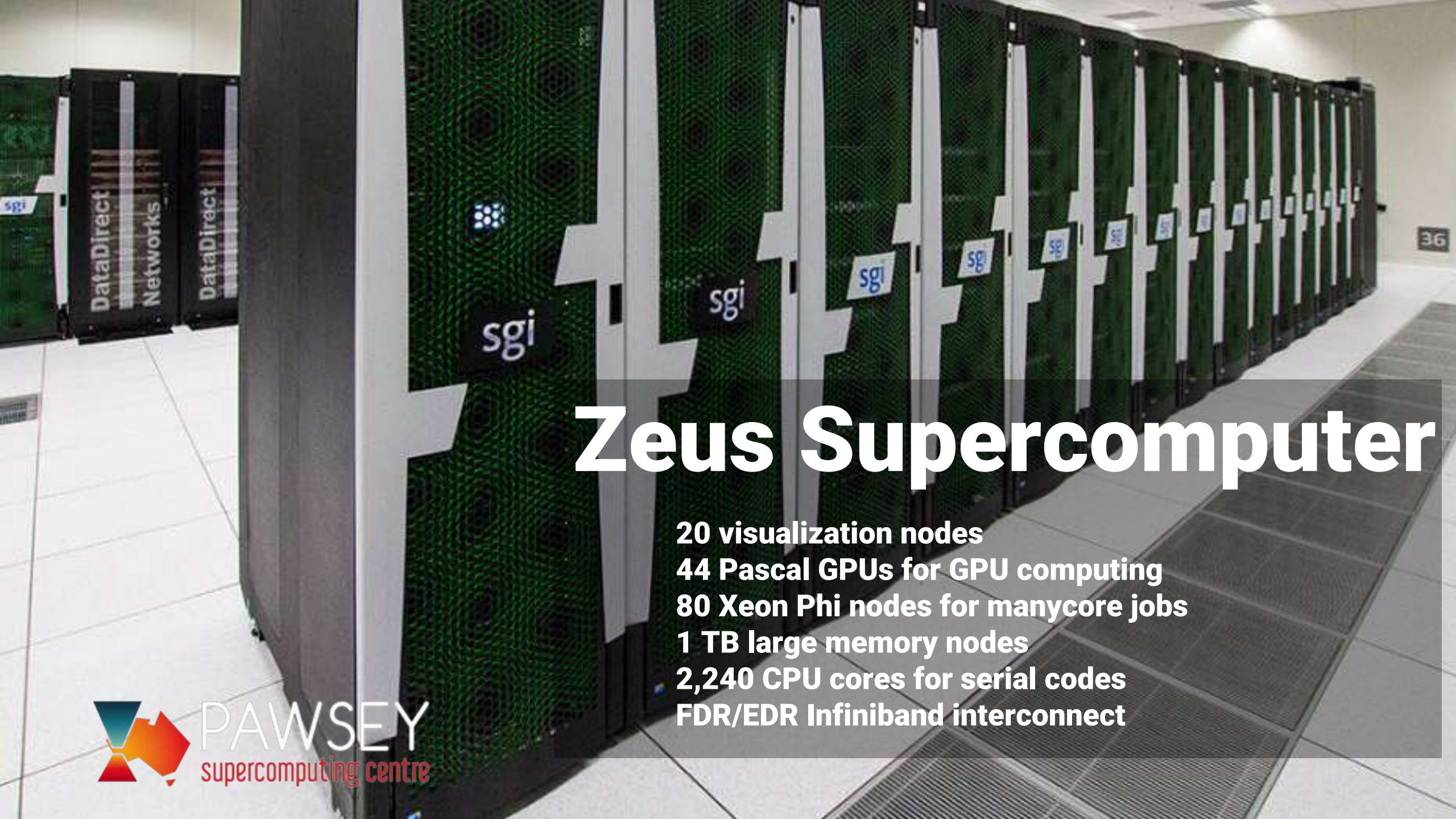
35,712 cores, 1.09 PFLOPS,
Aries dragonfly interconnect

Magnus Supercomputer

PAWSEY
supercomputing centre

9,440 CPU cores
64 K20X GPUs
Aries dragonfly interconnect

Galaxy Supercomputer



Zeus Supercomputer

20 visualization nodes
44 Pascal GPUs for GPU computing
80 Xeon Phi nodes for manycore jobs
1 TB large memory nodes
2,240 CPU cores for serial codes
FDR/EDR Infiniband interconnect

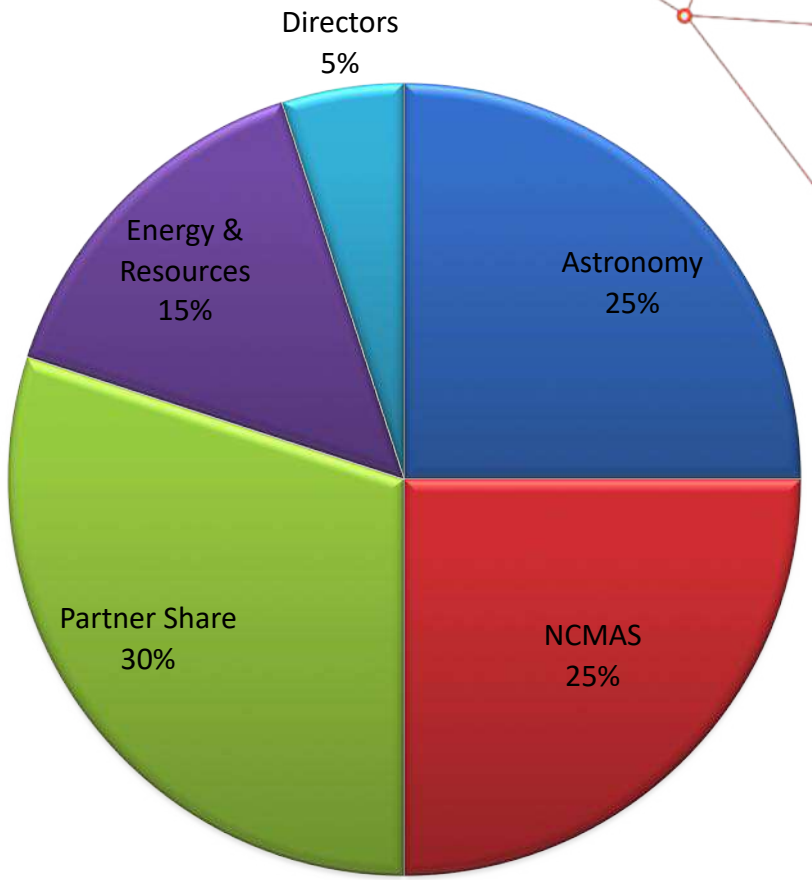
PAWSEY
supercomputing centre

Supercomputer Access

Access to Magnus via competitive merit calls:

