HPC

Introduction to Unix and HPC

What is HPC?

- HPC, or high-performance computing, refers to the application of supercomputers or clusters of computers to computational problems that typically arise through scientific inquiry.
- HPC is useful when a computational problem is either **too large** to solve on a conventional laptop or workstation (because it requires too much memory or disk space) or would **take too long** (because the algorithm is complex, the dataset is large, or data access is slow).

Parallelism on HPC

- HPC systems often derive their computational power by exploiting parallelism
- Programs for HPC systems must be split up into many smaller "programs" called threads, corresponding to each core.
- HPC systems can offer parallelism at a much larger scale, with 100's or 1000's, or (soon) even millions of tasks running concurrently.
- Writing parallel software can be challenging, and many existing software packages do not already support parallelism and may require a substantial investment in research and development.

Reasons to use HPC

- You have a program that can be recompiled or reconfigured to use optimized numerical libraries that are available on HPC systems but not on your own system.
- You have a "parallel" problem, e.g. you have a single application that needs to be rerun many times with different parameters.
- You have a serial application that you would like to run faster
- You have an application that has already been designed with parallelism
- You have an I/O-intensive application that will benefit from HPC high-performance file system

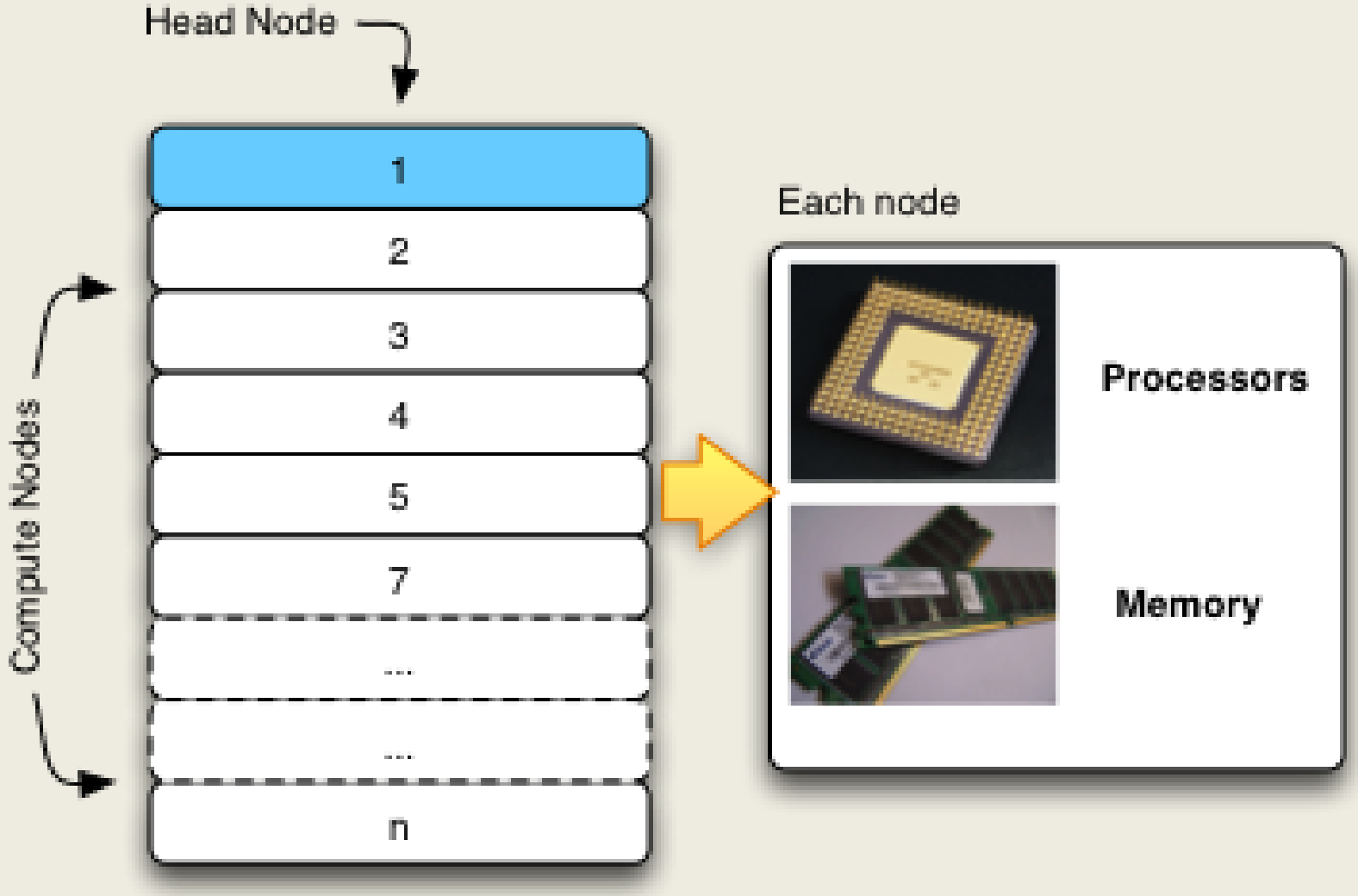
HPC machines

System	Memory Architecture	Cores	Nodes	Memory
Octane (training)	Distributed	48	3	48GB
Orange	**Distributed**	**1,600**	**100**	**8TB**
NCI - current	**Distributed**	**11,936**	**1492**	**37TB**
NCI - Q1 2013	**Distributed**	**57,472**	**3592**	**158TB**

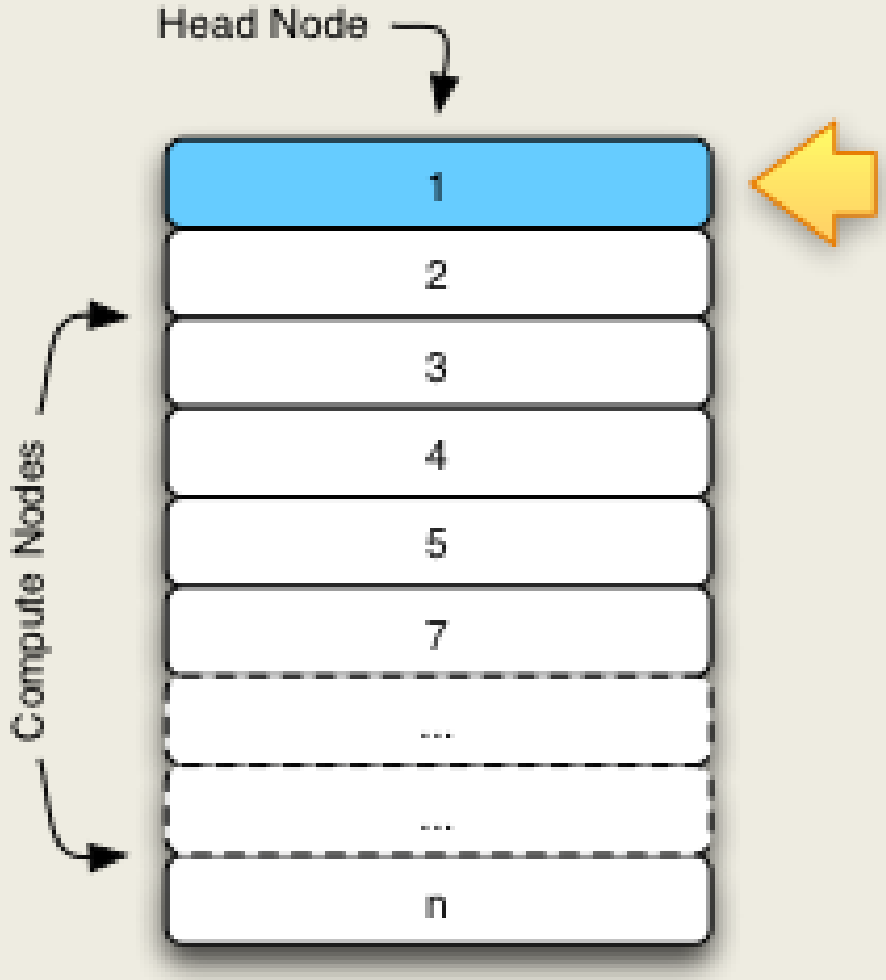
The typical HPC workflow

- In HPC we talk about **jobs**, these are simply commands we wish to run.
- They are generally time consuming and resource intensive.
- Jobs are typically run **non-interactively**, but can also be run interactively
- We add our jobs to a **queue**.
- When the machine has free resources the jobs run.
- Once jobs have completed, we can inspect their output.

