Introduction to NCI

http://nci.org.au/user-support/training/

help@nci.org.au

Gaurav Mitra

Agenda

- Introduction
- Accounts and Projects
- Connecting to Raijin
- Batch Processing on Raijin
- Filesystems

Agenda

What is NCI?

- Peak Facility
  - HPC system: Raijin
  - Cloud services
  - Data management
- Specialized Support
  - Staff Scientists
  - 5 full-time, 2 part time
  - Discipline-specific
  - Application-specific

Allocation Schemes

- National Computational Merit Allocation Scheme (NCMAS)
  - Highly-competitive, premier allocation scheme
  - Includes NCI (Raijin), Pawsey Centre (Magnus), Monash (MASSIVE), and UQ (FlashLite)
  - 15% share of Raijin
- Partner Shares
  - Government agencies, research centres, and universities
  - Each NCI partner has a share of the resources to distribute at their discretion
    - ANU Allocation scheme has 107.7 MSU in 2017 on Raijin
    - Applications open in Q4 each year
    - `anumas.nci.org.au`

Integrated Ecosystem

- World-class HPC system – Raijin
  - 2.6 PetaFLOP peak compute performance
  - 24th on Top500 list when built (121st, Nov 2016)
- Supercomputer-grade cloud infrastructure
  - Specialized *virtual laboratories*
  - Hosted data distribution services
- NCI-global Lustre Filesystems
  - Very high performance – up to 150GB/s read/write
  - Mounts available on HPC systems and NCI-managed virtual services

Integrated Ecosystem

- Large selection of software packages
  - Custom-compiled for the best performance on Raijin (where possible)
  - Includes several commercially licensed packages
- If there is one you need thats not already available on /apps, ask us!
  - Provided there is enough interest, we may install it in /apps
  - We can also help you install it in your own local space
- We provide the Intel and GNU GCC compiler suites for you to build your own
  - C/C++ and Fortran compilers
  - Performance libraries (IPP, MKL, MPI, TBB, ...)
  - Performance and debugging tools

HPC System - Raijin

- Fujitsu Primergy Cluster + Lenovo NeXtScale System
- 4472 compute nodes, 6 login nodes, 5 data movers
  - Sandy Bridge: 3592 nodes have 2× Intel Xeon E5-2670 (8 core, 2.6GHz Base)
  - Broadwell: 804 nodes have 2× Intel Xeon E5-2690 v4 (14 core, 2.6Ghz Base)
    - Thats $57,472$ Sandy Bridge cores + $22,512$ Broadwell cores...
  - 2/3 of the Sandy Bridge compute nodes have 32GB, 1/3 have 64GB, 72 have 128GB
  - 2/3 of the Broadwell compute nodes have 128GB, 1/3 have 256GB
    - Thats 158TB of RAM on Sandy Bridge and 100.5TB on Broadwell...
  - Sandy Bridge: 56 Gbit/s *fat-tree* FDR InfiniBand network
  - Broadwell: 100 Gbit/s *fat-tree* EDR InfiniBand network
    - Each node has a dedicated link back to the core of the network