- **NCMAS**
- **Energy and Resources**
- **Pawsey Partners**

Application call for 2019 closes on 19th of October



Astronomy share on Galaxy supports operation of ASKAP and MWA telescopes

Director share to help projects get started and prepare for merit

PAWSEY
supercomputing centre

3000 Cores, OpenStack,
Sahara, Volta GPUs

Nimbus Research Cloud

PAWSEY
supercomputing centre

65 PB Migrating Disk and Tape

Data Storage

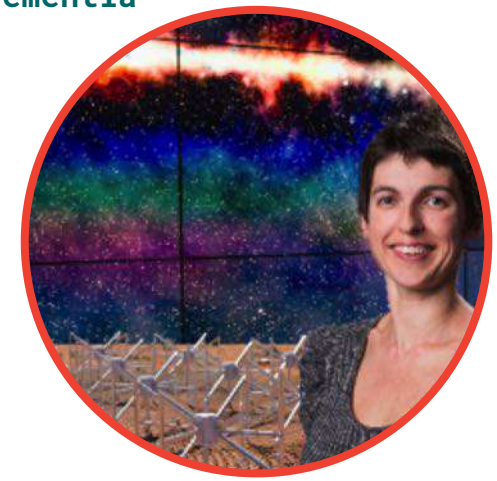
PAWSEY
supercomputing centre

Accelerating Scientific Outcomes

health
Combating Alzheimer's and dementia

190 projects

resources
Protecting Perth's aquifers



agriculture
AI for targeted weed control

energy
Understanding the ocean to improve offshore designs

astronomy
A panoramic view of the universe in colour

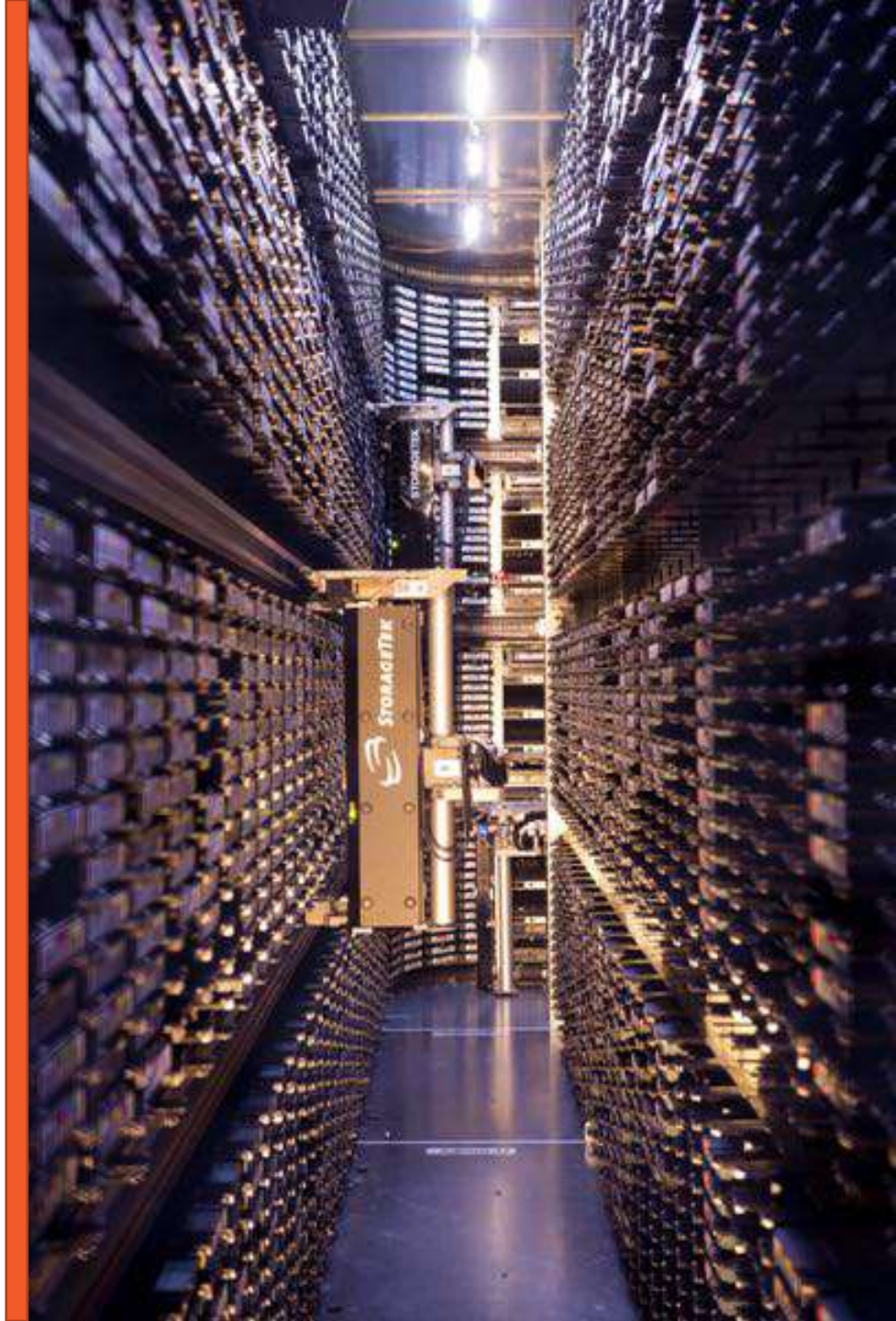
PAWSEY supercomputing centre

Emerging topics in HPC

Machine Learning

Containers

Data management



PAWSEY supercomputing centre

Find out more

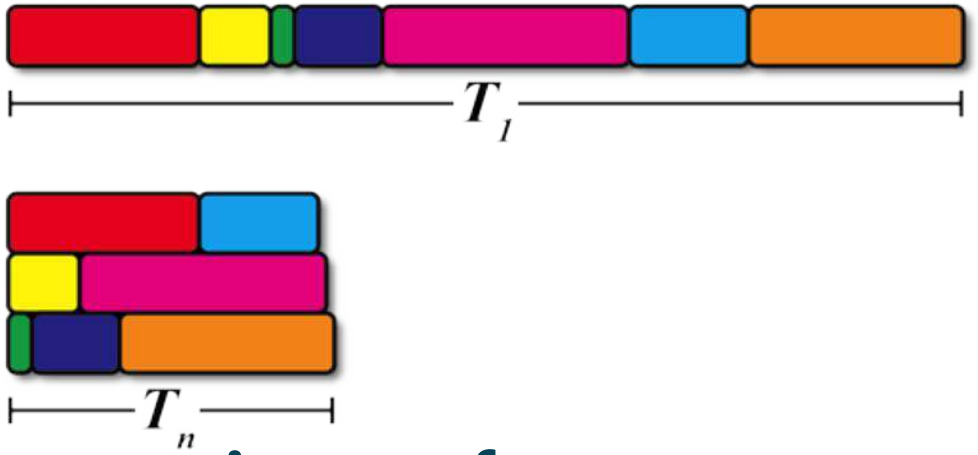
- Pawsey Website (www.pawsey.org.au)
- Pawsey Friends mailing list
- Pawsey Twitter feed (@PawseyCentre)
- User Support Portal (support.pawsey.org.au)

Basics of Parallel Computing

Parallelism in Workflows

Exploiting parallelism in a workflow allows us to

- **get results faster, and**
- **break up big problems into manageable sizes.**



A modern supercomputer is not a fast processor. It is many processors working together in parallel.

Baking a Cake

Stage ingredients

Measure ingredients

Mix ingredients

Preheat oven

Serial

Bake and cool cake

Parallel

Mix icing

Ice cake

PAWSEY supercomputing centre

Levels of Parallelism

Coarse-grained parallelism (high level)

- **Different people baking cakes in their own kitchens.**
- **Preheating oven while mixing ingredients.**

Greater autonomy, can scale to large problems and many helpers.

Fine-grained parallelism (low level)

- **Spooning mixture into cupcake tray.**
- **Measuring ingredients.**

Higher coordination requirement. Difficult to get many people to help on a single cupcake tray.

How many helpers?

What is *your* goal? – high throughput, to do one job fast, or solve a grand-challenge problem?

High Throughput:

- **For many cakes, get many people to bake independently in their own kitchens – minimal coordination.**
- **Turn it into a production line. Use specialists and teams in some parts.**

Doing one job fast:

- **Experience as well as trial and error will find the optimal number of helpers.**