The HPC "Cluster"

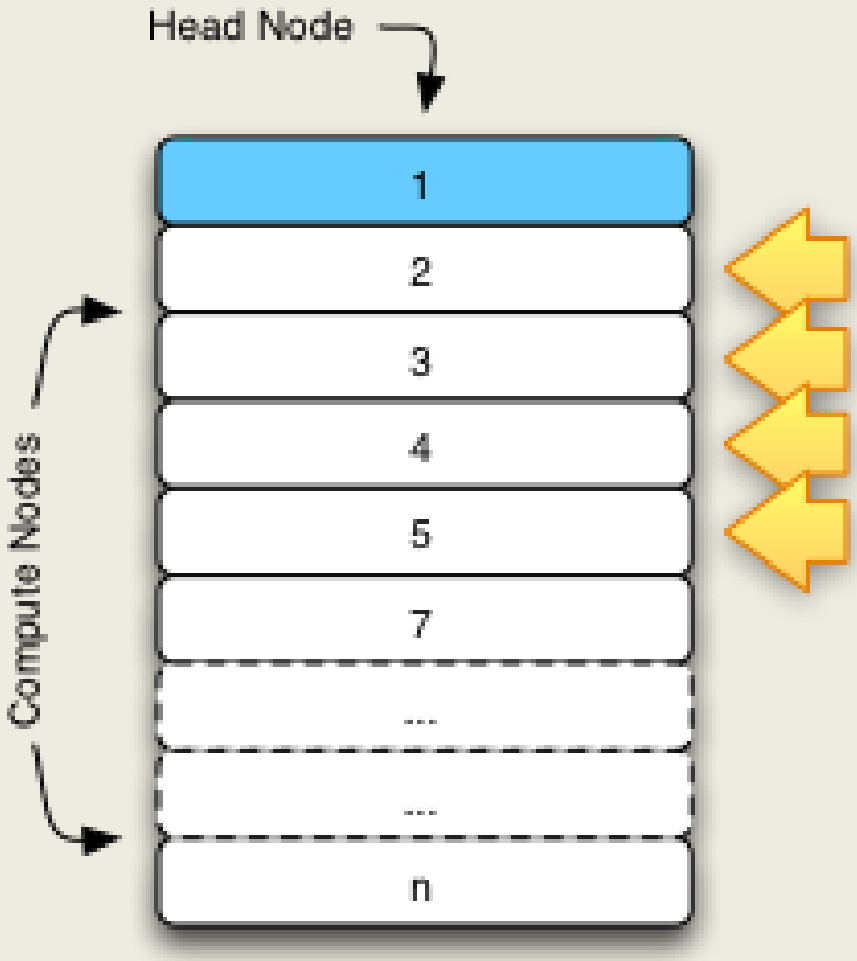


The Head Node



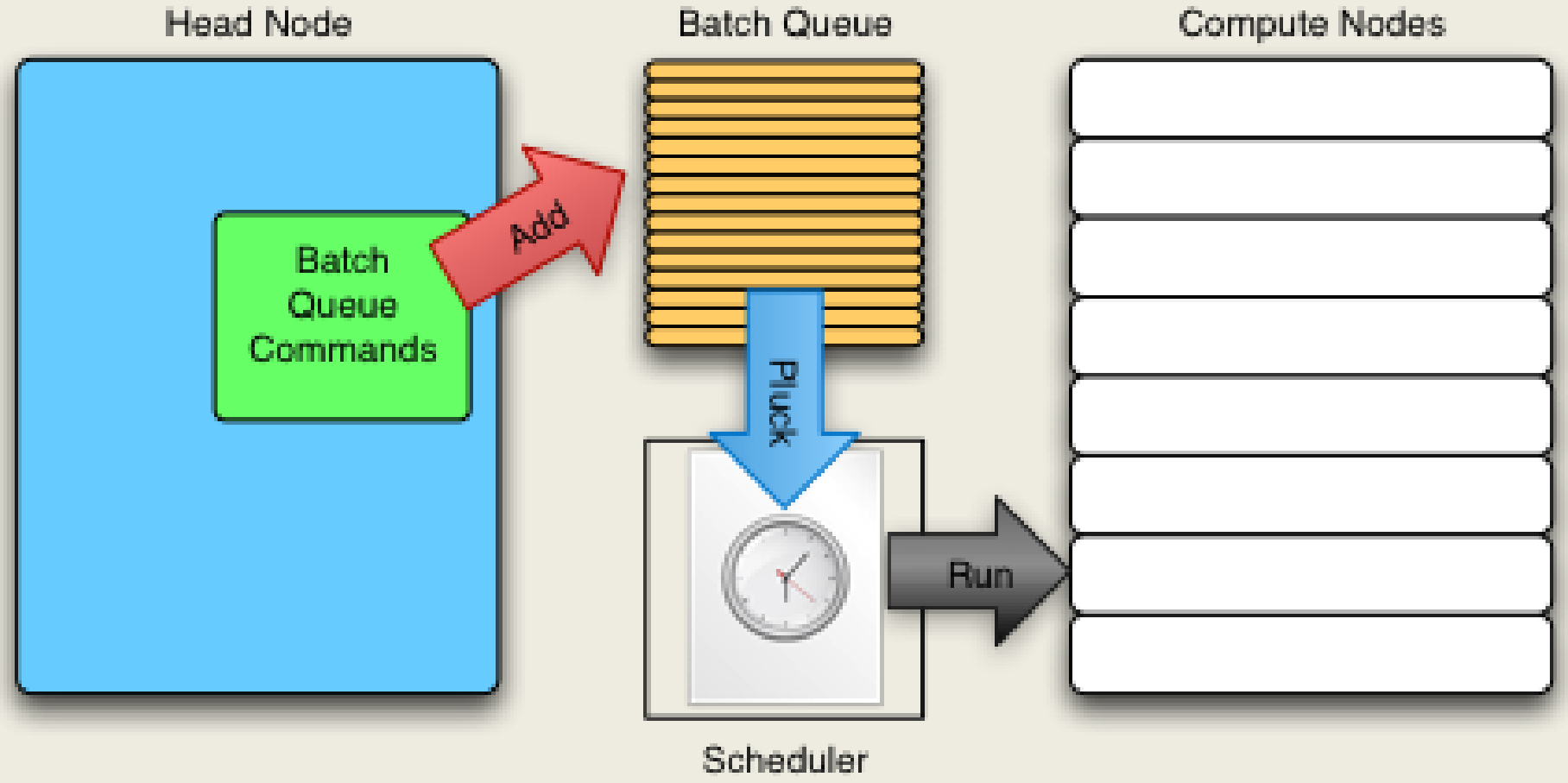
- Interactive programs
- SSH sessions
- Testing
- Compiling
- Queuing jobs

