INTERSECT

# Using Intersect's partner share of Raijin

Dr. Joachim Mai

Oct 2013

# Getting Resources

- Resource allocation round in Oct for the coming year.
- Smaller allocations (<20 kSUs per quarter) possible any time
- Use the forms:
  - http://nf.nci.org.au/accounts/forms/user_registration.php
  - https://nf.nci.org.au/accounts/projects_new/APP_form.php
- Connect to existing project:
  - http://nf.nci.org.au/accounts/forms/user_connection.php

# Account Details

- Once the CI approves your connection to the project, your account is set up and you are sent an email with account details.
- User names are of the form abc123 - abc for your initials and 123 for partner.
- Passwords are sent by SMS to the mobile number provided when you registered.
- Passwords can be given over the phone if necessary, but not by email.
- Use the passwd command to change this when you first log in.

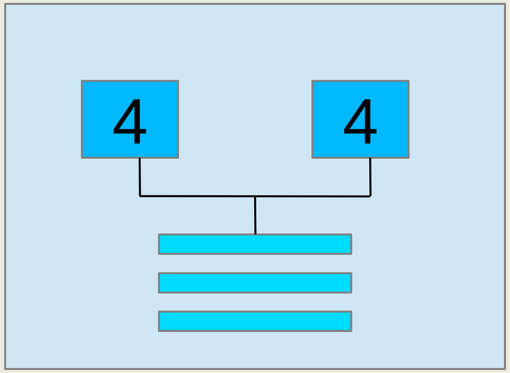
# Raijin (July 2013)

# Raijin: Overview

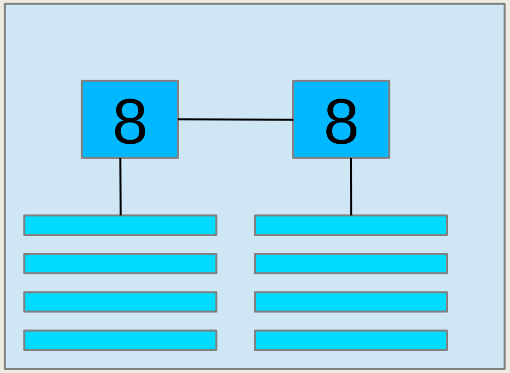
- Fujitsu Primergy
- Distributed memory machine
- 57,472 Sandy Bridge cores
- 158 TB RAM
- 9 PB disk
- Infiniband FDR
- Centos 6.4
- Intersect owns about 3.8%

# Vayu and Raijin Node Layout

Nehalem/Westmere
Vayu



Sandy Bridge/Ivory Bridge
Raijin, Orange



# Raijin: CPUs

- CPU: Intel Sandy Bridge E5-2570
- 2.6 GHz. Turboboost up to 3.2 GHz
- 4 memory channels (Vayu: Nehalem, 3 channels)
- 2 CPUs per node
- 8 cores per CP (Vayu: 4)
- 16 cores per node (Vayu: 8)
- OpenMP program can now use up 16 cores.
- But: asymmetric memory access
- But: do scaling experiments and see whether your code can make good use of it.

# Remark: Orange (Dec 2012)

- SGI Cluster
- Distributed memory
- 1,600 Sandy Bridge cores
- 10 x 256GB RAM
- 90 x 64 GB RAM
- 100 TB disk
- Infiniband QDR
- SLES 11

# Differences Orange ↔ Raijin

- Memory:

  Orange: 90 x 64 GB<br>
  10 x 256 GB

  Raijin: 2395 x 32 GB<br>
  1125 x 64 GB<br>
  72 x 128 GB

- Software
- Size: 100 nodes (Orange) vs 3,592 nodes (Raijin)

# Raijin Stakeholders

- NCMAS 15%
- CSIRO 21.4%
- BOM 18.9%
- ANU 17.7%
- Flagships 5.0%
- INTERSECT 3.8%
- GA 3.4%
- Monash 1.7%
- UNSW 1.7%
- UQ 1.7%
- USyd 1.7%
- University of Adelaide 1.7%
- Director's share and others 4%

# Disks Raijin

- Lustre parallel global FS
- 10 PB
- 300GB scratch disks per node

# Project Accounting

- Each user belongs to one or more projects
- To change or set the default project, edit your .rashrc file in your home directory, and change the PROJECT variable as desired. A typical .rashrc file looks like

```
setenv PROJECT c25
 setenv SHELL /bin/bash
```