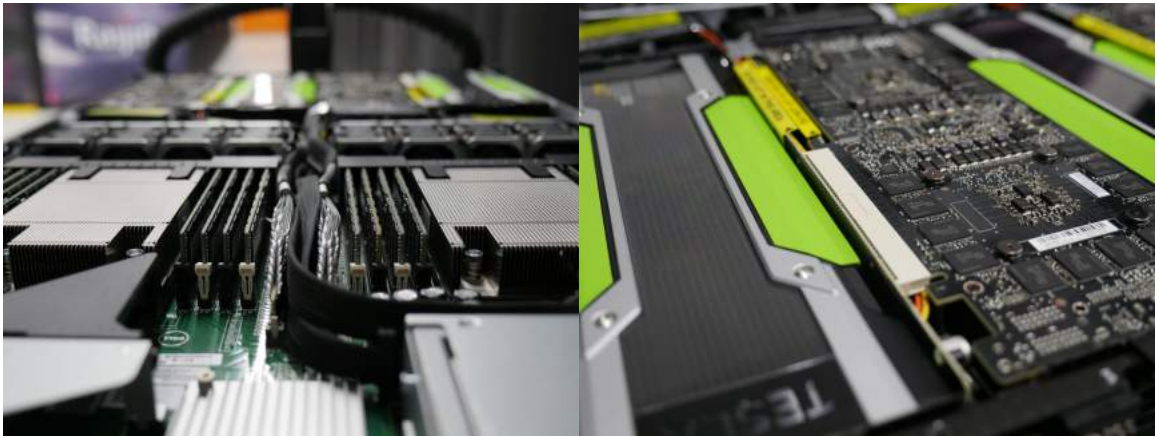
HPC System - Raijin

HPC System - Raijin

HPC System - Raijin

- 10 huge-memory nodes
  - 2 × Intel Xeon E5-2690 v4 (14 core, Broadwell, 2.66GHz)
  - 1TB RAM
- 30 GPU nodes (14 Haswell, 16 Broadwell)
  - Haswell: 2 x Intel Xeon E5-2670v3 (12 core, 2.3GHz)
  - Broadwell: 2 x Intel Xeon E5-2690v4 (14 core, 2.3GHz)
  - 4 x NVIDIA Tesla K80 GPUs (i.e. 8 x K40s) per node
  - 4992 NVIDIA CUDA cores per K80 (2496 per GPU)
  - 256GB RAM on host, 24GB RAM per K80
- 32 Intel Xeon Phi (Knights Landing) nodes
  - 1 x 64-core (256 threads with hyperthreading) Intel Xeon Phi 7230 CPU, 1.30 GHz base clock
  - 192 GB DDR4-2400 RAM (at 115.2 GB/s)
  - 16 GB on package high-bandwidth (at 380 GB/s) MCDRAM, used as L3 cache for the DDR4 RAM
  - 400 GB SSD local disk
  - 100 Gb/s InfiniBand interconnect between KNL nodes

HPC System - Raijin - GPU Node



HPC System - Raijin - KNL Cluster

HPC System - Raijin

- Primary Filesystems are all Lustre
  - Provided over the main InfiniBand network
  - Aggregate performance of up to 150GB/s
  - Single-node, single-OST performance is 1GB/s
- NCI-global Filesystems mounted at `/g/data1,2,3`
  - Also over the InfiniBand network
  - /g/data1 (54 GiB/s), /g/data2 (65 GiB/s) and /g/data3 (100GiB/s)
- Each node has a node-local disk (*jobfs*) for IOPS-intensive work
  - Sandy Bridge: 420GB HDD
  - Broadwell: 440GB SSD
  - Hugemem/KNL: 400GB SSD
  - GPU: 700GB SSD

Cloud Infrastructure - Nectar

- National eResearch Collaboration Tools and Resources
  - Every researcher at an Australian university has a small allocation
  - Can apply for more resources through Nectar
- NCI node is based on the same technology as Raijin Sandy Bridge
  - 2 x Intel Xeon E5-2670 (8 core, Sandy Bridge, 2.6GHz)
  - 56Gbit/s Ethernet network
  - SSD-backed ephemeral storage
  - Distributed, self-healing (Ceph) volume storage
- Designed for heavy floating-point computation, high-IOPS workflows, and high-speed data transfers

Cloud Infrastructure - Tenjin

- Exactly the same hardware as the Nectar cloud
- Our *private* cloud - available to NCI partners
- NCI-global Filesystems are available on request
  - Provided over multiple 10 Gbit/s Ethernet links via NFS
- Designed for services complementary to the HPC system
  - Exporting processed data sets to the world
  - On-demand (rather than batch) computation

Agenda

Applying for an account

- Go to `my.nci.org.au` and follow the prompts
- You need to use your institutional e-mail address
- You will be asked for a project code during the sign-up phase
- The Lead Chief Investigator (CI) of the project will be e-mailed for approval
- Once approved, a username will be generated and e-mailed to you
  - NCI usernames have the form abc123 or ab1234
  - Your username is used for logging in to most systems
- You can then log in to the appropriate systems

Joining another project

- You can join another project in addition to the one you signed up with
  - Will give you access to other allocations, data sets, software, ...
- Go to `my.nci.org.au`, log in, and follow the prompts
  - Shortcut: if the project code is *ab1*, go to `my.nci.org.au/mancini/project/ab1/join`
- The Lead CI of that project will be e-mailed for approval
- Your account will be automatically disabled once disconnected from all active projects

Proposing a new project

- You can propose a new project using `my.nci.org.au`.
- If you don't already have an account, you can also propose a project during signup
- You will need to specify which allocation scheme to apply under
  - Different schemes available: Partner, Startup (Max 4 KSU annually), MAS ...
  - Most schemes accept applications any time during the year
  - Notable exception is NCMAS, ANUMAS: Application round is open late in the year
  - ANU specific startup - ANU-Startup - Open any time during the year - Up to 100 KSU
- Each allocation scheme has it's own requirements

