WORKSHOP:
How to use HPC & Cloud Clusters to enhance your research outcomes

Paul Coddington, Deputy Director eRSA

9 March 2016

eRSA
Advancing Research Innovation

What is eRSA?

- An incorporated joint venture of the three South Australian universities.
- The point of focus in S.A. for eResearch infrastructure and support.
- The South Australian partner in national eResearch initiatives.
- A mechanism for attracting funding for shared eResearch infrastructure to SA, e.g. through ARC LIEF grants and federal eResearch infrastructure programs

University of South Australia

THE UNIVERSITY of ADELAIDE

Flinders UNIVERSITY

nectar

RDS Research Data Services

ands AUSTRALIAN NATIONAL DATA SERVICE

eRSA Services

- Large-scale data storage facilities
- Data sharing and data repositories
- High-performance computing facilities
- Hosting of dedicated compute facilities
- Cloud computing and virtual machines
- eResearch expertise and consulting
- Helpdesk, user support and training
- Software development
- Development and/or hosting of web applications
- Visualization services

How much does it cost you?

Nothing!*

*Some conditions apply ...

Funding and Fees

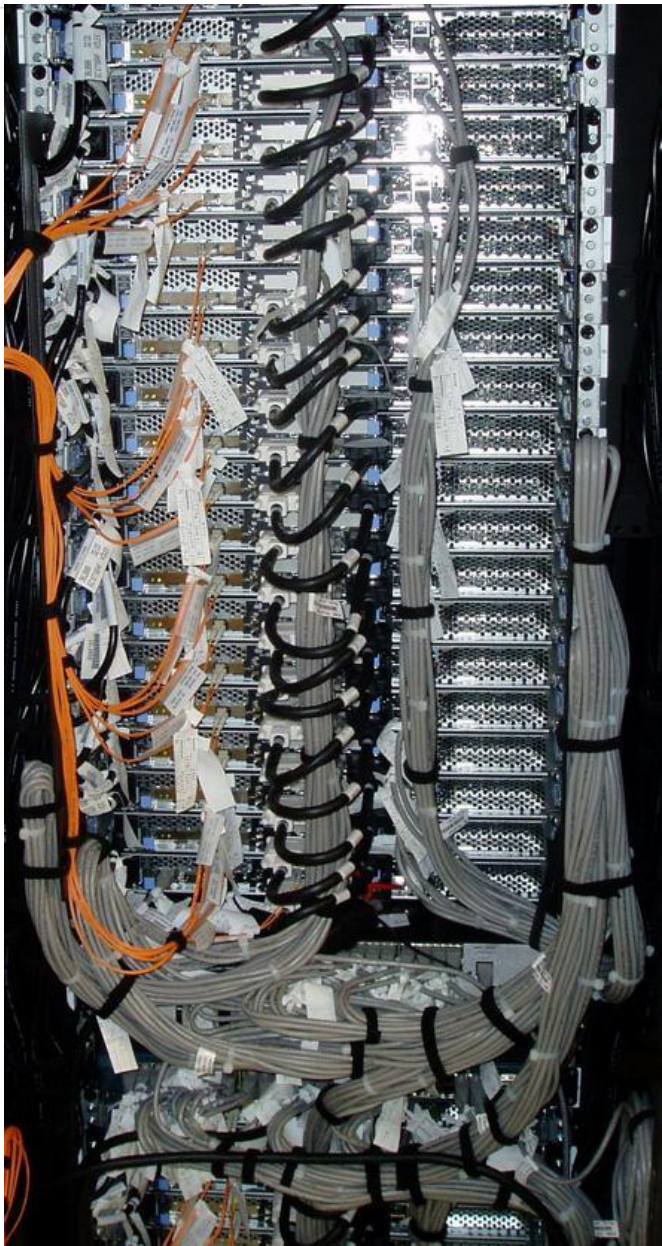
- Main funding for eRSA comes from:
  - The three S.A. universities
  - SA Government (some project funding)
  - National eResearch funding schemes (RDS, NeCTAR, etc)
  - ARC Linkage Infrastructure, Equipment and Facilities (LIEF) grants
- eRSA also does paid contract work, e.g. development of software, provision of custom services, consultancies, etc
- Usage costs of most shared services including HPC, storage, cloud are covered by the universities, so no cost to researchers
- Services that are not free are low cost for uni researchers:
  - Dedicated compute nodes or facilities
  - Software or database development
  - Significant ongoing dedicated support effort

Access to eRSA HPC Facilities

- Shared HPC facilities are available no cost to researchers at (or affiliated with) the 3 SA universities and other SA research organisations (e.g. SAHMRI)
- Usage costs are (currently) covered by the institutions
- Amount of resource available to a user is approx proportional to funds provided by their research group, School, Faculty, university to purchase the equipment
- Some dedicated facilities are directly paid for by their users
- eRSA also provides user support, helpdesk, software installs, online user guides, and training

High-Performance Computing

- Supercomputers provide hundreds or thousands of times more compute power and memory than a PC
- HPC enables much faster data processing, analysis, simulation and modelling
- Also enables researchers to tackle larger, more complex problems than could otherwise be done
- Has been used for many years in physics, chemistry, engineering
- Recently much broader usage, e.g. molecular biology, analysis of large data sets in many fields



Supercomputers

- Modern supercomputers are parallel (multi-processor) computers with hundreds or thousands of processors.
- Usually commodity processors – similar to those in a desktop PC.
- Usually commodity compute servers connected by fast networks (10Gbit Ethernet, Infiniband)
- This is a change from early custom-built supercomputers

Supercomputers

- Increase in speed of supercomputers over desktop computers is from using multiple CPUs at once, not from faster CPUs.
- This approach is now moving to desktop PCs with multicore processors.
- Servers and HPC moving to “manycore” processors
- So to gain benefit from supercomputers requires getting your application to run on multiple processors – parallel computing
- No free lunch!

HPC at eRSA

- eRSA manages facilities for what has traditionally been called High Performance Computing (HPC) or Supercomputing.
- Shared resources, managed by eRSA, for the academic research community of SA.
- Available for any researcher, in any discipline, from any of the 3 SA universities or SAHMRI.
- Aim to satisfy the different types of resource demands of quite different groups of researchers.
- Set up in a standard way for HPC facilities.

eRSA HPC Facilities

- Tizard supercomputer
  - CPU cluster, 28 nodes with 48 cores per node
  - GPU nodes, 5 nodes with 4 GPUs per node
  - Two big memory nodes
    - 1 TB and 512 GB RAM, 32 cores
    - More coming soon
- Dedicated servers and clusters
- Emu cluster in the cloud
- New HPC system coming in 2017

Other HPC facilities

- Some other HPC facilities are also available for SA researchers:
  - NCI – national HPC facility
  - Pawsey – national HPC facility
  - Colossus – Flinders Uni
  - Phoenix – Uni Adelaide
- Contact them directly or ask eRSA for help on which might suit your requirements
- Most of the information in today's training is also applicable to these facilities
  - The concepts are the same, but details of job submission are slightly different

Parallel Computing

- Two basic options for efficient parallel computing.
- **Reduce completion time of a single run**
  - Speed up the execution time of a single program run by dividing up the computation among the processors.
  - *High-performance computing*.
  - Need to modify (parallelize) the program.
- **Reduce total completion time of many runs**
  - Run many instances of the same program concurrently, each on a different processor.
  - *High-throughput computing*.
  - Don’t need to change the program.