Compute Nodes

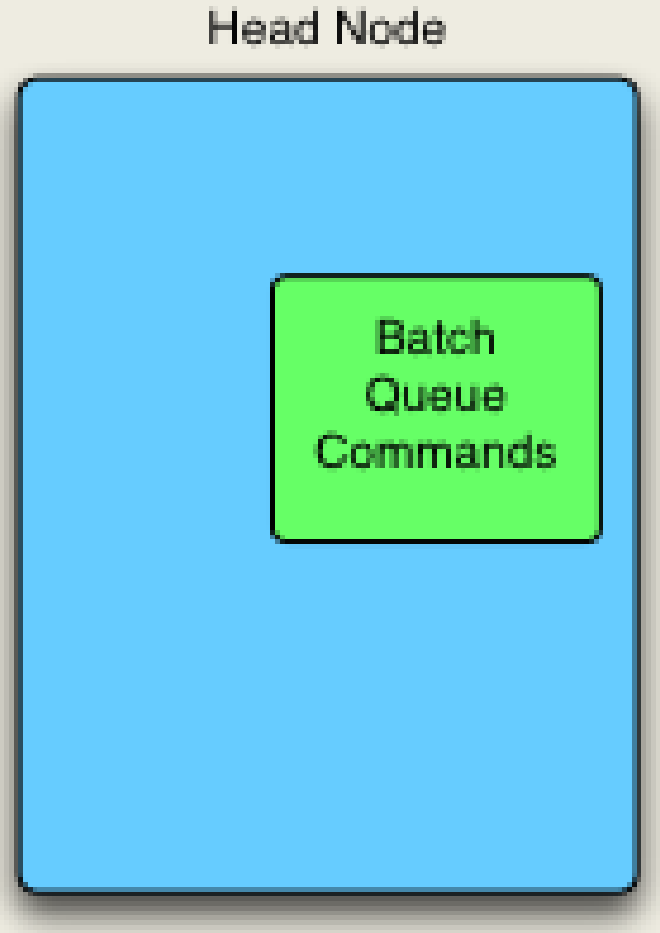


- These run your jobs
- Managed by the **scheduler**
- Typically you will not interact with the nodes directly (some users may need to)