Supercomputing Architecture

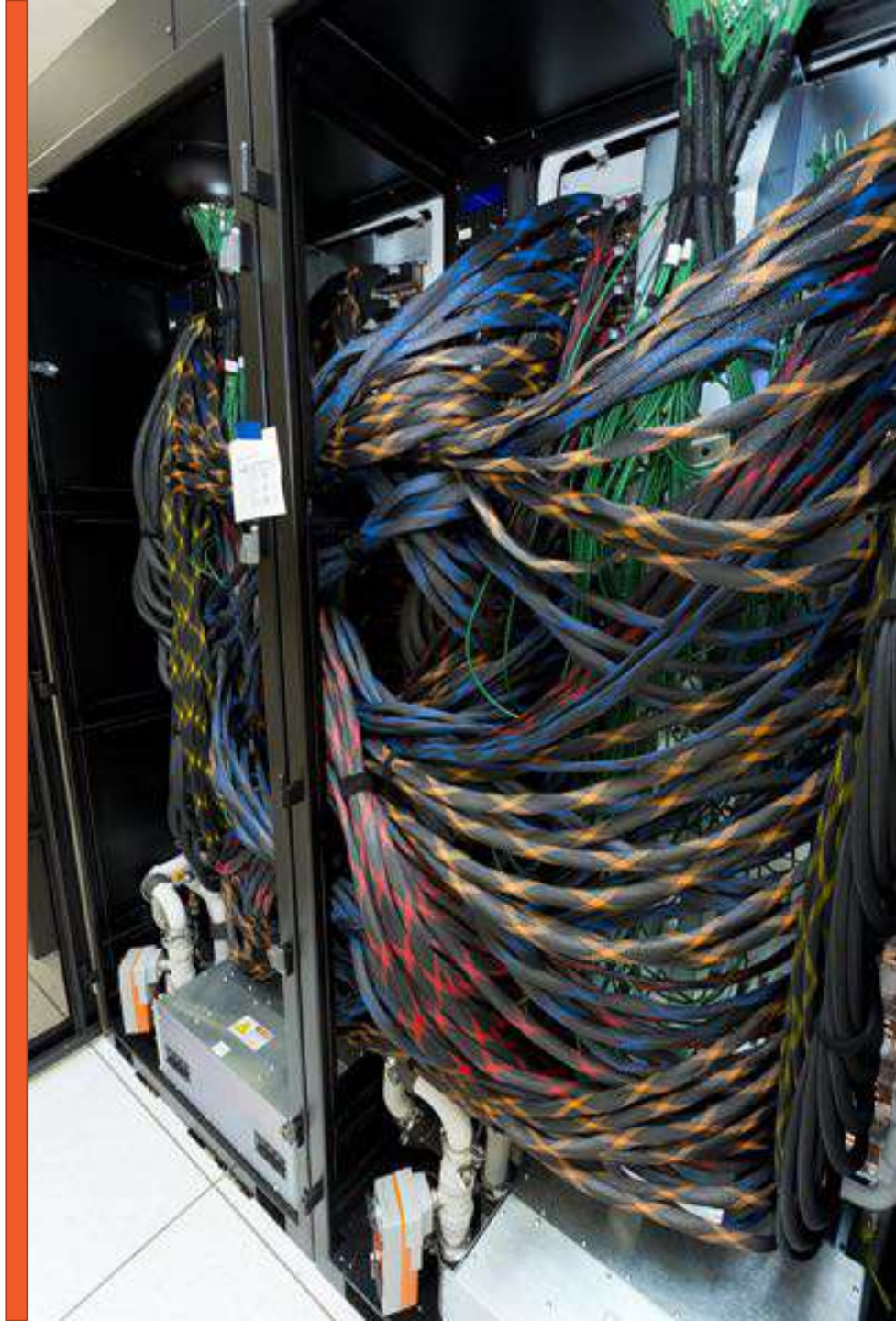
PAWSEY
supercomputing centre

What is a Supercomputer?

Many computers (nodes) closely connected and working together

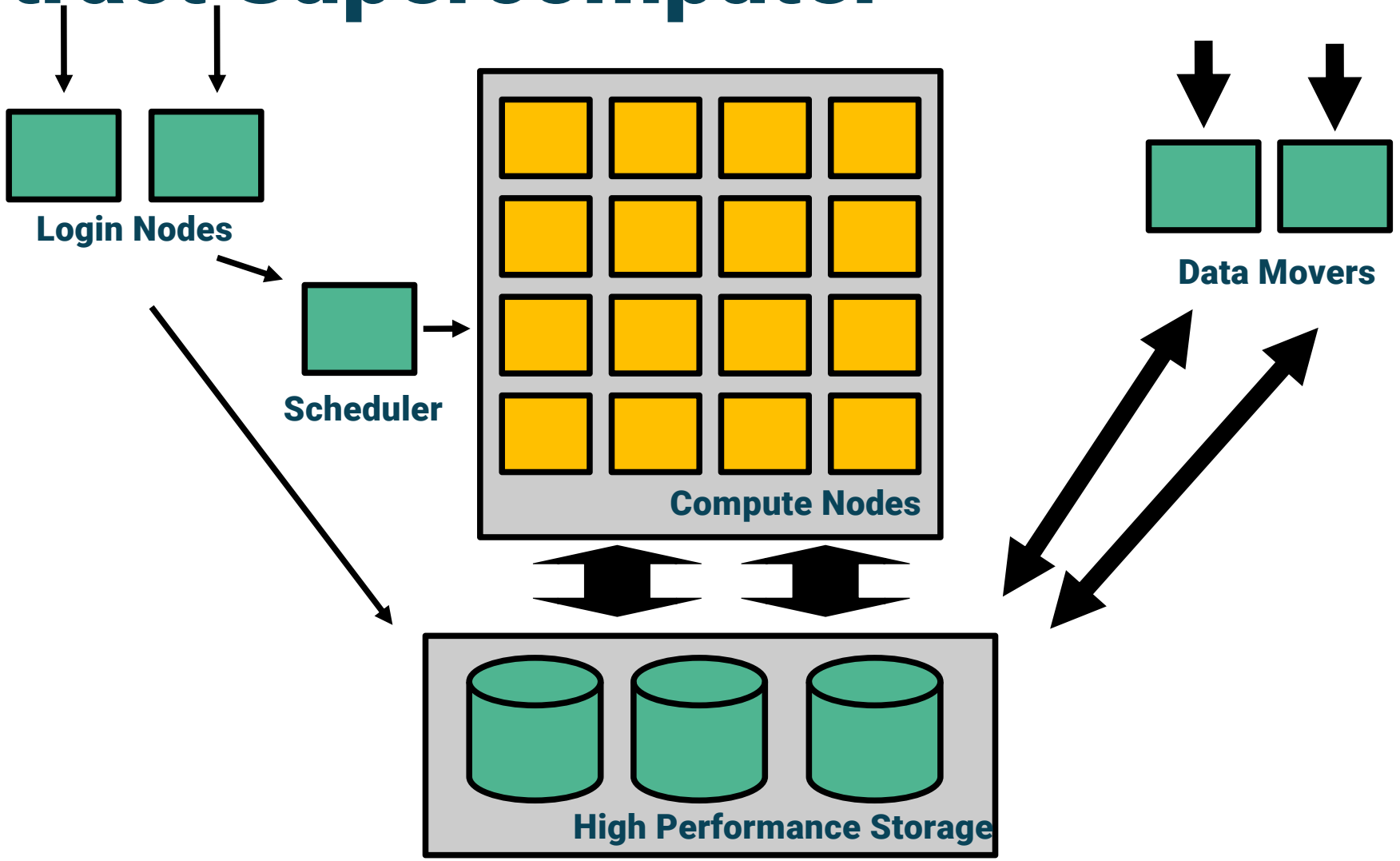
Significantly enhances the capability of computational research

- **reduces runtimes from months to days**
- **enables large scale simulations**
- **allows processing of huge datasets**



PAWSEY supercomputing centre

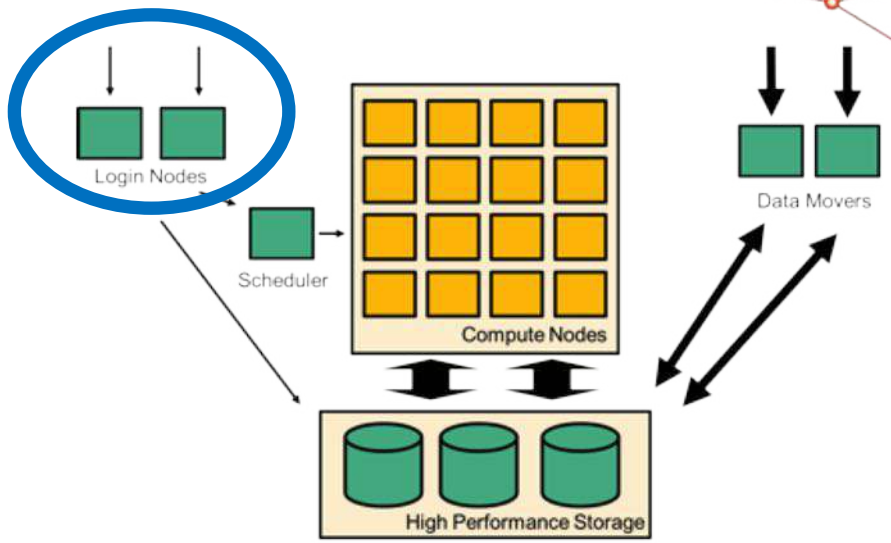
Abstract Supercomputer



PAWSEY
supercomputing centre

Login Nodes

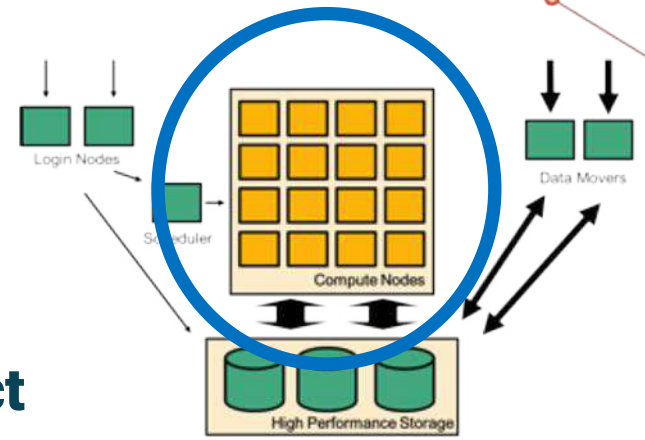
- Remote access to the supercomputer
- Where users should manage workflows
- Many people (~100) share a login node at the same time.



Do not run your programs on the login nodes!