Parallel Computers

- Many compute nodes (or servers) connected by a fast network
  - Usually Infiniband or 10 Gbit Ethernet (Tizard uses Infiniband)
- Each node has multiple processors
  - Usually 2 or 4 (Tizard has 4)
- Each processor has multiple processing cores
  - Usually 4 to 16 (Tizard has 12 cores per processor)
  - “Manycore” processors coming soon, new Intel Xeon Phi processor has up to 72 cores
  - GPUs and some custom chips have hundreds of cores
- A single compute node can have lots of cores
  - 32 or 64 now inexpensive and common (Tizard has 48)
  - Only 15 years ago a 64-processor SMP was a supercomputer!

Operating System

- Note that all eRSA HPC systems use a flavour of the Linux (Unix) operating system
  - Some dedicated servers or VMs use Windows and some applications have GUI or web interface
- So you will need to have a basic knowledge of Unix commands in order to use the HPC
- Unix text editor such as nano, vi/vim, emacs
  - Or edit files on your PC and copy them over
- Lots of online resources for learning Unix and text editors, including on the eRSA web site

Using eRSA HPC facilities

Windows users: We recommend the following software:

- **PuTTy**, an ssh client for logging in to eRSA facilities.
- **FileZilla or WinSCP**, secure file copying programs with a drag and drop interface.
- **Xming**, a free X Windows server, needed for any gui based editors, such as emacs (get all fonts).

BEWARE the ^M problem when copying files from your Windows desktop to the Linux HPC systems. Can fix with

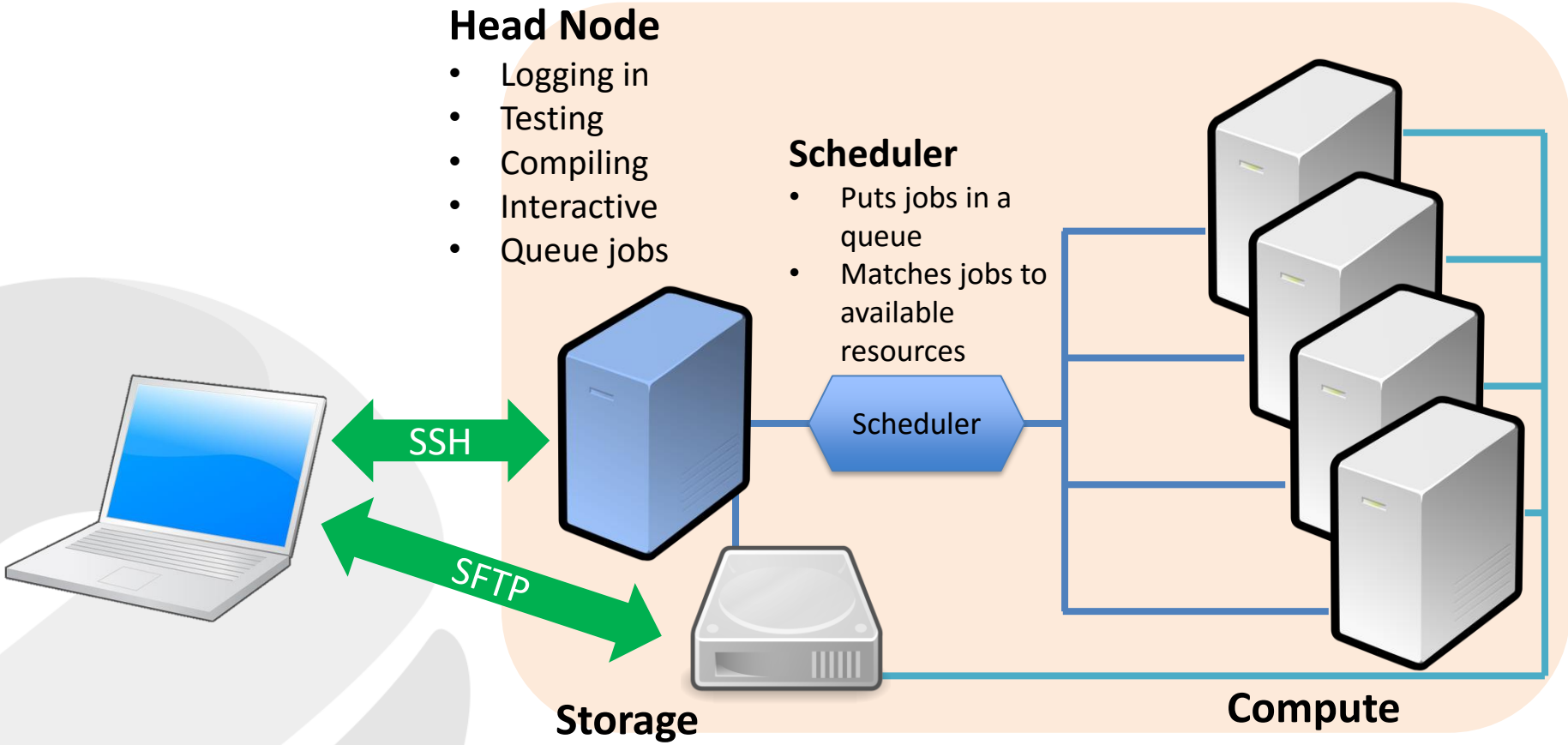
`dos2unix` *`filename`*

Or use programs like Notepad++ that can handle different formats

Mac or Linux users: can also use terminal window and Unix commands.