Resource allocation and accounting

- All usage of compute systems is accounted against projects
- If your account is connected to multiple projects, a default project will be debited unless another is specified
  - *project* attribute for PBS jobs
  - group ownership for filesystem objects
- Compute allocations on Raijin are applied on a quarterly basis
  - Unused time at the end of the quarter is lost
- Storage allocations are generally persistent
- A project may be funded by multiple allocation schemes
- Compute usage of a job is debited only when the job finishes or is terminated

Agenda

Connecting to Raijin

- The hostname for Raijin is `raijin.nci.org.au`
  - This will connect you to one of the 6 login nodes
- All interactive access to Raijin is command-line based via SSH
- UNIX-based operating systems (Linux, Mac OS X) have SSH built in
  - `ssh abc123@raijin.nci.org.au`
- Windows users will need to install a client
  - PuTTY, MobaXterm, Cygwin, ...
  - You may need to get your local ITS to install it for you

Connecting to Raijin

- File transfers also need to be performed via SSH
  - scp, sftp, rsync, ...
- For UNIX-like operating systems, these are probably already installed
- For Windows, youll need to install a client
  - These typically have nice GUIs
  - PSFTP, FileZilla, WinSCP, ...
- You should use the dedicated data-mover nodes, `r-dm.nci.org.au` for large file transfers
  - The Filesystems are mounted exactly as on the login and compute nodes
  - But you won't be able to start interactive sessions here

Connecting to Raijin

- If you to run a graphical application on Raijin and have the GUI open on your local machine, youll need to enable X-forwarding
- Youll also need to be running an X server locally
  - Linux and Max OS X 10.7 and below have this installed already
  - Mac OS X 10.8 and above need XQuartz
  - There are many Windows clients: MobaXterm, Xming, Xwin32, ...
- For UNIX-like operating systems, add -X to the ssh command
- For Windows, consult the documentation for your client

Connecting to Raijin: Exercise!

- Get a username from the list
- Use the password provided by the instructor
- Connect to Raijin and have a look around
  - `ls` will list the contents of the current directory
  - `df` will show mounted Filesystems (and their size)
  - `cd` will change directory
  - `env` will display your environment variables

Default Environment

- There's a second line in your `.rashrc` defining your default project
- You can change your working project at any time
  - `switchproj c25`
- You can also run a single command under another project
  - `nfnewgrp c25 cat /short/c25/my_file_under_c25`
- Of course, you must be part of that project for these to work...

Allocation Management

- You can easily view the status of your project allocations from the command line
- `nci_account [ -P c25 ] [ -p 2016.q4 ] [ -v ]`
  - `-P` specifies the project (uses your current project if not present)
  - `-p` specifies the period (remember, quarterly compute allocations)
  - `-v` produces more detail such as compute allocation usage per user
- **Exercise:** Look at the current allocation for project c25

Configuring your default environment

- You can customize your default environment by editing special files in your home folder
- There are two files – one controls login shells, the other non-login
  - A login shell is launched when connecting via SSH
  - A non-login shell is launched whenever you invoke a shell otherwise
- You generally want to keep the non-login shell configuration very simple – it gets parsed more often than you'd think

	Login Shells	**Non-login Shells**
sh and derivatives (sh, bash, ksh, zsh)	`.profile`	`.bashrc`
csh and derivatives (csh, tcsh)	`.login`	`.cshrc`

Environment Modules

- Different software packages have different environments
- Environment modules allow us to package these environments
- Modules on Raijin are named after the package and the version
- The `module` command allows you to manage your environment
- Further information: `module help` or `man module`
- **Exercise:** Take a look at available modules and load the `openmpi` module
  - Look at `module avail` and `module load`
  - Always `module load` a **specific version**

- We recommend loading modules as needed, both interactively and in your scripts
- If you really want particular modules loaded on login, add this to your `.profile` file
  - Adding them to your `.bashrc` will have unexpected results
- This is due to dependencies and conflicts between various modules.
- **Exercise:** Assuming the `openmpi` module is still loaded from before, try to load the `intel-mpi` module

Editing Text Files on Raijin

- There are several command-line based text editors on Raijin
  - vi / vim
  - emacs
  - nano
- Which to use is up to you!
  - I personally recommend vim
- You can also edit files on your local machine and upload them
  - But keep in mind that Windows uses a different new line character to UNIX
  - Need to run `dos2unix` on Raijin to convert once uploaded