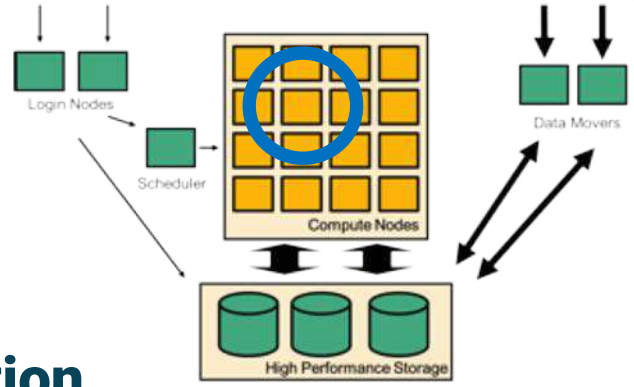
- Use the login nodes to submit jobs to the queue to be executed on the compute nodes
- Login nodes *can* have different hardware to compute nodes.
  - Some build tests may fail if you try to compile on login nodes.

Compute Nodes



- Programs should run on the compute nodes.
- Access is provided via the scheduler
- Compute nodes have a fast interconnect that allows them to communicate with each other
- Jobs can span multiple compute nodes
- Individual nodes are not that different in performance from a workstation
- Parallelism across compute nodes is how significant performance improvements are achieved.

Inside a Compute Node

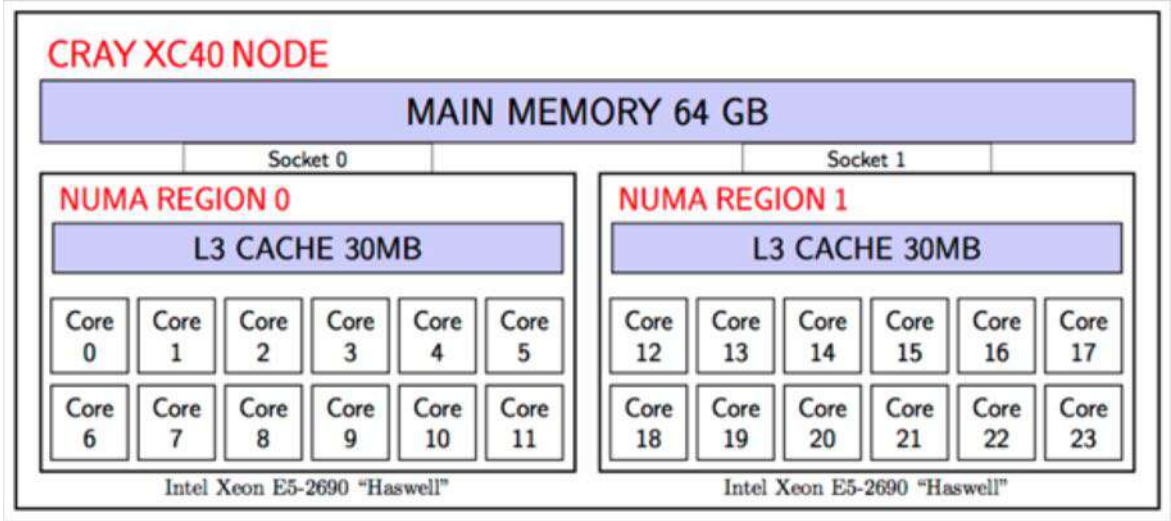


Each compute node has one or more CPUs:

- **Each CPU has multiple cores**
- **Each CPU has memory attached to it**

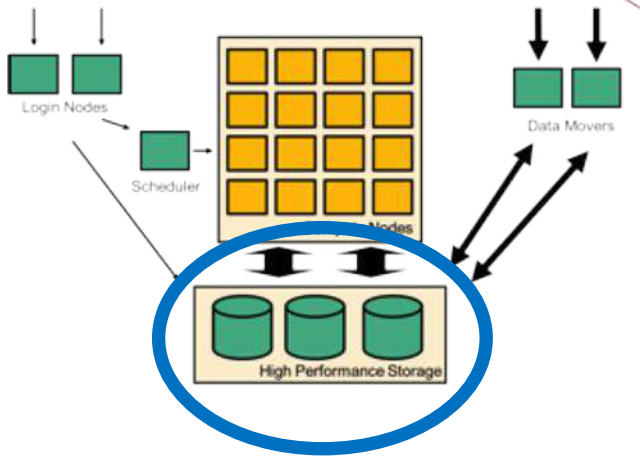
Each node has an external network connection

Some systems have accelerators (e.g. GPUs)



PAWSEY supercomputing centre

High Performance Storage

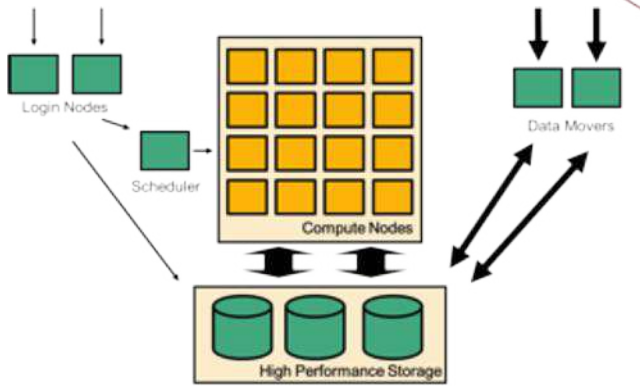


Fast storage "inside" the supercomputer

- **Temporary working area**
- **Might have local node storage**
  - **Not shared with other users**
- **Usually have global storage**
  - **All nodes can access the filesystems**
  - **Either directly connected to the interconnect, or via router nodes**
  - **The storage is shared. Multiple simultaneous users on different nodes will reduce performance**

Data Mover Nodes

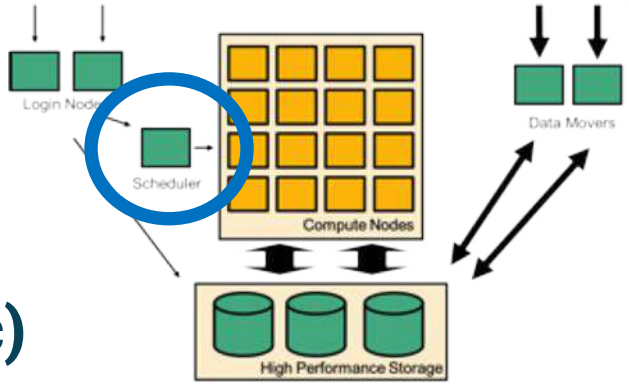
Externally connected servers that are dedicated to moving data to and from the high performance filesystems.



- **Data movers are shared, but most users will not notice.**
- **Performance depends on the other end, distance, encryption algorithms, and other concurrent transfers.**
- **Data movers see all the global filesystems**

`hpc-data.pawsey.org.au`

Scheduler



Submits jobs into the compute nodes

- **Jobs organized in a queue**
- **Scheduler determines place in queue (dynamic)**

At Pawsey we use SLURM

- **Other schedulers used at different sites (LoadLevler, PBS, etc.)**

As a user you interact with the queues

The scheduler runs jobs on the compute nodes on your behalf

Command line SSH

Within a terminal window, type:

```
ssh username@galaxy.pawsey.org.au
```

- **Common terminal programs:**
  - **Windows, use MobaXterm (download)**
  - **Linux, use xterm (preinstalled)**
  - **OS X, use Terminal (preinstalled) or xterm (download)**

Account Security

- SSH uses fingerprints to identify computers … so you don't give your password to someone pretending to be the remote host
- We recommend that you set up SSH key authentication to increase the security of your account:

https://support.pawsey.org.au/documentation/display/US/Logging+in+with+SSH+keys

- Do not share your account with others, this violates the conditions of use (have the project leader add them to the project)
- Please do not provide your password in help desk tickets, we never ask for your password via email as it is not secure.

Graphical Interfaces

- Remote graphical interface for some tasks available
  - Uses X11 forwarding
  - Graphical debuggers, text editors, remote visualisation
  - Low performance over long distances
- Very easy to set up for Linux, Mac and MobaXterm clients Add -X flag to ssh

```
ssh -X username@galaxy.pawsey.org.au
```

- X11 generally not recommended
- For higher performance remote visualisation, use FastX:

remotevis.pawsey.org.au

Common login problems

- Forgot password
  - Self service reset https://support.pawsey.org.au/password-reset/
- Scheduled maintenance
  - Check your email or https://support.pawsey.org.au/documentation/display/US/Maintenance+and+Incidents
- Blacklisted IP due to too many failed login attempts
  - This is a security precaution
  - Email the helpdesk with your username and the machine you are attempting to log in to