The Batch Queuing System



Batch Queuing component



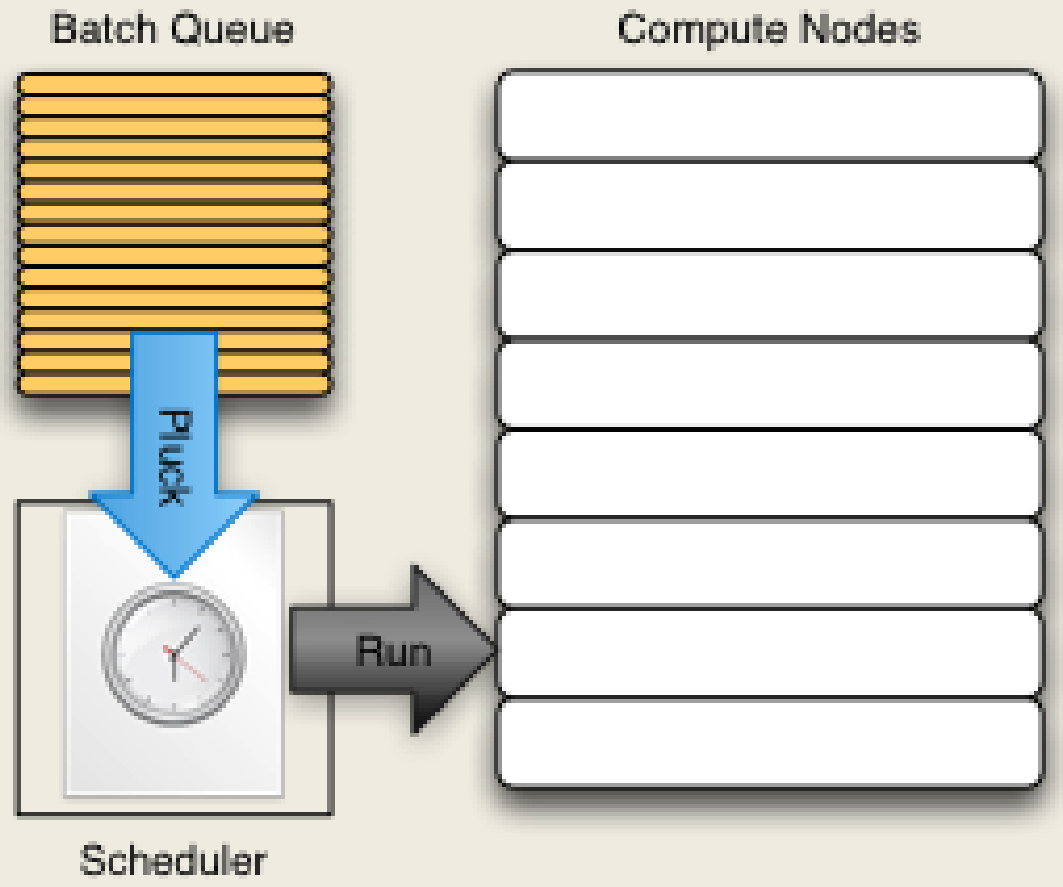
- The batch system is a normal program
- Lets you add and remove jobs from the queue and monitor the queue
- Script/command line driven

Queuing Systems

- **Portable Batch System** (PBS) is the name of computer software that performs job scheduling. Its primary task is to **allocate computational tasks, i.e., batch jobs, among the available computing resources**.
- The following versions of PBS are currently available:
  - OpenPBS
  - TORQUE
  - PSB Professional (PBS Pro)
- **NCI uses: ANU PBS**
- **Orange & Octane use: PBS Pro**

The Scheduler component

- Allocates jobs to compute nodes
- Optimizes usage of resources
- "Optimize" can mean many things
- Non-trivial
- Never interact with directly



