Intermediate HPC

Introduction to Unix for HPC
Raijin version

Parallelism on HPC

- HPC systems often derive their computational power by exploiting parallelism
- Programs for HPC systems must be split up into many sub processes
- HPC systems can offer parallelism at a large scale, with 1000's, or even millions of tasks running concurrently.
- Writing parallel software can be challenging, and might not even be possible if dependencies exist.

NOTE: Many tasks cannot be parallelised

What is HPC?

- HPC, or high-performance computing, refers to the application of supercomputers or compute clusters to computational problems
- HPC is useful when a computational problem:
  - **Is 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) or
  - **Are too many** – High Throughput Computing

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.
- HPC applications are already installed on the HPC machines which is a non-trivial task
- You have a "parallel" problem, e.g. you have a single application that needs to be rerun many times with different parameters.
- You have an application that has already been designed with parallelism
- To make use of the large memory/disk available
- Our facilities are reliable and regularly backed up

When not to use HPC?

- You have a single single threaded job which will only run one job at a time (typical of MatLab users)
- You need interactive use.
- You have a lot of data to transfer between your local machine and the HPC on a continuous basis (e.g. per job)
- You need to have a GUI to interact with your program or graphics.

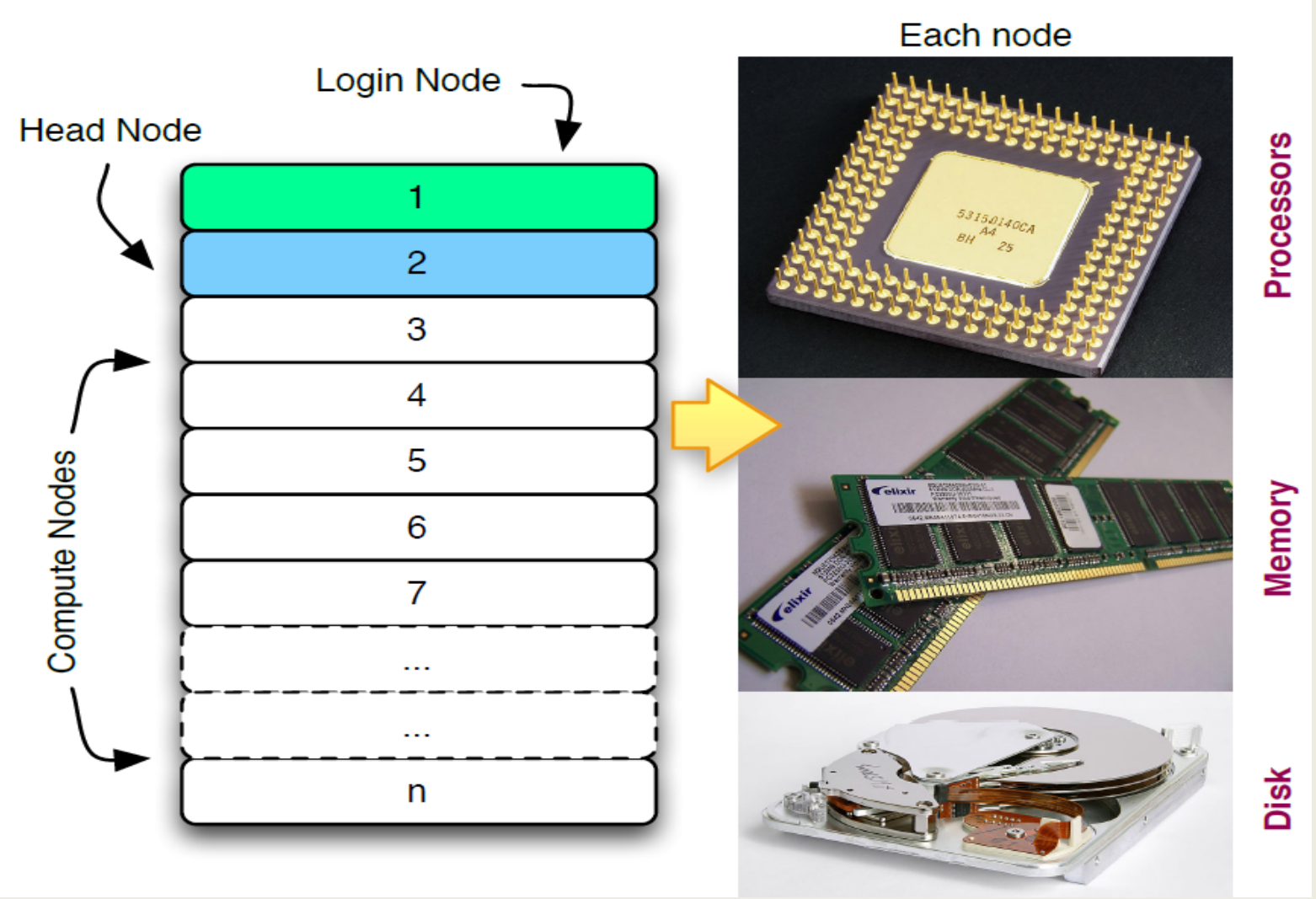
HPC machines

System	Memory Architecture	Cores	Nodes	Memory
Raijin (NCI)	**Distributed**	**84,656**	**4416**	**300 TB**

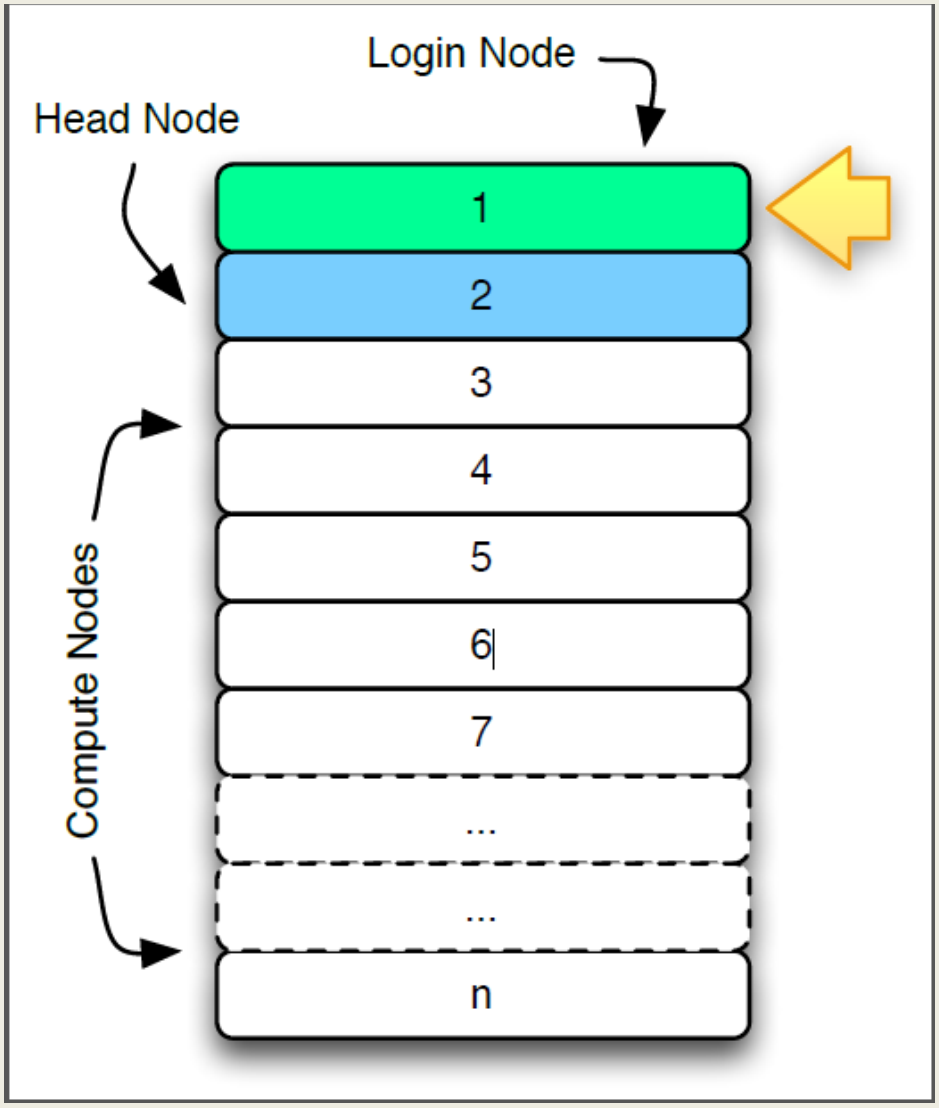
The typical HPC workflow

- In HPC we talk about **jobs**, these are simply commands we wish to run and requests for resources (e.g. compute time, disk space, memory requirements, setup of s/w env's etc.)
- Jobs are typically run **non-interactively**
- Can be run **interactively** for testing purposes
- We add our jobs to a **queue**.
- When machines have **free resources** jobs run