Exercise: Logging In

- **Try logging in with your course account:**

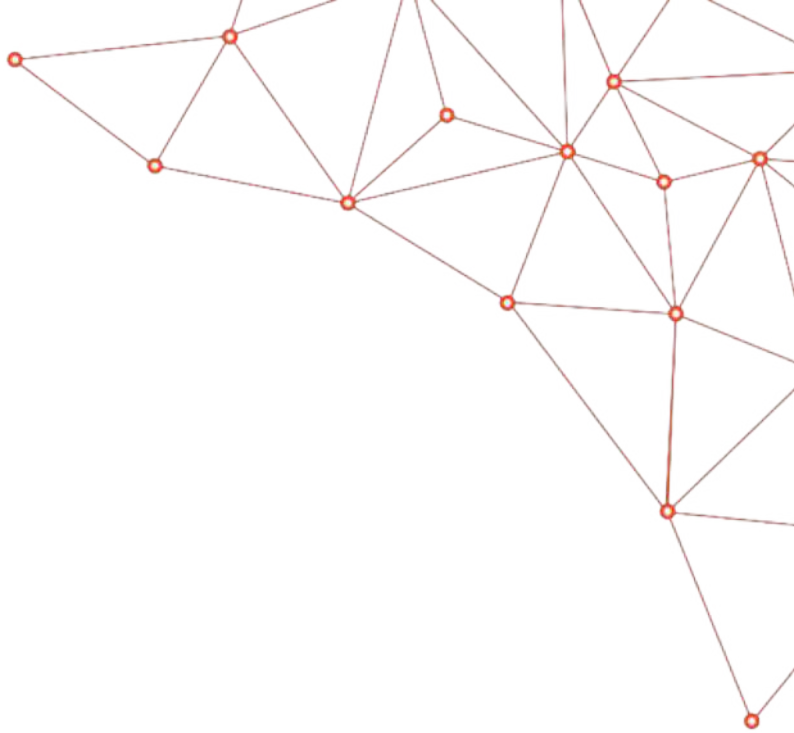
```
ssh couXXX@galaxy.pawsey.org.au
```

- **Tips**
  - **Password won't be visible (security measure)**
  - **Double-check hostname and username**

PAWSEY supercomputing centre

User Environment

Software Stack



Various software is provided to support workflows on the systems:

- Operating System (SLES or CLE)
- Compilers (e.g. Intel, GCC, Cray, PGI)
- Debuggers and Profilers (e.g. MAP, DDT)
- Performant mathematical libraries (e.g. MKL, Lapack, Petsc)
- Parallel programming libraries (e.g. MPI)
- File format libraries for parallel IO (e.g. HDF5)

Project groups are expected to manage the installation of their own software stack

Applications that are widely used by a large number of groups may also be provided

All of the above software is not immediately available as soon as you log in

Modules

To prevent conflicts between software names and versions, applications and libraries are not installed in the standard directory locations.

Modules modify the environment to easily locate software, libraries, documentation, or particular versions of the software

```
module load astropy
```

Module Commands

Command	Description
`module avail`	Show available modules
`module list`	List loaded modules
`module load` ***`modulename`***	Load a module into the current environment
`module unload` ***`modulename`***	Unload a module from the environment
`module swap` ***`module1 module2`***	Swap a loaded module with another
`module show` ***`modulename`***	Give help for a particular module
`module help`	Show module specific help

Module Prerequisites

Some modules have prerequisites and order is important

Most modules depend on an architecture and compiler

System	Architecture modules	Compiler modules
Magnus, Galaxy	cray-sandybridge cray-ivybridge cray-haswell	PrgEnv-cray PrgEnv-gnu PrgEnv-intel
Zeus	sandybridge broadwell	gcc intel pgi

Recommended Module Order

Example Load Order:

1) **CPU Architecture**
2) **Compiler**
3) **Compiler Version**
4) **MPI (if needed)**
5) **CUDA (if needed)**
6) **Python version**
7) **All other libraries**

Module Prerequisites

On Crays, switch to the desired programming environment first:

```
module swap PrgEnv-cray PrgEnv-gnu
```

- **Some modules can only be compiled for particular combinations of architectures and compilers**
- **Loading these modules with the wrong prerequisites will generate a conflict error:**

```
> module load casacore

casacore/2.3.0(72):ERROR:150: Module
'casacore/2.3.0' conflicts with the currently
loaded module(s) 'PrgEnv-cray/6.0.4'

casacore/2.3.0(72):ERROR:102: Tcl command
execution failed: conflict PrgEnv-cray/6.0.4
```

```
> module show casacore

Compiled with gcc/4.9.3 under PrgEnv-gnu/6.0.4

Compiled with gcc/5.3.0 under PrgEnv-gnu/6.0.4

Compiled with gcc/6.1.0 under PrgEnv-gnu/6.0.4

Compiled with gcc/7.2.0 under PrgEnv-gnu/6.0.4

Compiled for craype-sandybridge

Compiled for craype-ivybridge

Compiled for craype-haswell

setenv          CRAYOS_VERSION 6.0.4

conflict        PrgEnv-intel/6.0.4

conflict        PrgEnv-cray/6.0.4

conflict        craype-broadwell

module          load cray-mpich

module          load cfitsio

module          load wcslib
```

System vs user modules

Pawsey module paths are loaded into user environment by default

- **module avail**

Users can use group/user-installed modules as well

```
> module use /group/<project-name>/<user-id>/<pawsey_os>/modulefiles
```

Example

```
> module use /group/mwa/software/modulefiles
> module load wsclean
```

NOTE: User/group modules will appear first in module search/load operations

Filesystems

PAWSEY
supercomputing centre

Lustre Architecture

Clients {

Infiniband Interconnect

MDT - Meta-Data Server

open(), close(), stat(), unlink()

MDS

OSS

OSS

OSS

OSS

OSS - Object Storage Server

write(), read(), seek(), flock()

1 2

1 5

2 6

3 7

4 8

MDT - Meta-Data Target

- Backend disk for the MDS
- Contain information about a Lustre file system's directory structure and file names, permissions, extended attributes, and file layouts

OST - Object Storage Target

- Backend disks for the OSS
- Contain a number of binary objects representing the data for files in Lustre

PAWSEY supercomputing centre

Filesystems at Pawsey

Filesystem	Type	Size	User quota	Group quota	Purge Policy	Backup	System
/scratch	Lustre	3 PB	-	-	30 days	No	Magnus, Zeus
/group	Lustre	3 PB	-	1 TB	-	Yes	All
/astro	Lustre	2 PB		300 TB*	-	No	Galaxy only
/home	NFS	15 TB	10 GB	-	-	Yes	All

PAWSEY supercomputing centre

Filesystem Best Practices

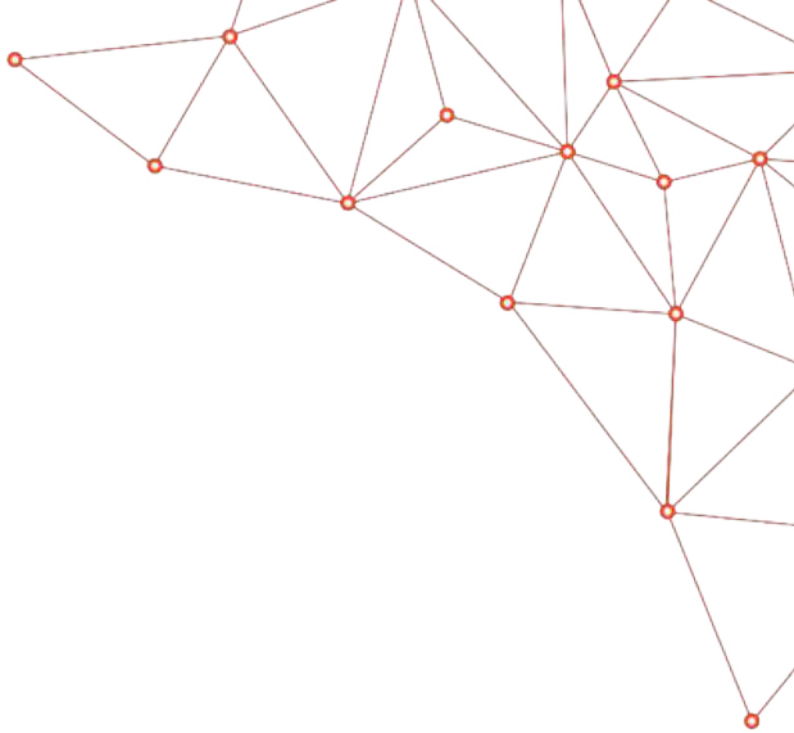
/home

- Don’t run jobs here
- Meant to store user-specific files (bashrc, ssh keys, etc.)