PBSPro Batch System

In order to use the batch system productively, we need to know how to perform three actions:

- Add a job to the queue
- Remove a job from the queue
- See where our job is in the queue

Command	Description
qsub <job-script>	Submit a job (add to queue) Returns a <job-number>
qdel <job-number>	Delete job (remove from the queue)
qstat <job-number>	Monitor jobs
qalter <job-number>	Modifies the attributes of the job or jobs

Monitoring the queue

Command	Description
qstat -a	List all jobs in the queue
qstat -u <username>	List all jobs of a particular user
qstat -f <job-number>	Show detailed information about a job

Exercise 1: Monitoring the queue with qstat

Add a job to the queue

- To add a job to the queue, we write a **job script**.
- The job script is a simply a script.
- It has some special comments that pass information to PBSPro.
- When we want to queue the job, we pass its filename as a parameter to **qsub**, e.g.
  - **qsub <job-script>**
- The batch queuing system will return a number that uniquely identifies the job.

A sample job script

```
#!/bin/bash -login
# Request resources
# * 10 minutes wall time to run
#PBS -l walltime=00:10:00
# * 2 nodes, 16 cores on each node, 60 GB memory
#PBS -l select=2:ncpus=16:mem=60G
# * Select queue. Here: workq (default)
#PBS -q workq
# Specify a project code (for accounting)
#PBS -W group_list=r99
# Move to directory job was submitted in
cd $PBS_O_WORKDIR
# Specify the job to be done
date
sleep 10
date
```

You've got mail!

```
# Set email address
#PBS -M fred@intersect.org.au
# Send an email when jobs
# begins (b), gets aborted (a)
# and ends (e)
#PBS -m abe
```

Exercise 2: Submitting a sample job

Useful Environment Variables

- These are available in the context of your job script.

Command	Description
PBS_O_WORKDIR	The directory the job was submitted from
PBS_JOBID	The job number given when the job was submitted

Job limits on Orange

- 200 hours of **walltime**
- 64GB of **memory** per standard node. e.g. 256GB for 2 nodes etc.
- 256GB of **memory** per large memory nodes

Priorities of Jobs

1. Number of jobs (fair share)
2. Walltime
3. Resources available to the project

Best strategy

- Submit jobs constantly/daily
- Have about 10-20 jobs in the machine
- Be realistic with walltime
- Don’t ask for resources you don’t need!

Disk Partitions - Orange

/home	
Mounted under:	/home/username
Size:	60GB default
Backed up:	Yes
Speed:	Intermediate disk (SGI Disks)
Life time:	Permanent

Disk Partitions - Orange

/projects/project-name	
Mounted under:	/projects/project-name
Size:	no default size
Backed up:	Yes
Speed:	High speed
Life time:	Till end of the running year - merit allocation period

- There will also be some “repository space” for large datasets, such as bioinformatics databases

Disk Partitions - Orange

/data2	
Mounted under:	/data2 on each node
Size:	Limit of disk - 2TB
Backed up:	No
Speed:	Fastest
Life time:	Job duration

Warning: This partition is shared among users, so can be “filled up” (with other jobs) while your job is running!

Disk Partitions

You can find out more about the partitions on the HPC machine using the **df** command.

Command	Description
df -h	Show disk free space for all partitions in human readable format

You can find out more about current disk usage, using the **du** command.

Command	Description
du -hs .	Show disk usage of current directory in human readable format

Quotas

- There is no quota on scratch disks for performance reasons.
- The quota on /projects/project-name depends on your allocation.
- 60 GB soft limit for /home.
- 80 GB hard limit for /home (30 days).

Next Steps

- Read more about Orange and NCI Facilities
  - http://www.intersect.org.au/hpc-news
  - http://www.intersect.org.au/orange
  - http://www.intersect.org.au/nci_next
  - http://www.intersect.org.au/orange-handbook

Next Steps

- Apply for an account on Orange or NCI
  - Register as **a new user**:
    http://nf.nci.org.au/accounts/forms/user_registration.php
  - Register **a project**:
    https://nf.nci.org.au/accounts/projects_new/APP_form.php
    NB: Pick INTERSECT under partner/scheme on the first page of the project registration form

Conclusion

- In this course we have covered the basics of the Unix command line, transferring data, and the specifics of our HPC machine.
- Please complete our survey!
- Any questions?