The HPC "Cluster"



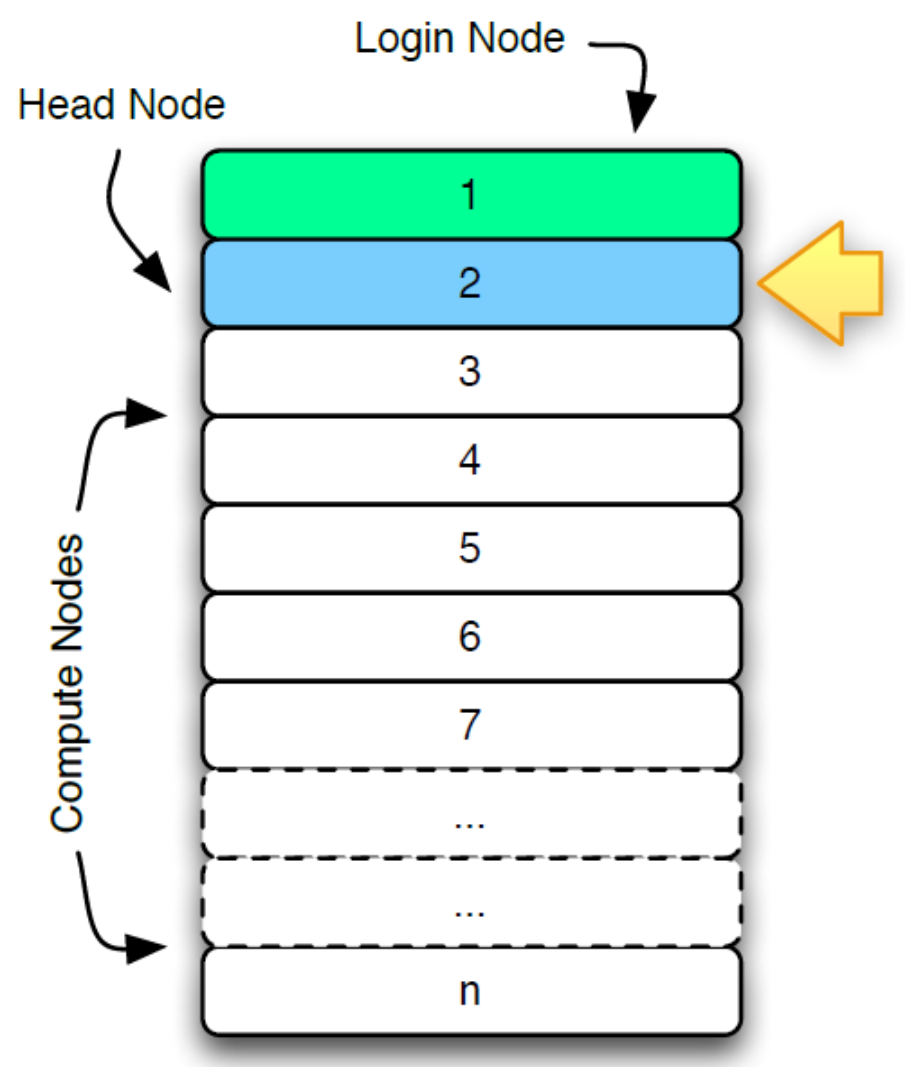
The Login Node



Login Node

- Interactive programs
- SSH sessions
- Testing