/group

- Medium-term storage (life of the project)
- Store “valuable” data here (e.g., input sets to be reused, final outputs)
- Software builds/modulefiles
- Stuff to share with group

Filesystem Best Practices



/astro

- Run jobs here
- Transient storage
- Be mindful of group quotas (shared resource)

/scratch

- Similar to astro (production jobs)
- 30-day purge policy (continuous)
- Don’t circumvent purge policy
- Don’t store your thesis here (yes….this has happened)

Filesystem Best Practices

Deleting files

- For small number of files (and small filesizes) standard linux command `rm` is fine
- For larger datasets, use `munlink` (see Pawsey Documenation)

```
> find ./directory -type f -print0 | xargs -0 munlink
> find ./directory -depth -type d -empty -delete
```

Avoids generating metadata operations (which is what produces a slowdown)

Similarly, avoid doing `ls` on large file counts (ls has to do a sort)

- **There are flags you can pass to ls to minimise metadata ops.**

Job Scheduling

Scheduling Your Job

Tell the scheduler what resources your calculation needs. (Usually how many nodes and for how long)

Overestimating the time required means it will take longer to find an available slot

Underestimating the time required means the job will get killed

Underestimating memory will cause your program to crash

Interacting with Pawsey Queues

All Pawsey supercomputers (Magnus, Zeus, Zythos and Galaxy) use SLURM to manage queues.

The three essential commands:

```
sbatch jobscript
squeue
scancel jobid
```

You'll get an identifier (i.e., *jobid*) when you sbatch the job:

```
> sbatch jobscript.slurm
Submitted batch job 2315399
```

Querying SLURM Partitions

To list the partitions when logged into a machine:

```
sinfo
```

To get all partitions in all local clusters:

```
sinfo -M all
```

For example:

```
username@magnus-1:~> sinfo
PARTITION AVAIL JOB_SIZE  TIMELIMIT   CPUS  S:C:T   NODES STATE      NODELIST
workq*    up    1-472      12:00:00     40 2:10:2       1 allocated* nid00492
workq*    up    1-472      12:00:00     40 2:10:2     471 allocated  nid00[008-063,072-091,096-123,160-163,200-255,264-319,324-491]
gpuq      up    1-64     1-00:00:00     16  1:8:2      50 allocated  nid00[124-127,132-152,164-188]
gpuq      up    1-64     1-00:00:00     16  1:8:2      14 idle       nid00[092-095,153-159,189-191]
longq     up    1-252    1-00:00:00     40 2:10:2       1 allocated* nid00492
longq     up    1-252    1-00:00:00     40 2:10:2     251 allocated  nid00[324-491,493-575]
```

Querying the Queue

`squeue` displays the status of jobs in the local cluster

```
squeue
squeue -u username
squeue -p debugq
```

```
charris@zeus-1:~> squeue
JOBID    USER     ACCOUNT    PARTITION  NAME           EXEC_HOST      ST  REASON      START_TIME    END_TIME     TIME_LEFT  NODES  PRIORITY
2358518  jzhao    pawsey0149 zythos     SNP_call_zytho zythos         R   None        Ystday 11:56  Thu 11:56    3-01:37:07 1      1016
2358785  askapops askap      copyq      tar-5182       hpc-data3      R   None        09:20:35      Wed 09:20    1-23:01:09 1      3332
2358782  askapops askap      copyq      tar-5181       hpc-data2      R   None        09:05:13      Wed 09:05    1-22:45:47 1      3343
2355496  pbranson pawsey0106 gpuq       piv_RUN19_PROD n/a            PD  Priority    Tomorr 01:53  Wed 01:53    1-00:00:00 2      1349
2355495  pbranson pawsey0106 gpuq       piv_RUN19_PROD n/a            PD  Resources   Tomorr 01:52  Wed 01:52    1-00:00:00 4      1356
2358214  yyuan    pawsey0149 workq      runGet_FQ      n/a            PD  Priority    20:19:00      Tomorr 20:19 1-00:00:00 1      1125
2358033  yyuan    pawsey0149 gpuq       4B_2           n/a            PD  AssocMaxJo  N/A           N/A          1-00:00:00 1      1140
2358709  pbranson pawsey0106 workq      backup_RUN19_P n/a            PD  Dependency  N/A           N/A          1-00:00:00 1      1005
```

Querying the Queue (cont'd)

```
charris@zeus-1:~> squeue
JOBID    USER      ACCOUNT     PARTITION  NAME             EXEC_HOST      ST  REASON      START_TIME   END_TIME     TIME_LEFT
NODES  PRIORITY
2358518  jzhao     pawsey0149  zythos     SNP_call_zytho   zythos         R   None        Ystday 11:56 Thu 11:56    3-01:37:07   1        1016
2358785  askapops  askap       copyq      tar-5182         hpc-data3      R   None        09:20:35     Wed 09:20    1-23:01:09   1        3332
2358782  askapops  askap       copyq      tar-5181         hpc-data2      R   None        09:05:13     Wed 09:05    1-22:45:47   1        3343
2355496  pbranson  pawsey0106  gpuq       piv_RUN19_PROD   n/a            PD  Priority    Tomorr 01:53 Wed 01:53    1-00:00:00   2        1349
2355495  pbranson  pawsey0106  gpuq       piv_RUN19_PROD   n/a            PD  Resources   Tomorr 01:52 Wed 01:52    1-00:00:00   4        1356
2358214  yyuan     pawsey0149  workq      runGet_FQ        n/a  PD  Priority    20:19:00     Tomorr 20:19 1-00:00:00   1        1125
2358033  yyuan     pawsey0149  gpuq       4B_2             n/a            PD  AssocMaxJo  N/A          N/A          1-00:00:00   1        1140
2358709  pbranson  pawsey0106  workq      backup_RUN19_P   n/a            PD  Dependency  N/A          N/A          1-00:00:00   1        1005
```

NAME – job name. Set this if you have lots of jobs.

ST – job state. R=running. PD=pending.

REASON – the reason the job is not running

- Dependency – job must wait for another to complete before it
- Priority – a higher priority job exists
- Resources – the job is waiting for sufficient resources

Individual Job Information

scontrol show job *jobid*

```
charris@magnus-1:~> scontrol show job 2474075
JobId=2474075 JobName=m2BDF2
   UserId=tnguyen(24642) GroupId=tnguyen(24642) MCS_label=N/A
   Priority=7016 Nice=0 Account=pawsey0199 QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=03:13:09 TimeLimit=1-00:00:00 TimeMin=N/A
   SubmitTime=12 Dec 2017 EligibleTime=12 Dec 2017
   StartTime=10:41:04 EndTime=Tomorr 10:41 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=workq AllocNode:Sid=magnus-2:53310
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=nid0[0041-0047,0080-0082,0132-0133,0208-0219,0224-0226,0251-
0253,0278-0279,0284-0289,0310-0312,0319,0324-0332,0344,0349-0350,0377-
0379,0385-0387,0484-0503,0517-0520,0525-0526,0554-0573,0620-0628,0673-
0686,0689-0693,0732,0894-0895,0900-0907,1036-1037,1048-1051,1134-
1138,1202-1203,1295-1296,1379-1380,1443-1446,1530-1534]
   BatchHost=mom1
   NumNodes=171 NumCPUs=4104 NumTasks=171 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=4104,mem=5601960M,node=171
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:1 CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=1365M MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)

Command=/scratch/pawsey0199/tnguyen/run_test_periodicwave/stiff_problem/forMagnus/4thO
rder/accuracy_check/eta_1/PeriodicBCs/BDF2/m2/gpc.sh

WorkDir=/scratch/pawsey0199/tnguyen/run_test_periodicwave/stiff_problem/forMagnus/4thOrd
er/accuracy_check/eta_1/PeriodicBCs/BDF2/m2

StdErr=/scratch/pawsey0199/tnguyen/run_test_periodicwave/stiff_problem/forMagnus/4thOrde
r/accuracy_check/eta_1/PeriodicBCs/BDF2/m2/m2BDF2

   StdIn=/dev/null

StdOut=/scratch/pawsey0199/tnguyen/run_test_periodicwave/stiff_problem/forMagnus/4thOrde
r/accuracy_check/eta_1/PeriodicBCs/BDF2/m2/m2BDF2

   Power=
```

PAWSEY supercomputing centre

Reservations

Nodes can be manually reserved for a certain time by the system administrators.

- **Email the helpdesk to ask for a reservation. Only ask if you cannot work via the standard queues.**
- **For scheduled maintenance - we reserve the whole machine.**
- **For interactive use – debugging a many-node job or for a training course.**
- **A once-off urgent deadline.**

```
[reaper@magnus-2 ~]> sinfo -T
RESV_NAME     STATE           START_TIME             END_TIME     DURATION  NODELIST
courseq      ACTIVE             09:00:00             17:00:00     08:00:00  nid000[16-23]
```

PAWSEY supercomputing centre

Job Request

SLURM needs to know two things from you:

1. **Resource requirement.**
   - **How many nodes and how long you need them for.**

2. **What to run.**
   - **You cannot submit an application directly to SLURM. Instead, SLURM executes on your behalf a list of shell commands.**
   - **In batch mode, SLURM executes a jobscript which contains the commands.**
   - **In interactive mode, type in commands just like when you log in.**
   - **These commands can include launching programs onto the compute nodes assigned for the job.**

Common sbatch directives

```
#SBATCH --job-name=myjob
```
→ makes it easier to find in squeue

```
#SBATCH --account=courses01
```
→ project accounting

```
#SBATCH --nodes=2
```
→ number of nodes

```
#SBATCH --time=00:05:00
```
→ walltime requested

```
#SBATCH --export=NONE
```
→ start with a clean environment

Example Jobscript

```
#!/bin/bash -l
#SBATCH --job-name=myjob
#SBATCH --account=courses01
#SBATCH --nodes=1
#SBATCH --time=00:05:00
#SBATCH --partition=workq
#SBATCH --reservation=courseq
#SBATCH --export=NONE

module load python/2.7.14

#The next line is executed on the compute node
srun --export=all –n 1 python --version
```

SLURM Output

Standard output and standard error from your jobscript are collected by SLURM, and written to a file in the directory you submitted the job from *when the job finishes/dies*.