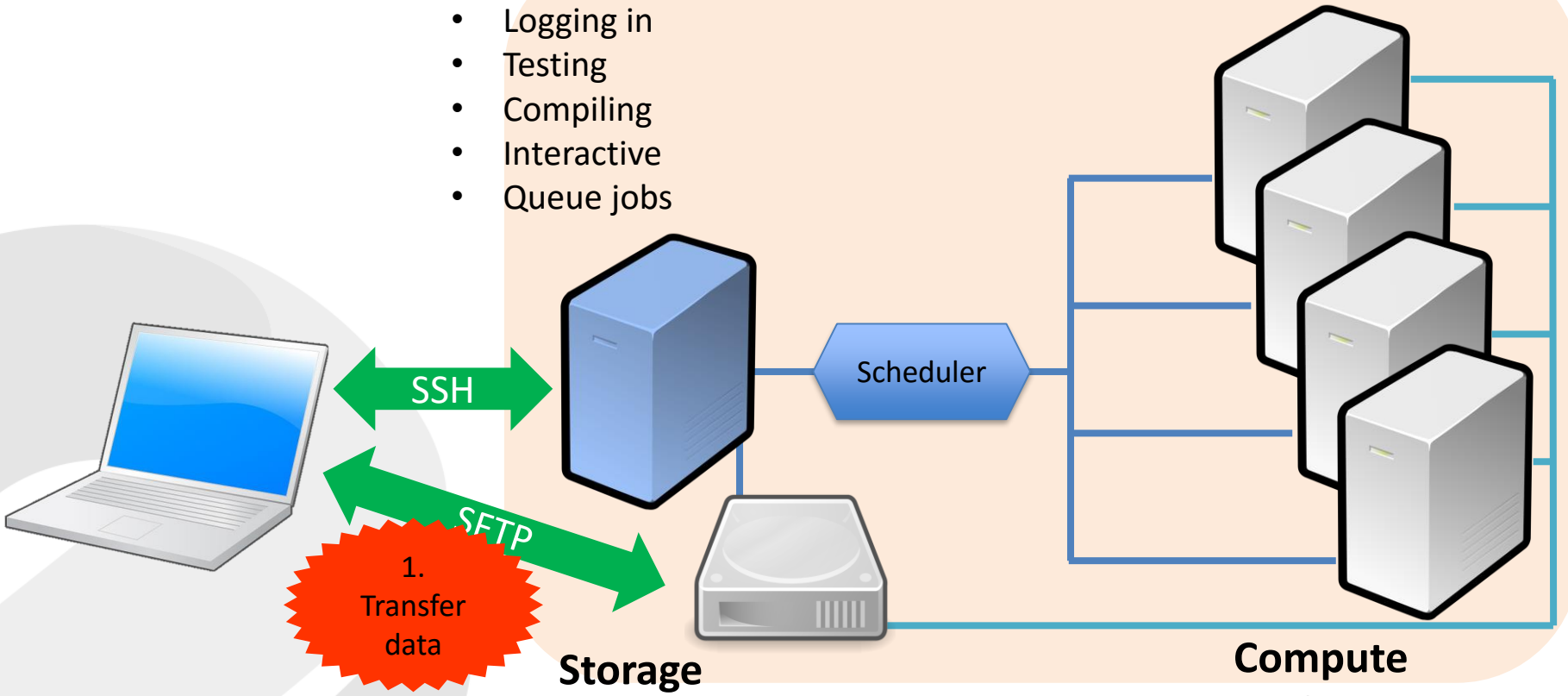
Running Compute Jobs

- Your application software must run on Unix
  - Option to use cloud or VM server if Windows
- You can compile and run your own programs
- or use programs that are already installed
- You need to know (or figure out from application user guide) how to run the application software
- Think about how to run jobs using multiple processors (we can give guidance)
- Then make a script to run the software in batch mode on the supercomputer


Head Node
• Logging in
• Testing
• Compiling
• Interactive
• Queue jobs
Scheduler
• Puts jobs in a queue
• Matches jobs to available resources
Scheduler
SSH
SFTP
Storage
• Home Folder
• /data access
• /scratch as a temporary location for files that need to be written / read
Compute Nodes
• Run jobs
• Managed by scheduler
• Non-Interactive

Head Node

- Logging in
- Testing
- Compiling
- Interactive
- Queue jobs

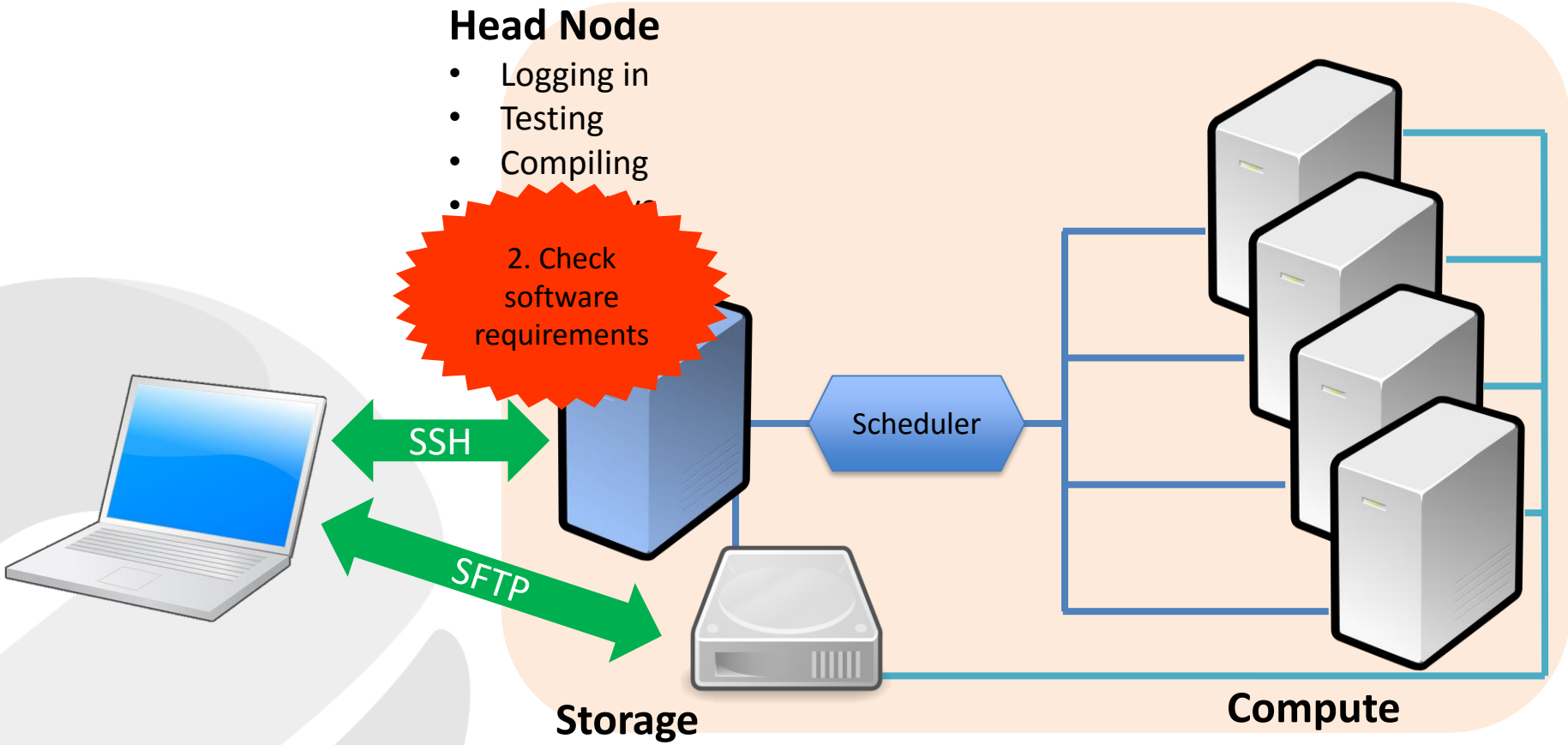


Storage

- Home Folder
- /data access
- **/scratch** as a temporary location for files that need to be written / read