- Compiling

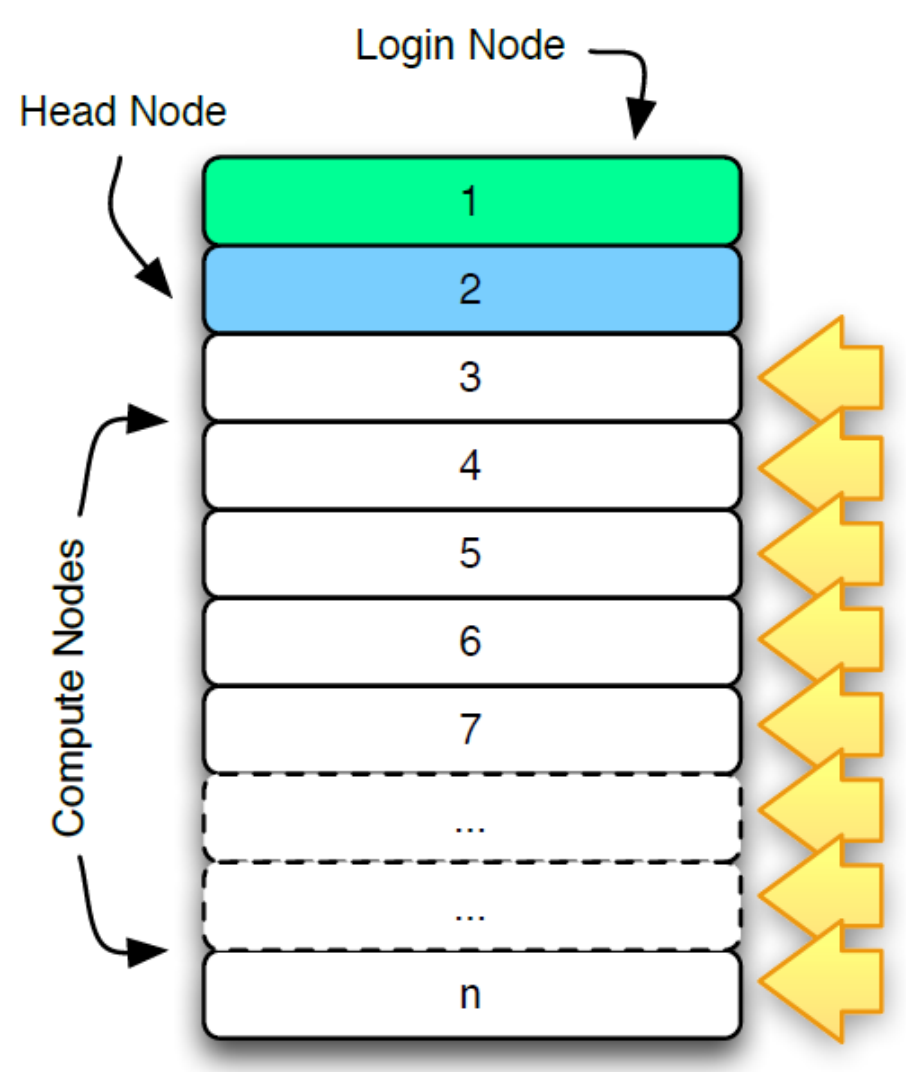
The Head Node



Head Node

- Queuing jobs

Compute Nodes

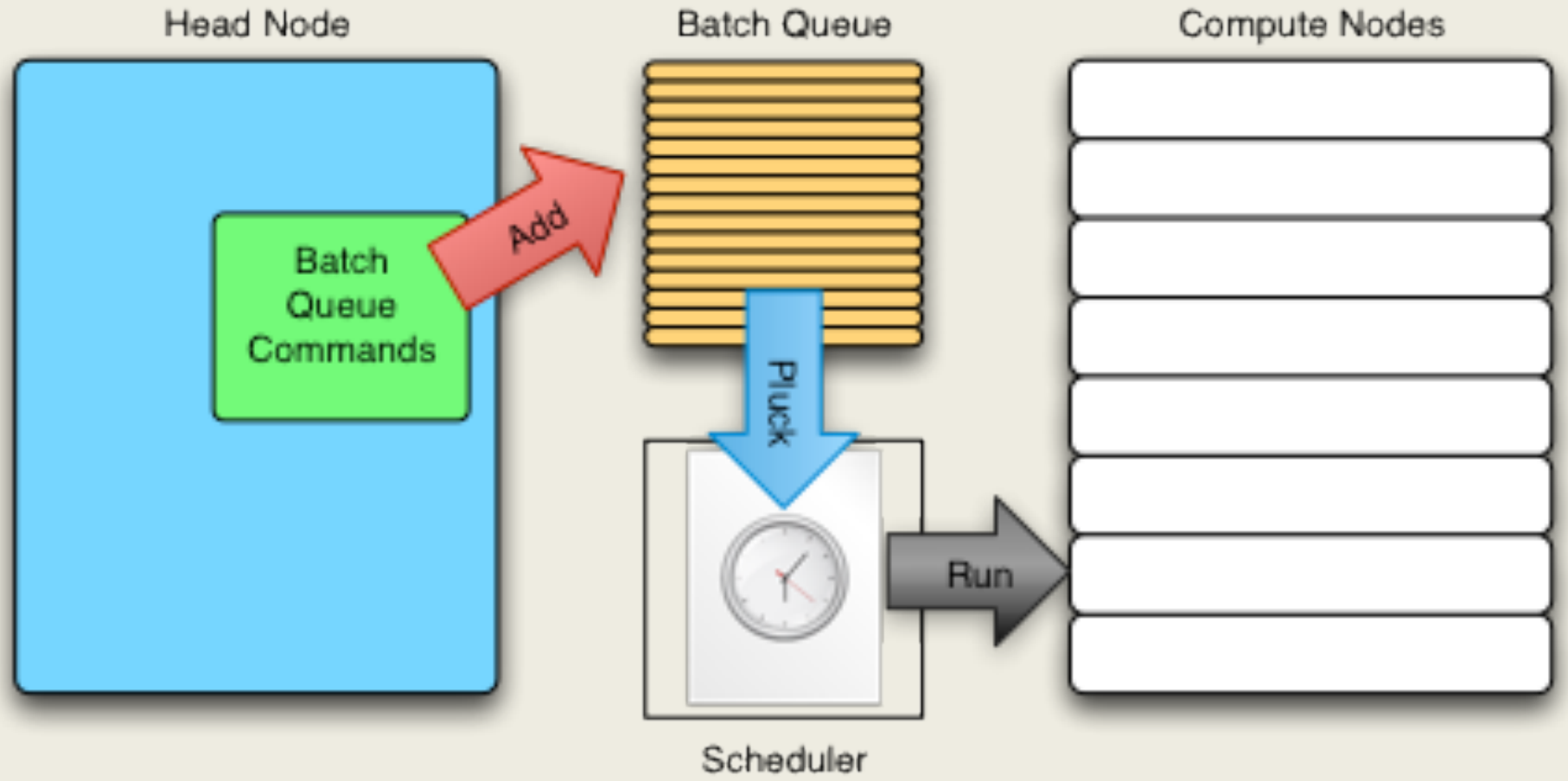


- These nodes run your jobs
- Managed by the **scheduler**
- Typically you won't interact with the nodes directly