`slurm-`***`jobid`***`.out`

SLURM has options to modify stdout/stderr writing

- **Renaming files**
- **Specifying location**
- **Splitting stderr/stdout**

Interactive Jobs

If there are no free nodes, you may need to wait while the job is in the queue.

```
ddeeptimahanti@galaxy-1:~> salloc --nodes=1 --reservation=courseq
    salloc: Pending job allocation 2315927
    salloc: job 2315927 queued and waiting for resources
```

It may appear to hang – waiting for resources to become available.

For small interactive jobs on Magnus use the debugq to wait less.

```
    salloc --tasks=1 --time=10:00 -p debugq
```

Queue Best Practices

Get good estimates on wall time

For Galaxy, no need to specify memory

- **You get it all (nodes are exclusive)**

Zeus has shared nodes

- **Need to specify number of cores and/or memory**

Queues are dynamic

- **Priority and position in queue may change**

Project Accounting

PAWSEY
supercomputing centre

Allocations at Pawsey

Merit allocations are awarded typically for 12 months

Merit allocations are divided evenly between the four quarters of the year, to avoid end-of-year congestion. Priorities reset at the start of the quarter for merit allocations

Director share allocations are typically awarded for up to 12 months, or until the time is consumed, and do not reset automatically

The job priority in the queue is affected the following:

- **usage relative to allocation**
- **size of request**
- **length of time in queue**

Project Usage

CPU usage can be checked using the pawseyAccountBalance tool:

```
pawseyAccountBalance -p projectname -u
```

```
charris@magnus-2:~> pawseyAccountBalance -p pawsey0001 -u
Compute Information
------------------
          Project ID     Allocation          Usage     % used
          ----------     ----------          -----     ------
          pawsey0001         250000         124170       49.7
         --mcheeseman                        119573       47.8
           --mshaikh                           2385        1.0
             --maali                           1109        0.4
          --bskjerven                           552        0.2
     --ddeeptimahanti                           292        0.1
```

PAWSEY supercomputing centre

Filesystem Usage

pawseyAccountBalance can also be used to query disk usage:

```
pawseyAccountBalance -p projectname -u
```

```
charris@magnus-2:~> pawseyAccountBalance -p pawsey0001 –storage

Storage Information
-------------------
/group usage for pawsey0001, used = 6.21 TiB, quota = 15.00 TiB
/astro usage for pawsey0001, used = 5.29 TiB, quota = 0.00 bytes
```

Lustre Tools

Can also use Lustre command-line tools to get disk usage:

```
lfs quota -g projectname /astro
```

```
lfs quota -g projectname /group
```

And for home (not a Lustre FS):

```
quota -s -f /home
```

PAWSEY
supercomputing centre

Job Information

The sacct tool provides high-level information on the jobs that have been run:

```
sacct
```

There are many arguments, some commonly used options are:

-a	**display jobs for all users, not just the current user**
-A projectname	**display jobs from this project account**
-S yyyy-mm-ddThh:mm:ss	**display jobs after this start time**
-E yyyy-mm-ddThh:mm:ss	**display jobs before this end time**
-X	**Ignore job steps (i.e. srun lines)**

Job Information (cont.)

```
charris@magnus-1:~> sacct -a -A pawsey0001 -S 2017-12-01 -E 2017-12-02 -X

       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
2461157            bash     debugq pawsey0001         24  COMPLETED      0:0
2461543       bubble512     debugq pawsey0001         24     FAILED      1:0
2461932            bash      workq pawsey0001         24     FAILED      2:0
2462029            bash      workq pawsey0001         24     FAILED    127:0
2462472            bash     debugq pawsey0001         24  COMPLETED      0:0
2462527      jobscript+      workq pawsey0001        960  COMPLETED      0:0
```

Calculating storage quota

Files have 2 ownership categories: User and Group

```
> ls -alh test.slm
-rw-r--r-- 1 bskjerven pawsey0001 283 Jun  9 13:17 test.slm
```

Group quota is based on group ownership of a file (e.g. pawsey0001 above)

Users are also a valid group:

```
> ls -alh $HOME/test.slm
-rw-r--r-- 1 bskjerven bskjerven 120 Sep 25 11:10 /home/bskjerven/test.slm
```

Need to be aware when moving/copying files…can generate ”disk quota exceeded errors”

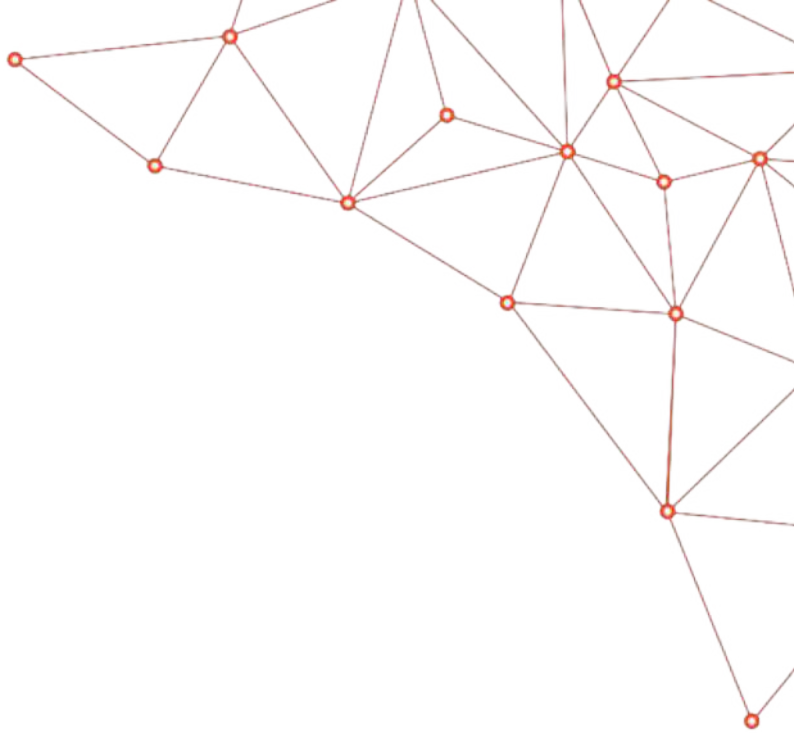
Script available to fix permissions:

```
fix.group.permission.sh projectname
```

Data Transfer

PAWSEY
supercomputing centre

Data Transfer Nodes



All transfers handled via secure copies

- **scp, rsync, etc.**

Interactive use on login nodes is discouraged

- **Small transfers may be okay**

Dedicated servers for transferring large amounts of data

/home, /scratch and /group visible

Externally visible scp

Supercomputer	Hostname
Magnus / Zeus	hpc-data.pawsey.org.au
Galaxy	hpc-data.pawsey.org.au

Data Transfer Nodes

For file transfers, run scp from the remote system using the data transfer nodes.

For example, to copy a file to Pawsey:

```
scp filename username@hpc-data.pawsey.org.au:/group/projectname/username
```

And to copy a file from Pawsey:

```
scp username@hpc-data.pawsey.org.au:/group/projectname/username/filename $PWD
```

Use the same username and password as a normal ssh login.