- You can also use the script: switchproj project-code
- Display usage:

```
nci_account -P project -p 2013.q3 –v
```

- On Vayu was:

```
quotasu
```

# Connecting to Raijin

- ssh -Y username@raijin.nci.org.au
- For connecting under Windows use putty
- For file transfer use scp, sftp or a GUI client such as Filezilla and upload.
- Upload larger data to r-dm.nci.org.au (not the login node of Raijin)

# Setting the Environment

| Command | Description |
|---|---|
| `module list` | To see the modules loaded |
| `module avail` | To see available modules |
| `module show` | To load the environment settings package |
| `module load` | Modifies the attributes of the job or jobs |
| `module unload` | To remove a previously loaded software package. This is useful in situations here different package settings clash (multiple MPIs for example) |

# Exercise 1

1. Log in to Raijin. Your project code is c25 and you username is aaa777 (bbb777, etc.).
2. Read the message of the day (MOTD).
3. Try the following commands:

| Command | Description |
|---|---|
| `hostname` | To see the node you are logged in to |
| `nci_account` | To see the current state of the project |
| `printenv` | To look at your environment settings |
| `module list` | To check which modules are loaded on login |
| `module avail` | To see which software packages are installed |
| `module show pbs` | To see what environments are set by a module |

# Getting information and help

- Message of the Day - MOTD
- Downtime http://nf.nci.org.au/notices_news/
- User Guide at http://nf.nci.org.au/wiki/RaijinUserGuide
- FAQs at http://nf.nci.org.au/facilities/faq
- Software pages at http://nf.nci.org.au/facilities/software
- Email to hpc_support@intersect.org.au or help@nf.nci.org.au for NCI related problems

# Which file-system to use

Source code and important input files: /home

Job input/output files: /short

Temporary or scratch files: JOBFS

Long term archived files: MDSS

Processing of large data files: /projects

# Filesystem properties and limits

/home: for program, 2GB default, backup, global availability, permanent

/short: for I/O, 80GB, no backup, global availability

/JOBFS: for heavy I/O100MB, no backup, local to node, during job

MDSS: for archiving, 20GB, 2 copies, external accessing, permanent

# Monitoring disk usage

```
$ lquota
-------------------------------------------------------
fs Usage Quota Limit iUsage iQuota iLimit
-------------------------------------------------------
mhk900 home 20.7MB 0kB 0kB 421 0 0
z00 short 1153GB 0kB 1500GB 1527750 0 10000000
z10 short 0kB 0kB 78.0GB 1 0 200000
c25 short 428kB 0kB 200GB 107 0 200000
y03 short 557GB 0kB 10.0TB 163891 0 1000000
z29 short 0kB 0kB 78.0GB 0 0 200000
ua6 short 0kB 0kB 195TB 0 0 200000
c25.data short 0kB 0kB 0kB 0 0 0
z34 short 636kB 0kB 78.0GB 29 0 200000
-------------------------------------------------------
```

# Optimise usage

Lots of small IO to /short (or /home) can be very slow and can severely impact other jobs on the system.
Avoid "dribbly" IO, e.g. writing 2 numbers from your inner loop.
Writing to /short every second is far too often!

Avoid frequent opening and closing of files (or other file operations)

Use /jobfs instead of /short for jobs that do lots of file manipulation

To achieve good IO performance, try to read or write binary files in large chunks (of around 1MB or greater)

# Exercise 2

- Use the commands lquota and du to determine the disk space available to you in /home and /short.
- Have a look at your /short. Any user of the project can write data here.

```
cd /short/$PROJECT
ls -ld .
ls -l DATA
ls $USER
```

Extract the examples here:

```
 cd $USER
tar xvf /short/c25/intro_exercises.tar
cd INTRO_COURSE
pwd
ls -l
```

# Exercise 2 cont.

1. Change the permissions on your files and directories to allow/disallow others in your group to access them.

| Command | Description |
|---|---|
| `man chmod` | |
| `chmod g+r filename` | Allow group read to filename |
| `chmod g-r filename` | Disallow group read to filename |
| `chmod g+w filename` | Allow group write to filename |
| `chmod g+x filename` | Allow group execute to filename |

# Exercise 2 (cont)

2) Use the MDSS with the following commands:

```
cd /short/$PROJECT/$USER
mdss get Data/data.tar
ls -l
tar xvf data.tar
ls
rm data.tar
mdss mkdir $USER
```