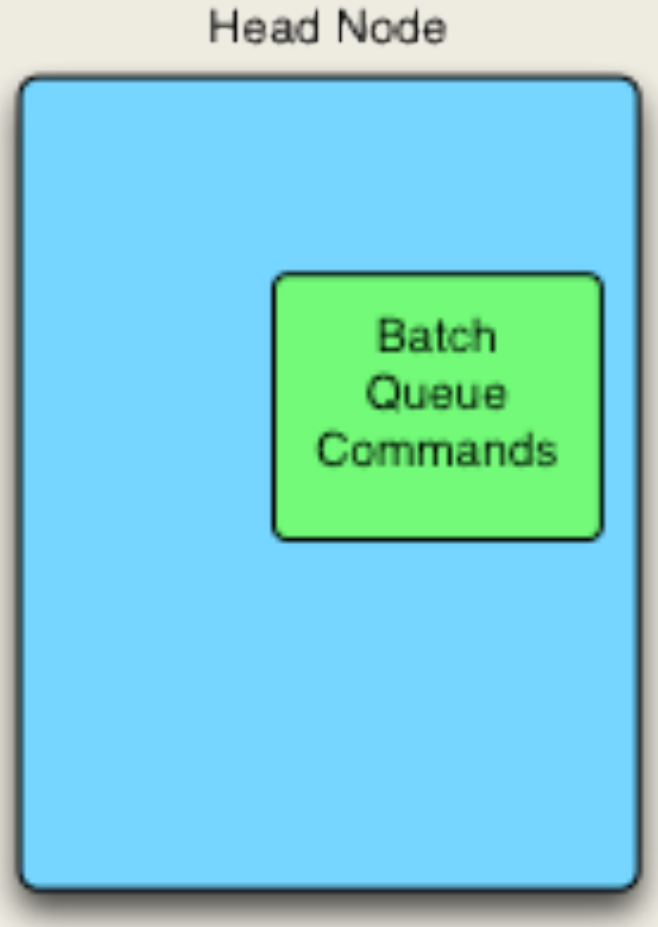
Queuing Systems

- **Portable Batch System** (PBS) is a scheduler that performs job management. Its primary task is to allocate computational tasks, i.e., batch jobs, among the available computing resources.
- Other schedulers are available such as SLURM.
- Schedulers can be highly customised. So expect differences between data centres.

The Batch Queuing System



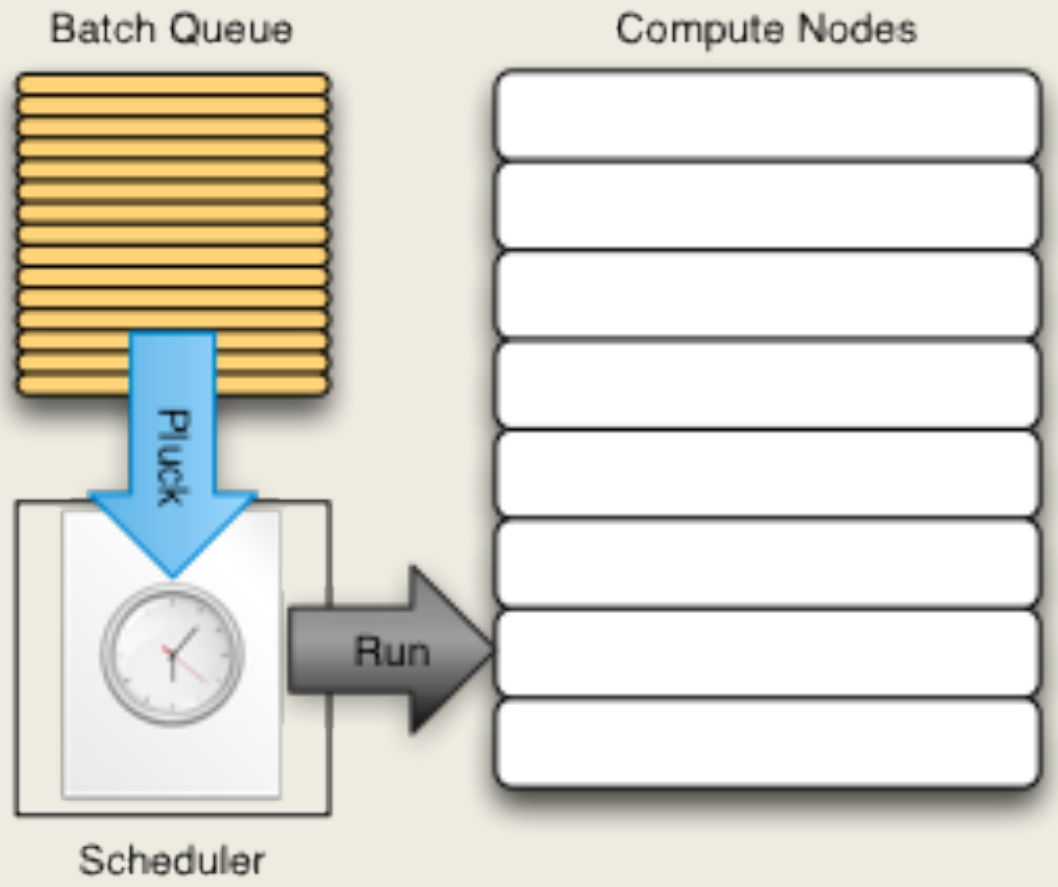
Batch Queuing component



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

The Scheduler component

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



PBS Pro Commands

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

Exercise 1

Monitoring the queue with qstat

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

- Log into Raijin
- List all jobs on Raijin
- Pick a user and list jobs of this user
- Pick a job and see all details about this job

INTERSECT

All about Modules

- **What do Modules do?**
  - Set up the environment for a software package
  - Adds paths to executables to $PATH
  - May change other shell variables, and/or load other modules.
  - Allow you to have different versions of the same software package, e.g. load the Intel compilers v12 or v15.