SCP clients available (Filezilla, WinSCP, etc.), but be aware of things like file permissions and group ownership

copyq

- **Batch job access to data transfer nodes**
- **“copyq” partition**
- **Located on Zeus**
  - **Available to all Pawsey machines**
- **Serial job**
  - **No srun needed**

```
#!/bin/bash –login

#SBATCH --partition=copyq
#SBATCH --cluster=zeus
#SBATCH --ntasks=1
#SBATCH --account=[user-account]
#SBATCH --time=06:00:00
#SBATCH --export=NONE

# stage data
module load python
python ./data-mover.py
```

Advanced Jobscripts

PAWSEY
supercomputing centre

Job Arrays

- A mechanism for submitting collections of jobs
- Running the same program on many different data sets
- More efficient than submitting lots of individual jobs
- User defined range of elements
  - 0,1,2,3
  - 0-9
  - 0-9,20-29

```
#SBATCH --array=<indexes>
```

- Maximum number of elements is 1000
- Identify which index:

```
$SLURM_ARRAY_TASK_ID
```

- Overall job:

```
$SLURM_ARRAY_JOB_ID
```

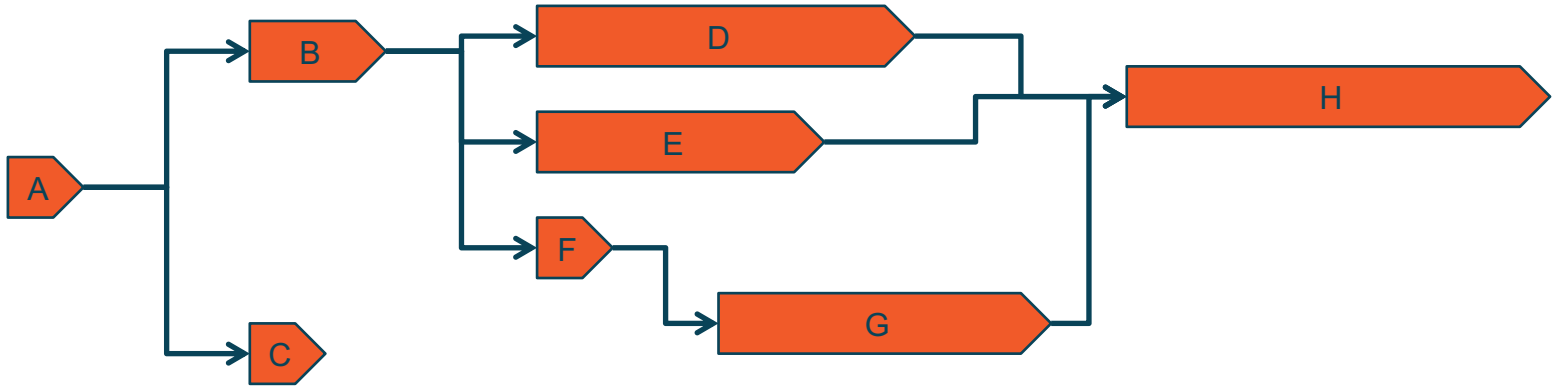
Example: Job Arrays

```
#!/bin/bash --login

#SBATCH --array=8,16,32
#SBATCH --output=array-%j.out
#SBATCH --nodes=1
#SBATCH --time=00:01:00
#SBATCH --account=pawsey0001
#SBATCH --export=NONE

time srun -n 24 --export=all ./darts-mpi $SLURM_ARRAY_TASK_ID
```

Job dependencies

- Useful tool for creating advanced workflows
- Supported between jobs, job arrays, array elements
- Not between jobs on different clusters

#SBATCH --dependency=type:jobid,...



PAWSEY supercomputing centre

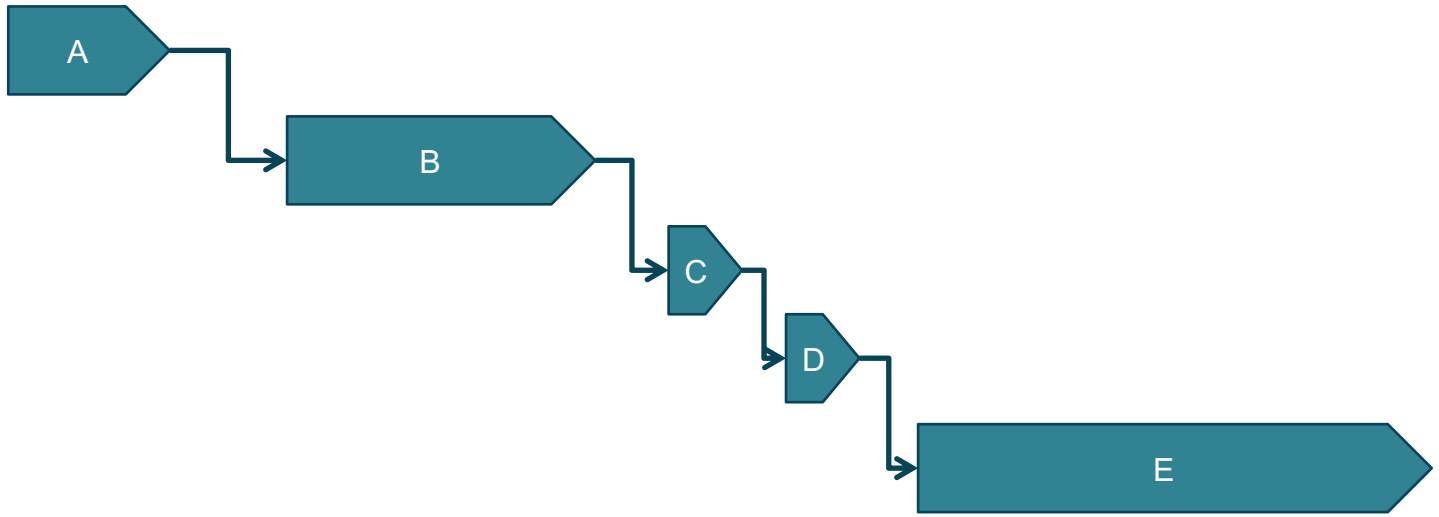
Job dependencies

Dependency List	Description
`after:jobid`	Begin after the listed job has begun execution
`afterany:jobid`	Begin after the listed job has terminated
`afternotok:jobid`	Begin after the listed job has terminated in a failed state
`afterok:jobid`	Begin after the listed job has successfully executed
`singleton`	Begin after any job of the same job name and user has terminated

- **Multiple jobids allowed, eg jobid:jobid**
- **Job array elements referenced as jobid_index**
- **Jobs that are requeued after failure treated the same**

Chaining Jobs

- **Submit the next job from within a batch job at the start or the end of the job**
- **Useful when running jobs across clusters**



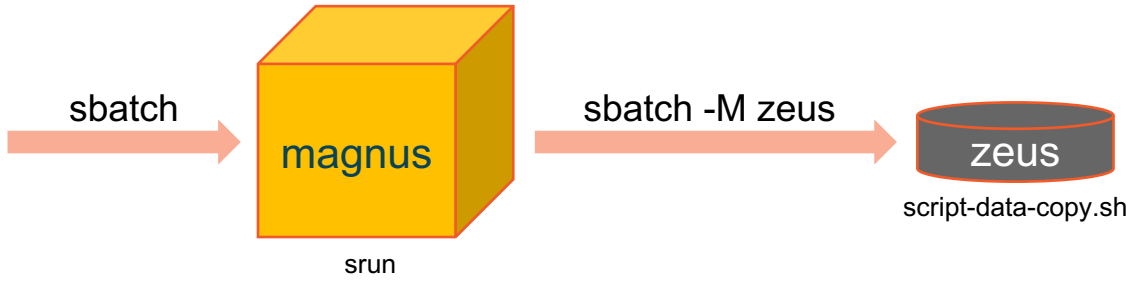
Example: Chaining Jobs

```
#!/bin/bash -l
#SBATCH --account=courses01
#SBATCH --nodes=1
#SBATCH --time=00:05:00
#SBATCH --export=NONE

: ${job_number:="1"}
my_job_number=${job_number}
job_number_max=5

echo "running job ${SLURM_JOB_ID}"
```

```
if [[ ${job_number} -lt ${job_number_max} ]]
then
  (( job_number++ ))
  next_jobid=$(sbatch --export=job_number=${job_number} -d
          afterok:${SLURM_JOB_ID} job.slurm | awk '{print $4}')
  echo "submitted ${next_jobid}"
fi

echo "doing some computations for " ${my_job_number} " seconds"
sleep ${my_job_number}
echo "${SLURM_JOB_ID} done”
```

Example: Data staging

```
#!/bin/bash --login
#SBATCH --partition=workq
#SBATCH --ntasks=1
#SBATCH --account=pawsey0001
#SBATCH --time=00:05:00
#SBATCH --export=NONE

# run simulation
export OMP_NUM_THREADS=24
srun --export=all -n 1 -c 24 hostname


# transfer results
sbatch -M zeus --partition=copyq script-data-copy.sh
```



User Documentation:

portal.pawsey.org.au

Helpdesk:

help@pawsey.org.au

PAWSEY
supercomputing centre

Questions?

Acknowledgements: The Pawsey Supercomputing Centre is supported by $90 million funding as part of the Australian Government's measures to support national research infrastructure under the National Collaborative Research Infrastructure Strategy and related programs through the Department of Education. The Centre would also like to acknowledge the support provided by the Western Australian Government and its Partner organisations.

www.pawsey.org.au

PAWSEY
supercomputing centre