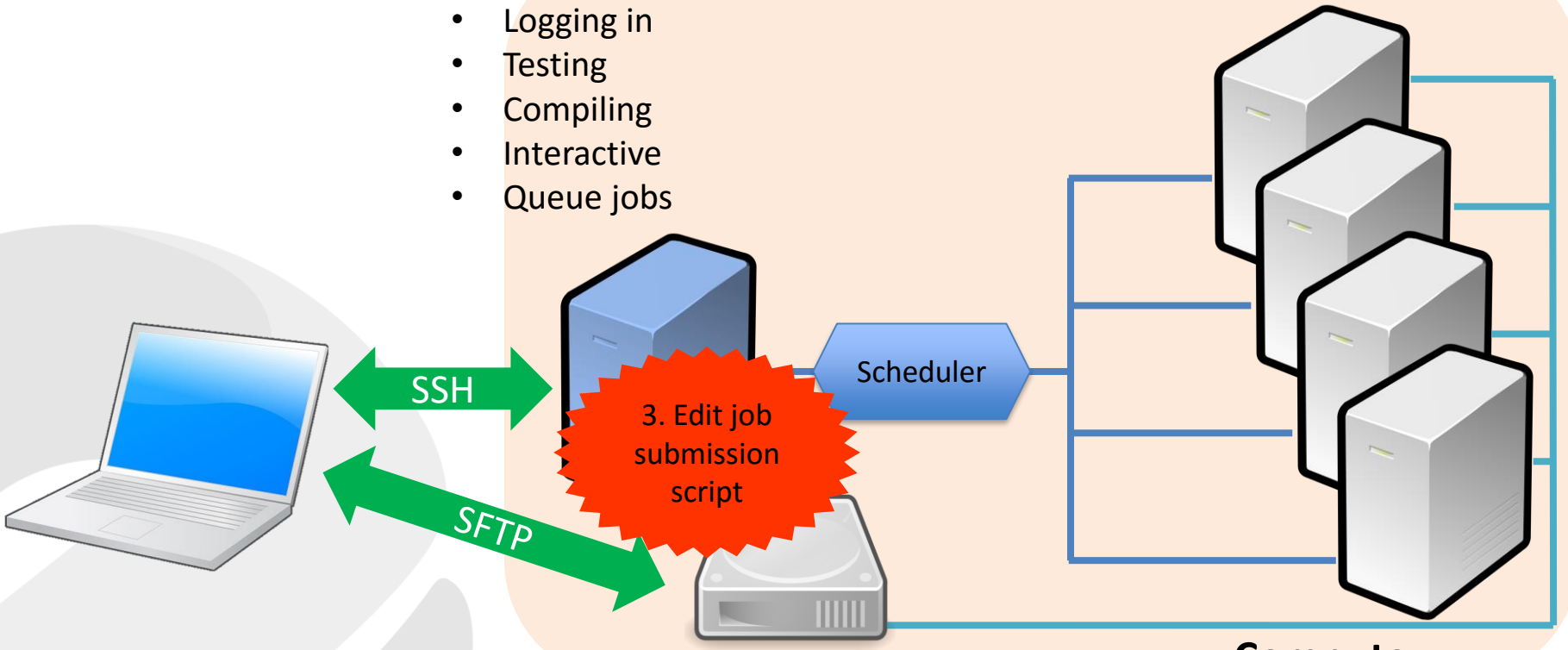
Compute Nodes

- Run jobs
- Managed by scheduler
- Non-Interactive


Head Node
• Logging in
• Testing
• Compiling
2. Check software requirements
SSH
SFTP
Scheduler
Storage
• Home Folder
• /data access
• /scratch as a temporary location for files that need to be written / read
Compute Nodes
• Run jobs
• Managed by scheduler
• Non-Interactive

Head Node

- Logging in
- Testing
- Compiling
- Interactive
- Queue jobs



Storage

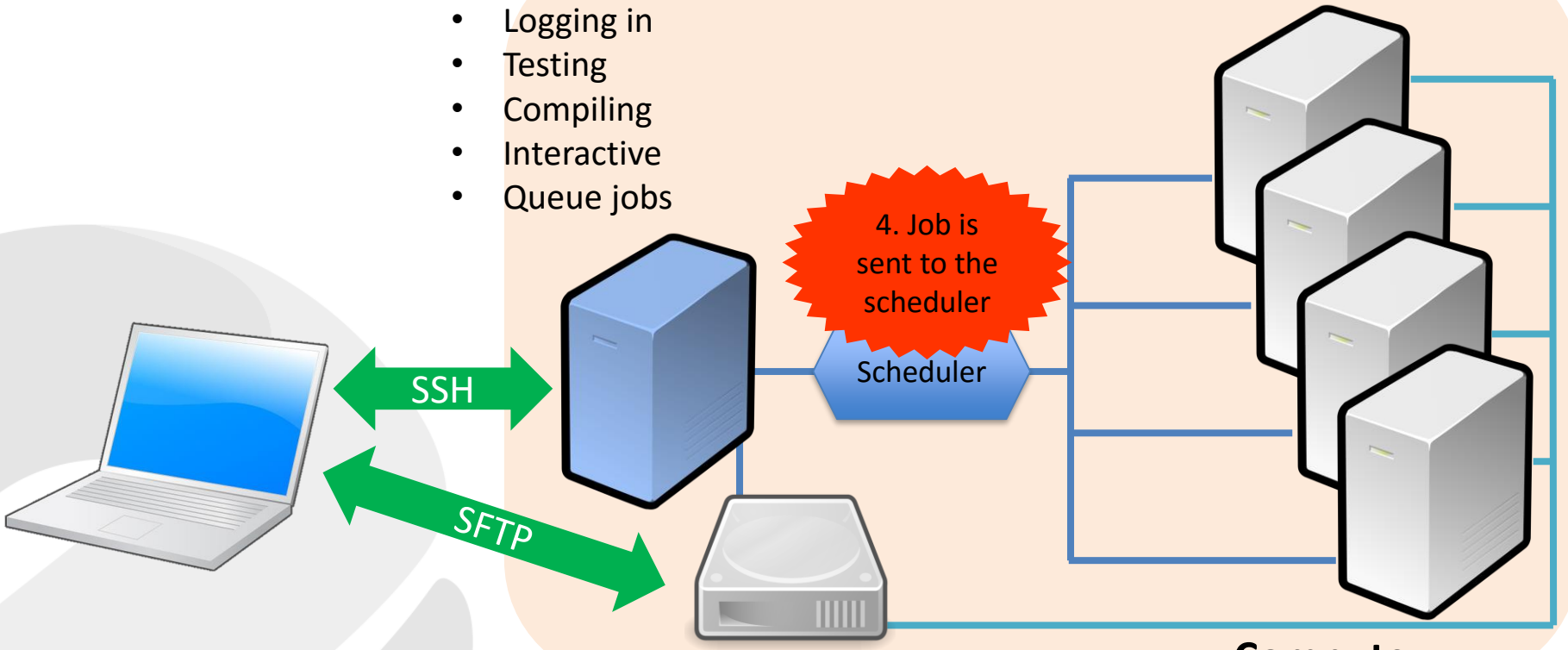
- Home Folder
- /data access
- **/scratch** as a temporary location for files that need to be written / read