Agenda

Interactive Limits

- Typically more than 100 users connected to each login node
- *Only* 96GB of RAM in each login node – less than 1GB each
- To avoid running out, we limit user processes to 2GB
- Also limit process CPU time to 30 minutes
- Most programs need more than this – use the batch queues
  - Small test cases are okay on the login nodes
  - But still be careful – even with limits, easy to use all RAM

Batch Queues

- Lots of jobs in the queue
  - Some small, some big
  - `http://nci.org.au/user-support/current-job-details/`
- The queuing system has several advantages
  - Distributes jobs evenly over system
  - Ensures jobs don't impact each other
  - Provides equitable access to all users (based on allocation)
- We run PBS Professional (version 13) on Raijin
  - Well-defined API, the same across all PBS implementations
  - We also have our own custom integration between PBS Pro and Raijin

Batch Queues: Exercise

- Interact with the batch system and see what is running
  - Make sure you have the pbs module loaded first: `module list`
  - The `qstat` command will list all jobs on the system
  - Using `qstat -a` will give an alternative view
  - Might want to pass the output to less: `qstat | less`
    - To scroll, use arrow keys or the space bar
    - To exit `less`, press q
  - Alternative commands: `nqstat, nqstat_anu`
    - `nqstat` updates every 30 seconds, and covers both queued and executing jobs
    - `nqstat_anu` updates instantaneously and covers executing jobs

Batch Queues

- Not all jobs look the same – multiple queues
- *normal* queue
  - For general, everyday jobs on Sandy Bridge nodes
  - Charged at 1 SU per core-hour (i.e. walltime x ncpus)
- *normalbw* queue
  - For general, everyday jobs on Broadwell nodes
  - Charged at 1.25 SU per core-hour (i.e. walltime x ncpus)
- *express* queue
  - For quick-turnaround jobs on Sandy Bridge nodes, e.g. interactive or debugging
  - Charged at 3 SU per core-hour
- *expressbw* queue
  - For quick-turnaround jobs on Broadwell nodes, e.g. interactive or debugging
  - Charged at 3.75 SU per core-hour

Batch Queues

- *copyq* queue
  - Runs on data-mover nodes, has access to external resources
  - Charged at 1 SU per core-hour
- *hugemem* queue
  - Runs on a *huge-memory* node
  - Charged at 1.25 SU per core-hour
- *gpu* queue
  - Runs on the GPU nodes
  - Charged at 3 SUs per core-hour
- *knl* queue
  - Runs on the KNL nodes
  - Charged at 0.25 SUs per core-hour

Queue Limits

- The various queues have different limits based on their purpose
- These are generally flexible, within reason
- If you need them changed, ask us
  - Well probably ask you to explain why you need the exception

Queue Limits

Queue	Jobs in Execution	CPU/GPU	Walltime
normal/normalbw	300 per project	56960 CPU (SB), 22512 CPU (BW) Multiple of 16 above 16 (SB) Multiple of 28 above 28 (BW)	48 hours for 1-256 CPUs 24 hours for 256-511 CPUs 10 hours for 512-1023 CPUs 5 hours for 1024-56960 CPUs
express/expressbw	50 per project 10 per user	200 Multiple of 16 above 16	24 hours for 1-160 CPUs 5 hours for 176-3200 CPUs
copyq	200 per project	1 CPU	10 hours
hugemem	200 per project	28 CPUs Minimum of 6 CPUs	96 hours for 1-6 CPUs 48 hours for 7-12 CPUs 32 hours for 13-18 CPUs 24 hours for 24 CPUs
gpu	20 per project	144 CPUs. Multiple of 2 GPUs Multiple of 6 CPUs	48 hours
knl	20 per project	2048 CPUs	48 hours

Exercise: Find your limits!

```
nf_limits -P project -n ncpus -q queue
```

Example Job Script

```
#!/bin/bash
#PBS -l walltime=00:01:00
#PBS -l mem=1GB
#PBS -l jobfs=1GB
#PBS -l ncpus=4
#PBS -q expressbw
#PBS -P c25

echo ''-----------''
echo ''TOTAL CPUS''
echo ''-----------''
cat /proc/cpuinfo | grep processor | wc -l
echo ''-----------''
echo ''MEM INFO''
echo ''-----------''
free -g
echo ''-----------''
echo ''CPUS ALLOWED''
echo ''-----------''
cat /proc/self/status | grep Cpus_allowed_list
echo ''-----------''
qstat -f $PBS_JOBID | egrep 'used|exec_host'
```