All about Modules (cont.)

Things to know about modules

- Only the PBS module is loaded when you login.
- Some modules exclude each other, e.g.
  - You can't load Intel compilers v8 & v9. You can only load one.
  - You can only load 1 MPI (Intel MPI or OpenMPI) and only in one version.

Module Commands

Command	Description
`module avail`	Will list all available module files
`module load/unload`	Will load/unload a modulefile into the shell environment e.g. `module load mpt/2.06`
`module list`	List all loaded modules which are loaded
`module show [modulefile]`	Will show what the modulefile will do

Add a job to the queue

- To add a job to the queue, we write a **job script**. A job script is a simply a script.
- The # symbol signifies a comment for the Shell
- It has some special comments that pass info to PBS Pro.
- **#PBS** is a keyword for PBS and specifies that this line is for PBS. The Shell will ignore it
- 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.

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

A sample PBS job script for Orange

Note the "-l" below is a lowercase letter "L" not a one "1"!

```
#!/bin/bash
# * Specify your project code
#PBS -P do18
# Request resources
# * 10 minutes wall time to run
#PBS -l walltime=00:10:00
# * 1 node, 1 processor
# * 100 megabytes physical memory allocated to job
#PBS -l ncpus=1
#PBS -l mem=100mb
# Specify the express queue
#PBS -q express
cd $PBS_O_WORKDIR
# Specify the job to be done
date
sleep 60
date
```

Exercise 2

Create the previous script and submit it as a very simple job

Command	Description
`#PBS`	**`#PBS`** is a keyword for PBS and specifies that this line is for PBS. The Shell will ignore it
`#`	Signifies a comment for the Shell, e.g. **`# Next line will create a job`**
`qsub`	Submit a job to PBS
`qstat`	List jobs in the queue
`cat` *<filename>*	Print a file to the terminal (catenate)
`less` *<filename>*	Like **`cat`**, but less at a time
`nano` *<filename>*	Will open file *<filename>* in a text file editor
`Sample PBS Script`	*see exercises document*

INTERSECT

Job limits on Raijin

- 48hours of **walltime**
- 32GB, 64GB, 128GB, 256GB nodes are standard (scheduler will decide)
- 3 x 1TB available via hugemem queue
- 3 x 3TB coming soon
- **NOTE**: If you grab a node with 64GB, you can effectively use about 60GB as the OS uses memory

Raijin queues

Queue	Charge weight	Comments
normal	1.00	Normal queue
express	3.00	Express queue. Jobs are submitted faster.
normalbw	1.25	Queue using Broadwell nodes (can be cost effective)
expressbw	3.75	Express queue using Broadwell nodes
knl	0.25	Intel KNL nodes, 64 cores per node
gpu	3.00	4 x K80 per node. 30 nodes.
gpupascal	4.00	4 x P100 per node. Just 2 nodes.
hugemem	1.25	For access to the 3 x 1TB and 3 x 3TB nodes

NCI Facilities

Disk Type	Disk Usage
NCI Raijin Sandy Bridge	• **2 sockets with 8-core CPUs = 16 cores** • 57,472 cores in the compute nodes • Approximately 160 TBytes of main memory; • Infiniband **FDR** (five data rate) interconnect
NCI Raijin Broadwell	• **2 sockets with 14-core CPUs = 28 cores** • 27,184 cores in the compute nodes • Infiniband **EDR** (eight data rate) interconnect
NCI Raijin Lustre Filesystem	• Approximately 42 PBytes of usable fast file system

Software on Raijin

Area	Software
Computational Chemistry	ABINIT, Amber, CPMD*, GULP*, NAMD*, Molpro etc.
Bioinformatics	AbySS, BEAST, BIOPERL, Cufflinks, MAW, etc.
Math Libraries	ARPACK, BLACS, Boost, FFTW, GSL, MKL, Tao
Statistics & Maths Env's	Maple*, Mathematica*, MatLab*, Octave*, R, Stata*

- Asterisked items indicates that discussion with NCI facility staff is required before use (Licensing issues)
  https://opus.nci.org.au/display/Help/Application+Software

Disk Partitions

/home	
Mounted under:	`/home/{username}`
Size:	2GB
Backed up:	Yes
Life time:	Permanent

/short	
Mounted under:	`/short/{project_code}`
Size:	Depending on the project allocation
Backed up:	**No**
Life time:	**60 days**

Disk Partitions

/g/data1, 2, 3	
Mounted under:	`/g/data[1,2,3]`
Size:	Depending on project allocation
Backed up:	**No**
Life time:	Permanent

Massdata	
Mounted under:	`/mdss`
Size:	Depending on project allocation. Intended to be a backup.
Life time:	Permanent

Disk Partitions

Find out your quotas and usage.