Compute Nodes

- Run jobs
- Managed by scheduler
- Non-Interactive

Head Node

- Logging in
- Testing
- Compiling
- Interactive
- Queue jobs



Storage

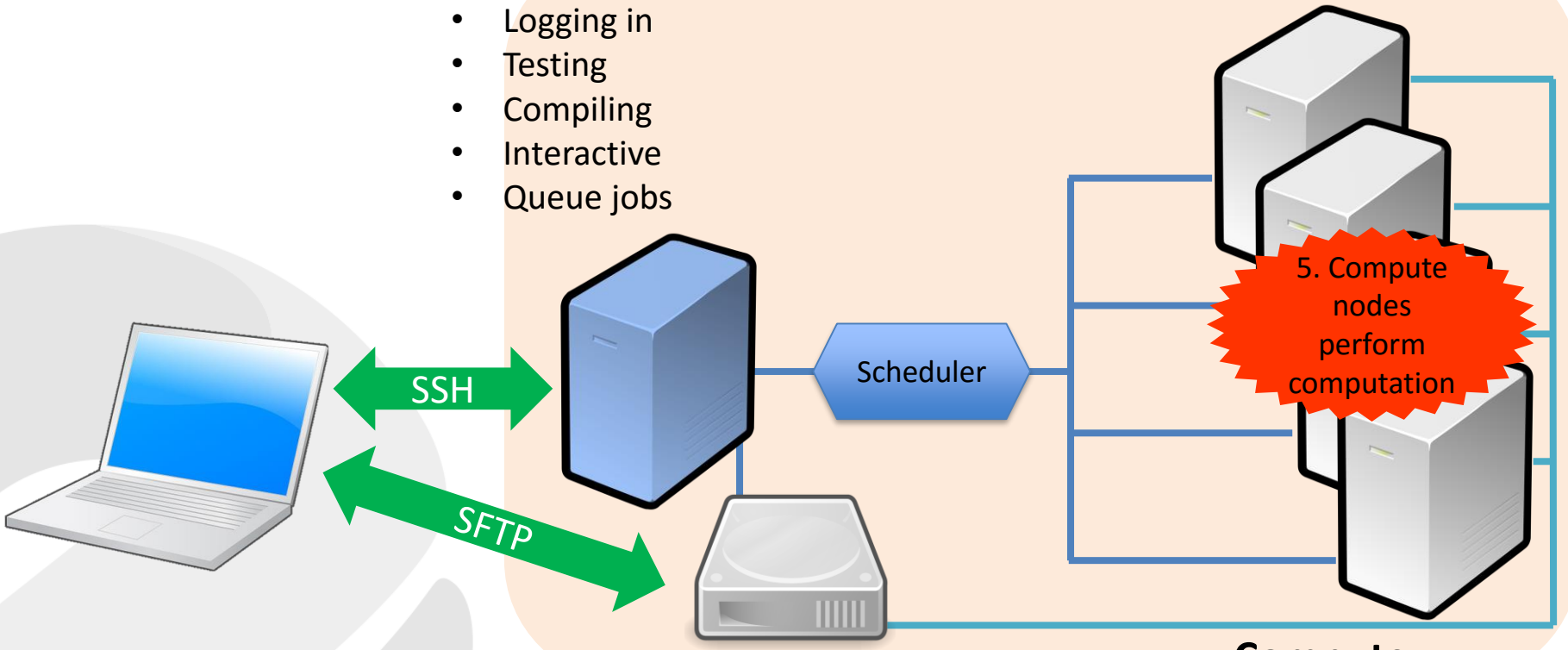
- Home Folder
- /data access
- **/scratch** as a temporary location for files that need to be written / read

Compute Nodes

- Run jobs
- Managed by scheduler
- Non-Interactive

Head Node

- Logging in
- Testing
- Compiling
- Interactive
- Queue jobs



Storage

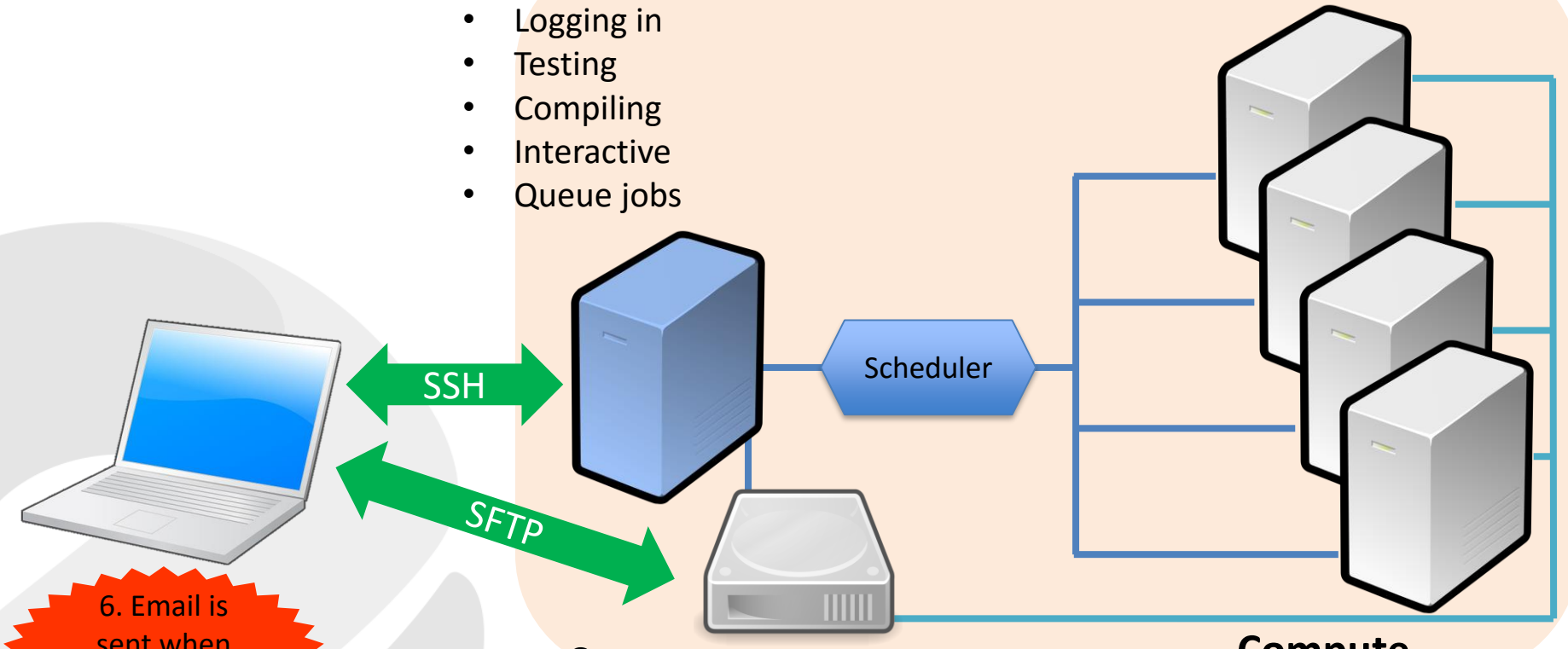
- Home Folder
- /data access
- **/scratch** as a temporary location for files that need to be written / read