Interacting with PBS

```
# To interact with PBS, load the 'pbs' module
# The three most useful commands:

# 1) qsub: Submit a job
> qsub myscript.sh
# Returns the job ID

# 2) qstat: Get the status of job(s)
> qstat # All jobs
> qstat 12345 # Just job 12345
> qstat -u abc123 # Jobs of user abc123

# 3) qdel: Delete a job
> qdel 12345
```

Interacting with PBS: Exercise

```
# Submit a job to PBS and wait for it to finish

# Create a simple job file (emacs is OK too :)
> vim runjob

# Submit the job
> qsub runjob

# Look at job details
> qstat -f <jobID>
```

Job Outputs

- The standard out and error streams of your script are collected by PBS
- These get saved to files in the submission directory on exit
  - `<name>.o<jobid>` for standard out
  - `<name>.e<jobid>` for standard error
- You can also redirect the output from individual commands
- **Exercise:** Have a look at the output files from the previous exercise

Interactive Jobs

- Some times you need to interact with a job as it is running
  - For example, using the MATLAB desktop
- You can submit an interactive job using the `-I` option to qsub
- If you need X windows forwarded from the job, add the `-X` option
- **Exercise:** Submit an interactive job
  - `qsub -I -l ncpus=2,mem=1G,walltime=00:15:00 -q expressbw`
  - Have a look around the compute node

Raijin Job History

https://usersupport.nci.org.au/report/job_history

Agenda

Filesystems on Raijin

Mount	Purpose	Default Quota	Backup	Availability	Persistence
/home	Irreproducible data e.g. Source code, scripts	2GB per user	Yes	Raijin	Permanent
/short	Working data	72GB per user	No	Raijin	365 days
/g/data1,2,3	Large data sets	Negotiable	No	NCI-global	Permanent
$PBS_JOBFS	Job-specific data	100MB per job	No	Node	Jobs
MDSS	Archiving	Negotiable	Dual-copy	Unmounted	Permanent

- There are several Filesystems available on Raijin
- Which to use depends on the files you are storing
- Not all projects have access to all Filesystems

Monitoring Disk Usage

- If you exceed a project quota on any filesystem, your access to PBS is suspended
- You will get automated e-mails regarding your usage
  - A warning at 90% disk usage
  - A monthly reminder for exceeding 90%
  - A message at 100% asking you to reduce your usage
  - Daily reminders while above 100%
- Be proactive about monitoring your usage

Monitoring Disk Usage: Exercise

```
# Have a look at your usage on the various Filesystems

# Query Lustre for current usage
> lquota

# What our accounting and PBS systems sees
> nci_account

# Breakdown of usage by user
> short_files_report
> gdata1_files_report # Similar for gdata2, gdata3
```

I/O: The good, the bad, and the ugly

- Writing to `/short` every second is **far too often**
- If you program does this:
  - Change the program if possible
  - Otherwise use the node-local disks (jobfs)
- Since jobfs is not shared, there is no locking overhead
- Filesystem cache also much more effective
  - Disk-memory = 100MB/s, memory-memory = 12GB/s
- You can request space on jobfs using the `-ljobfs=xxx` PBS option
  - Inside a job, the path to jobfs is in the `PBS_JOBFS` environment variable

Filesystem Permissions

- POSIX permissions are the standard way of controlling access
- Have read, write, and execute permissions
- *user*, *group*, and *world* permission sets
- Extra, special permission bits for other behaviour
  - setuid, setgid, sticky, restricted delete, ...
- Often expressed as a string like `rwxr-xr-x`
- Use `chmod` to change these permissions
  - `chmod u+w,g=rx,o= my_file`
  - Can also express this as a sequence of octal numbers

Filesystem Permissions

- Can assign more fine-grained permissions using ACLs
  - Give specific user access to file, even though not in the group
  - Give another group read permission but not write
- Highly recommend you consult with us first
  - **Very easy to get it wrong** and leave your files open to the world
- Use the `setfacl` command to set them:
  - `setfacl -m u:abc123:rw my_file`
- Use the `getfacl` command to view them:
  - `getfacl my_file`

Massdata Storage System

- Our *massdata storage system* consists of a large tape library with a 1PB cache in front.
- Used for long term storage of large files
  - If you have lots of small files, tar them up first
- Not mounted as a filesystem on Raijin
  - **Its not designed for constant read/write**
- Access is via `mdss` command
  - `mdss get`
  - `mdss put`

Data Recall from Tape

Conclusion

```
# Thank You!
# Questions?
# Wiki:      opus.nci.org.au
# Helpdesk:  help.nci.org.au
# Email:     help@nci.org.au
```