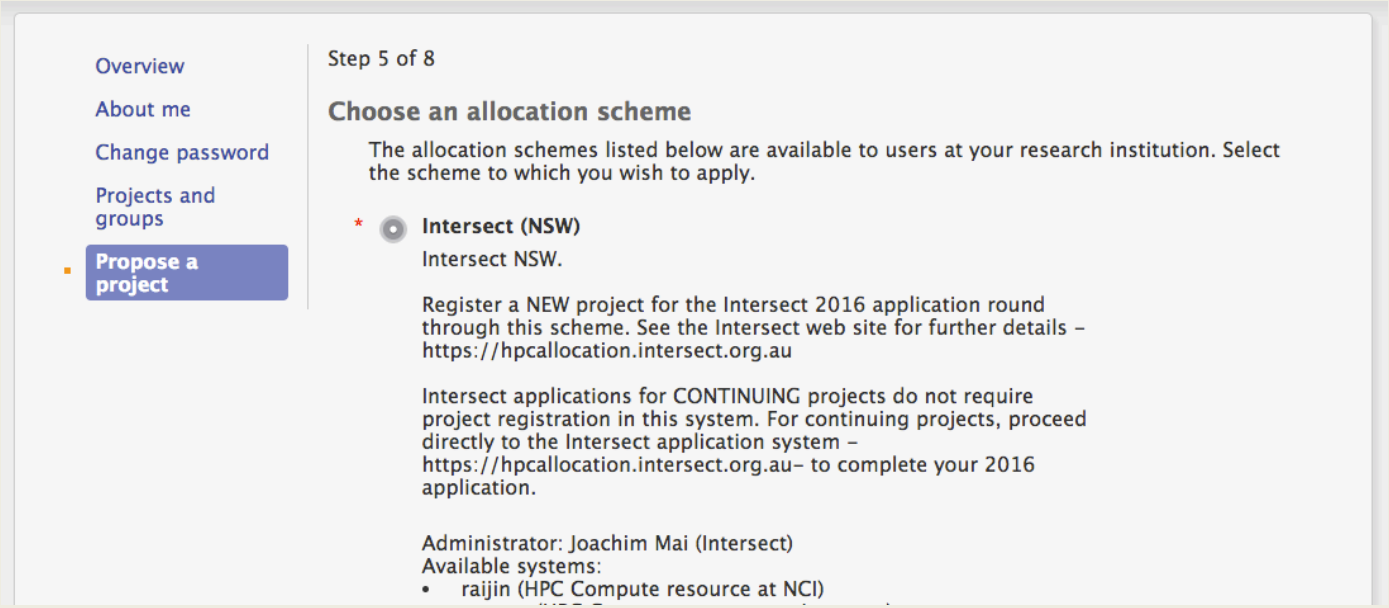
Command	Description
`lquota`	Shows disk quota and usage in all partitions

You can find out more about current disk usage of files or directories:

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

To Apply

- Create an ID in NCI Mancini
- Apply for a new project (*propose a project*)
- Select Intersect under allocation scheme



Applications continued

- If you have any problems requesting resources, email help@intersect.org.au who can advise you
- You can also add accounts to an existing project via Mancini
- You can also add various software groups (e.g., MATLAB) via Mancini

- Check usage and allocation on Raijin via
- nci_account -P project-code

Conclusion

- In this course we have covered
  - the basics of the Unix command line
  - transferring data onto Raijin
  - how to queue a job on Raijin and get the output

Thanks for attending!

Please complete our **course survey** at:

https://goo.gl/EbC9eO <- Capital 'Oh'

Any **further questions**, contact us at training@intersect.org.au

INTERSECT