Compute Nodes

- Run jobs
- Managed by scheduler
- Non-Interactive

Head Node

- Logging in
- Testing
- Compiling
- Interactive
- Queue jobs



6. Email is sent when job is complete

Storage

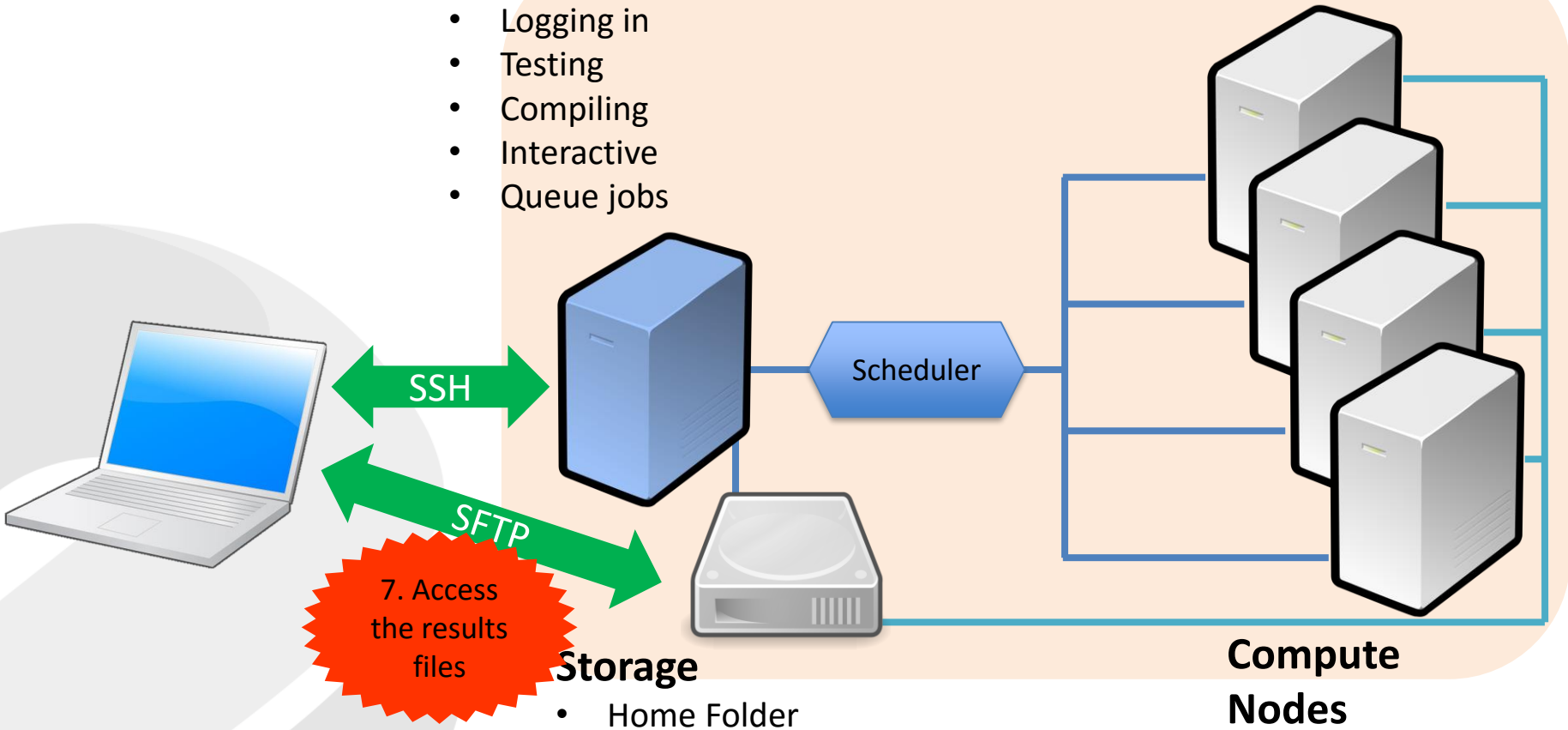
- Home Folder
- /data access
- **/scratch** as a temporary location for files that need to be written / read

Compute Nodes

- Run jobs
- Managed by scheduler
- Non-Interactive

Head Node

- Logging in
- Testing
- Compiling
- Interactive
- Queue jobs



Storage

- Home Folder
- /data access
- **/scratch** as a temporary location for files that need to be written / read

Compute Nodes

- Run jobs
- Managed by scheduler
- Non-Interactive

eRSA
Advancing Research Innovation

Let’s connect to the Tizard head node and look around…

Shared and Distributed Memory

Shared memory (SMP)

- Processing cores on a compute node all have shared access to all the memory on the node
- Parallel programs often written using multiple *threads*, usually with one thread per processing core
- Shared memory multi-threaded programs can only run on a single compute node

Distributed memory (cluster)

- Processing cores on one compute node can't directly access memory from other nodes
- Program needs to send information using *message passing,* usually using the Message Passing Interface (MPI) standard
- Message passing programs can run on multiple compute nodes

Application Software

- Over 300 software packages already installed on Tizard!
  - Some with a few different versions available
- Some of the software available on Tizard:
  - **Chemistry**: Gaussian09, NWChem, Terachem, NAMD, Amber
    Bioinformatics: Blast, BEAST, MrBayes, Paup, ClustalW2, RepeatMasker, Bowtie, STACKS, R, etc etc.
  - **Engineering & Maths**: OpenFOAM, R, Matlab
- To find all software installed, type

```
module avail
```

- Other software can be installed on request to the eRSA helpdesk.
- User/University must purchase (network) license for commercial licensed software

Compilers

- If you write your own code it needs to be compiled and linked against system libraries.
- GNU Compilers, gcc, g77, g++
- Intel compiler suite, icc, ipCC, ifort. Includes Intel's MKL with optimised BLAS, LAPACK, PARDISO libraries. Usually best performance.
- OpenMPI for parallel MPI programs.
- CUDA development environment for GPUs.

Software Libraries

- OpenMPI (MPI library)
- Common maths libraries
  - fftw
  - lapack, scalapack, atlas
- Other libraries
  - hdf5
  - guile
  - oomph
  - bioperl, biopython

Others can be installed on Tizard by request to the helpdesk.

Queueing System

- To ensure efficient and fair use of resources all jobs run on eRSA's facilities need to be submitted as batch jobs to a job management system or *queuing system*.
- This is standard practice for running HPC systems.
- The queuing system we use is Torque, a variant of the PBS queuing system.
  - very similar to the system run by NCI and Pawsey
  - similar to other HPC clusters e.g. Phoenix, Colossus
- Note that you can't (easily) run jobs interactively, they must be specified in a script so they run automatically when the resources they need (memory and CPU cores) become available.
- The cloud is more suited to interactive jobs.

Overview of Torque queuing system

- Jobs are allocated dedicated compute resources (processing cores, memory) on the cluster.
- Users request the resources in a Torque *job script*, a Unix shell script with formatted comments to specify Torque attributes.
- The job script is submitted to the Torque batch queue, where it waits until the required resources are available.
- The *scheduler* (we use one called Maui) allocates resources (nodes, cores, memory) to a job
- Torque runs the job on the appropriate nodes.

```
less .templates/tizard.sub
```

To look at the template job submission script.

To run a job, you will copy and modify this script.

Example job script - sequential

```
#!/bin/tcsh
### Job name
#PBS -N MyJobName
### Output files
#PBS -j oe
### Mail to user when job ends or aborts
#PBS -m ae
#PBS -M fred.bloggs@ersa.edu.au
### Queue name
#PBS -q tizard
### Number of nodes, memory, walltime.  REQUIRED
#PBS -l nodes=1:ppn=1
#PBS -l mem=Xmb,vmem=Ymb
#PBS -l walltime=01:00:00
cd $PBS_O_WORKDIR
# Load modules if required
module load application
# Run the executable
applicationExe < InputFile.dat > OutputFile.log
```

Parameters to job script

```
### Number of nodes, memory, walltime.
#PBS -l nodes=N:ppn=P
#PBS -l mem=Xmb,vmem=Ymb
#PBS -l walltime=01:00:00
```

- The parameters in red need to be specified for your job
- What should you set them to be?

Walltime

```
### Number of nodes, memory, walltime.
#PBS -l walltime=01:00:00
```

- Walltime is an estimate of the time your job should take to run (HH:MM:SS)
- **Don't underestimate** – if the job takes longer than the specified walltime Torque will kill it!
  - Checkpoint if you can
- Don't overestimate too much – shorter jobs are likely to run (be scheduled) earlier
- There is a max walltime – 100 hours on Tizard

Number of processors

```
### Number of nodes, memory, walltime.
#PBS -l nodes=N:ppn=P
```

- N is the number of nodes
- P the number of processors per node
  - N <= 28 and P <= 48 on Tizard
- Total number of processors is N x P
  - Make sure you get this right when you specify the number of processors to your program!
- Check that your application can actually make use of multiple processors before putting P>1

Number of processors

```
### Number of nodes, memory, walltime.
#PBS -l nodes=N:ppn=P
```

- Check that your application can actually make use of multiple nodes before putting N>1!
  - Usually this means it's an MPI program
  - More nodes often easier to schedule
- Don't use more processors than your application can effectively use.
- Start with small numbers of processors and increase, compare the execution time, check that you are getting speedup with more processors

Memory

```
### Number of nodes, memory, walltime.
#PBS -l mem=Xmb,vmem=Ymb
```

- Read the user guide for your application to see if you can estimate memory requirements
- If not, run a test job with mem 2GB per processor
  - vmem is tricky – just set Y to be 2X
- If the program fails, double mem and try again
- If the program runs, check the PBS output file which will tell you how much mem and vmem were actually used, and use that (but may change with problem size)
- If you need >4GB per processor use Big Memory node

Submitting a job

To submit your job simply type:

qsub <shellscript>

where shellscript is the name of your Torque job submission script file, eg

```
qsub submit.pbs
```

You will get a response something like:

```
Job submitted.
Torque JobId: 28195.tizard
```

Checking status of a job

To view the status of your job simply type:

qstat –a

```
tizard:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
25016.tizard     sandery  seq      sef8        21497   1  --     --  1000: E 627:1
28167.tizard     hongyi   seq      newtest64s  29764   1  --     --  2440: R 02:03
28168.tizard     hongyi   seq      newtest64s    --    1  --     --  2440: Q   --
28193.tizard     honcao   seq      xcomp13     24585   1  --     --  1000: R 29:47
28195.tizard     honcao   seq      xcomp15     32087   1  --     --  1000: S 29:47
28196.tizard     honcao   seq      xcomp16     29707   1  --     --  1000: R 29:47
28201.tizard     honcao   seq      rxttfunc3    4870   1  --     --  1000: R 27:52
```

To see other options to qstat consult the Unix manpage.

```
man qstat
```

Delete a job

To delete a job from the queues simply type:

qdel <JobId>

where the JobId is the numeric part of the job identifier given to your job by the queuing system, eg

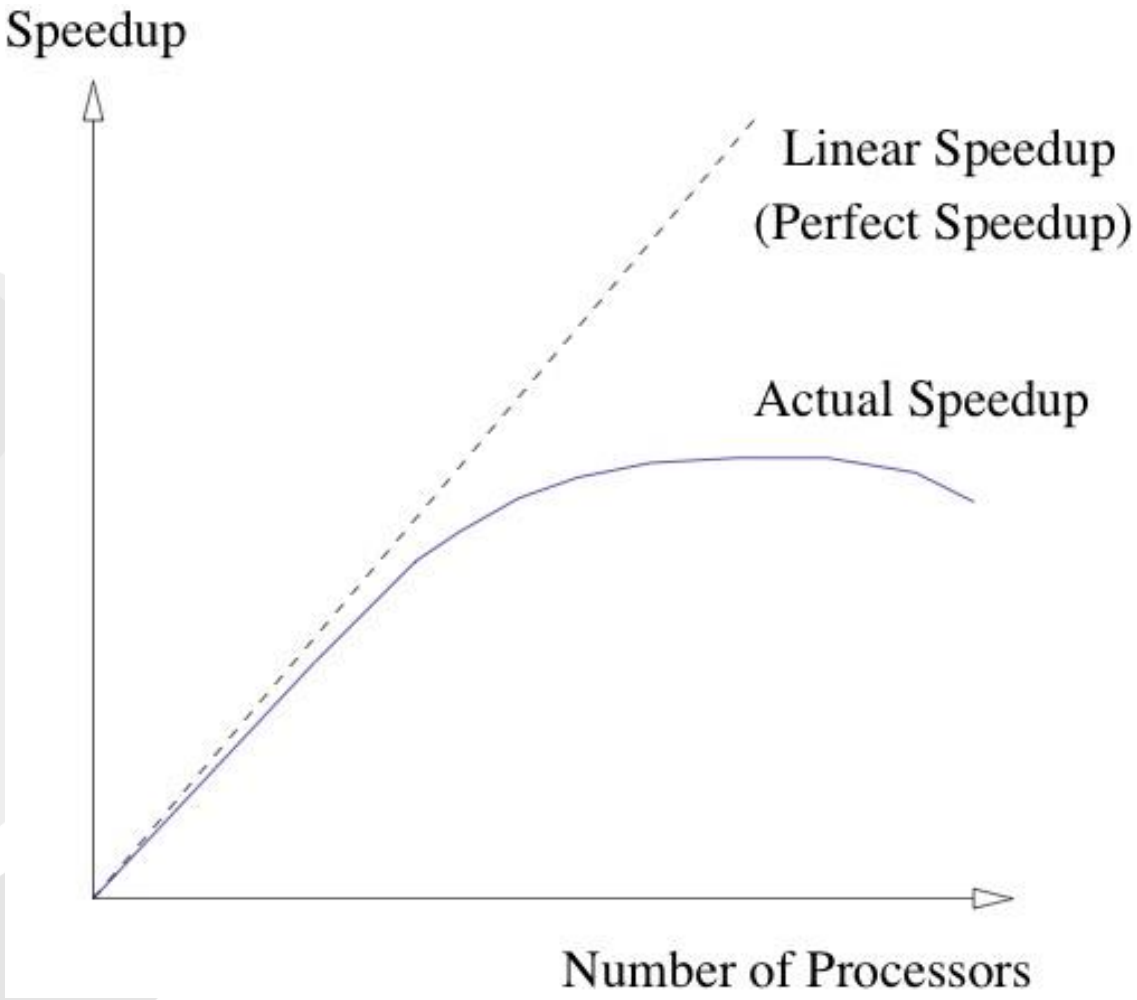
```
qdel 28195
```

you can find the JobId from qstat

Speedup

- Aim to get *speedup* on multiple processors, i.e. program runs faster
- Speedup = Time on 1 core / Time on N cores
- Would like to get speedup of N on N cores, i.e. program runs N times faster
- But not guaranteed – can go slower with more processors!
- Parallel overheads
  - Imbalance of workload between processors
  - Communication time to send information between processors
  - Sequential parts of the program

Speedup


Speedup
Linear Speedup
(Perfect Speedup)
Actual Speedup
Number of Processors

Find the right number of cores

- Run your program and measure speedup on different numbers of cores
  - e.g. 1,4,8,16,32,48 on Tizard CPU nodes
  - And more cores on multiple nodes if the application software can run on multiple nodes (e.g. an MPI program)
- Find the best number of cores to use
  - Where adding more cores doesn't give much additional speedup
  - e.g. using 8x more cores for 2x more speedup is wasteful!
- Note that best number of cores will vary depending on the problem size and complexity, i.e. the amount of computation required
  - So optimum cores may vary for different problem sizes

Queues

- The queueing system on Tizard has several different queues for different types of jobs.
  - tizard : CPU nodes (the default queue)
  - gpu : GPU nodes
  - bigmem : big memory nodes
  - short : jobs with walltime < 5 hours and <= 16 cores

- Specify the queue you want in your job script.
- For any Torque command specify a queue using –q *queuename*

Modules

- The Environment Modules package provides for the dynamic modification of a user's environment via module files.
- Better than static settings, e.g. in .cshrc.
- Each module file contains the information to configure the environment for an application.
- Modules can be loaded and unloaded dynamically and automically.
- Modules are useful for managing different versions of applications, or multiple applications with conflicting paths or environment variables.
- Look at the HPC user guides for more information.

Module commands

- **module avail** – what modules are available
- **module list** – what modules are loaded
- **module show** – what environment will be set
- **module load** – load the module
- **module unload** – unload the module
- **module whatis** – other information including what other modules need to be loaded (e.g. compiler for MPI)

File systems

- User files are stored in a large, scalable, high-speed file storage system that is directly accessible to the supercomputers.
- **/home/users/*username*** - your home directory
- **/scratch** – large shared scratch area of very high speed disk, available on all compute nodes.
- **/tmp** – fast local temp space on each node
- Please move or delete files from /scratch and /tmp after your job has finished.
- Home directory is backed up, but not scratch or tmp

Storage

- Each HPC user gets 200GB by default
- Projects or research groups can obtain additional storage
- eRSA has large amounts (Pbytes) of storage that can be mounted (via NFS) on eRSA supercomputers
- Currently the costs are covered by universities, but allocations require university approval
- To request a larger storage allocation for a research project or group, contact eRSA helpdesk or your local ITS

File I/O

- Performance of some applications can be limited by speed of file I/O (input/output).
- It is recommended that you run jobs from /scratch since the file I/O performance is much greater than your home directory.
- Create a sub-directory in scratch for your job.
- Copy job files (job script, input files) to /scratch and submit your job from there, then copy output to your home directory.
- If your program writes temporary files (e.g. Gaussian does this) use local /tmp on the node.
- Doing lots of small reads/writes and/or reading/writing lots of small files is inefficient and puts high load on file server, which slows down your job (and others). Try to aggregate them.

GPUs

- Consumer computer gaming market has driven huge performance increases in GPUs, which now have higher performance than CPUs.
- Modern GPUs are like parallel computers on a chip.
- nVIDIA Tesla M2090 GPU has 512 cores and peak performance of 1.3 TFlops single precision, 0.66 TFlops double precision.

GPUs

- But GPUs have a specialised architecture and programming model, so programs need to be rewritten for GPUs.
- Many applications have now been ported to GPUs.
- Some applications run very well on GPUs and can scale across multiple GPUs.
- However some give little or no performance benefit over a multi-core compute node.
- Your mileage may vary – check speedups for the application you are interested in.

Applications on GPUs

- Terachem
- OpenFOAM
- BeagleBEAST
- NAMD
- Amber
- Matlab
- LAMMPS
- Many others

- Some are installed on Tizard already. Ask the helpdesk if there are others you want installed.

Emu Cluster in the Cloud

- Emu is an eRSA cluster that runs in the NeCTAR national research cloud
- Aimed to be like HPC
- But a bit different
  - Dynamically created VMs in the cloud
  - More memory per core (4 GB)
  - 8-core compute nodes (32GB memory)
  - Can have 16-core (64GB) VMs if you really need it
  - Can have private compute nodes
  - Aimed at fairly small, single-node jobs

Using Emu

- Looks to user like a standard HPC cluster
- Accessible to anyone with an eRSA account
- But worker nodes are cloud VMs
- Emu is a *dynamic* cluster – compute nodes are added and removed based on work load
  - Between a max and min size
- Almost the same as using Tizard HPC
  - Same eRSA account, home directory, software applications, Torque queueing system, job submission, wall time limit, file transfer (sftp)
- Log in via ssh to `emu.ersa.edu.au`

Modifying a Tizard submission script for Emu is easy

```
#!/bin/csh

#PBS -V

### Job name
#PBS -N Experiment1

### Join queuing system output and error files into a single output file
#PBS -j oe

### Send email to user when job ends or aborts
#PBS -m ae

### email address for user
#PBS -M my.name@email.edu.au

### Queue name that job is submitted to
#PBS -q tizard

### Request nodes NB THIS IS REQUIRED
#PBS -l nodes=1:ppn=1
#PBS -l mem=4gb,vmem=1gb
#PBS -l walltime=01:00:00

# This job's working directory
echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR
echo Running on host `hostname`
echo Time is `date`

#Load module(s) if required
module load gnu/4.8.0
module load stacks

# Run the executable
process_radtags -P -p ~/data --renz_1 ecoRI --renz_2 mseI -c -q -s 20 -i gzfastq
```

```
#!/bin/csh

#PBS -V

### Job name
#PBS -N Experiment1

### Join queuing system output and error files into a single output file
#PBS -j oe

### Send email to user when job ends or aborts
#PBS -m ae

### email address for user
#PBS -M my.name@email.edu.au

### Queue name that job is submitted to
#PBS -q emu

### Request nodes NB THIS IS REQUIRED
#PBS -l nodes=1:ppn=1          - For Emu, always set nodes=1
### PBS -l mem=4gb,vmem=1gb    - Blank this line with ###
#PBS -l walltime=01:00:00

# This job's working directory
echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR
echo Running on host `hostname`
echo Time is `date`

#Load module(s) if required
module load gnu/4.8.0
module load stacks

# Run the executable
process_radtags -P -p ~/data --renz_1 ecoRI --renz_2 mseI -c -q -s 20 -i gzfastq
```

Limitations of Emu

- A good additional compute resource, but some limitations
- Only for small jobs, high-throughput computing
  - Single node jobs
  - Usually only 8 cores per job (or at most 16)
  - No more than 4GB memory per core
- Not as fast for file I/O as Tizard
  - Small /tmp space
  - slower /scratch and /home
- Smaller number of total cores than Tizard

Private compute nodes

- By default Emu is a shared resource like Tizard
- But can use your own NeCTAR cloud resource allocation to have private compute nodes
  - Like your own private sub-cluster!
  - No job wall-time limit!
  - No queue!*
- Log in to the NeCTAR web dashboard
- Apply for a project allocation
  - Ask the eRSA heldesk if you need assistance
- Once it is approved, email the eRSA helpdesk to say you want to use it with Emu

*queue with users in your project allocation only

Using your private cluster

- Tell eRSA helpdesk
  - the short name of your Nectar project allocation
  - which researchers can use your private nodes
  - what size worker nodes you would like
- eRSA will set it up for you and tell you when it's ready
- Add one line to your Torque job submission script for Emu to specify the name of your private group

```
#PBS -A myprojectname
```

- Now you have your own private sub-cluster on Emu!
  - Jobs will also go to the shared cluster if it has free worker nodes and yours doesn't

Service Desk and User Support

- Online user guides and help on eRSA web site http://support.ersa.edu.au
- For any question or problem, contact the eRSA helpdesk
  - http://www.ersa.edu.au/support/
  - Email servicedesk@ersa.edu.au (preferred)
  - Or call 7228 6236
- Email goes to a ticketing system so we can track your request and the right person responds, via email, phone or in person.
- Don't contact eRSA system administrators or support staff directly – always use the service desk email or phone


eRSA
Advancing Research Innovation