Note that this creates a job in the copy queue which you can monitor

```
netmv -t $USER.tar DATA $USER  nqstat
more DATA.o*
mdss ls $USER
mdss rm $USER/$USER.tar
```

# Compiling and Optimizing

- We recommend using the Intel compilers (icc, ifort, icpc).
- Check which versions are available:

```
module avail intel-fc
module avail intel-cc
```

- Read the manual pages (man icc, man ifort, man icpc)
- Compiling and linking of a Fortran code:

```
ifort -o matmulf matmul.f
```

- Options: default is O2 for Intel and O0 for Gnu compilers
- O0 means no optimisation and is very slow (-g implies -O0)
- O3 means aggressive optimisation, be careful.

Best performance with -xHost (Orange: -O3 -xAVX). This needs a never version of the compiler which makes use of the vector features.

# Compiling and Optimizing

Default versions:

- Intel v12. Some problems with v13. Try it!
- Brand new: v14

- Gnu v4.4. Vector features need 4.7. Load module for that.

- OpenMPI: 1.6.3
  - Also available: 1.4.3
  - Normally: use newer versions. If that causes problems use the old one.

# Exercise 3

1) Compile the sample code matmulf.f  and matmulf.c with the Intel compilers using O3 and O0 optimisation. Compare the runtime using the time command.

2) Compile the code netcdfex.f using the netcdf library:

```
module load netcdf
module show netcdf
module list
cat netcdfex.f
ifort -o netcdfex netcdfex.f -lnetcdff -lnetcdf
netcdfex
ncdump simple_xy.nc
```

# The batch system PBSPro

The batch system distributes work evenly over the system
and ensures that jobs cannot impact each other (e.g. exhaust memory or other resources)

Raijin uses a customized version of PBSPro on Raijin. It is currently being tuned.

Batch queues:
normal: default
express: 3 times the charges of Sus
copy: to copy data, e.g. to MDSS

Walltime limits: 48h (1-255 cores), 24h (256-511 cores), 10h (512-1024 cores), 5h (>1024 cores).

# Using PBS

- Submit your job via qsub
- Specify resources (walltime, mem, nodes PBS_JOBFS etc).
- Read the handbook!!!

Scheduling

- Jobs start when sufficient resources get available
- Jobs can be suspended when jobs with higher priority wait
- Priority depends on the amount of resources available

Use check pointing for longer running jobs

# Differences to ANUPBS

- Use mem (not vmem)

- Use -l wd (not -wd)

- select and nodes do not work at the moment. Use ncpus as on Vayu)

-CPU sets are not supported in the current version of PBSPro. So 2 small mpijobs in one node could start on the same CPUs. to avoid this use:

```
mpirun -np 8 -bind-to-none prog.exe
```

This is a working progress. Expect changes in the future while NCI is working with Altair to further optimize and customize PBSPro.

# Monitoring Jobs

## Useful Commands

| Command | Description |
|---|---|
| `qstat` | Show the status of the PBS queues |
| `nqstat` | Enhanced display of the status of the PBS queues |
| `qstat -s` | Display additional comment on the status of the job |
| `qps jobid` | Show the processes of a running job |
| `qls jobid` | List the files in a job's jobfs directory (to come) |
| `qcat jobid` | Show a running job's stdout, stderr or script |
| `qcp jobid` | Copy a file from a running job's jobs directory (To come) |
| `qdel jobid` | Kill a running job |

# Exercise 4

1) Submit the following PBS script via qsub:

```
#!/bin/csh
#PBS -l wd
#PBS -q express
#PBS -l walltime=00:10:00,mem=52MB
#PBS -P c25
time ./matmuls
time ./matmulf
```

2) Monitor it's progress via:

```
nqstat
qps jobid
pbs_rusage jobid
```

# Exercise 4 (cont)

3) Try an interactive batch job:

```
[aaa777@raijin5 ~]$ qsub -I -l walltime=00:10:00,mem=500Mb,wd -P c25
qsub: waiting for job 215984.r-man2 to start
qsub: job 215984.r-man2 ready
[aaa777@r73 ~]$ hostname
r73
[aaa777@r73 ~]$ module list
Currently Loaded Modulefiles:
1) pbs 3) intel-cc/12.1.9.293
2) intel-fc/12.1.9.293 4) openmpi/1.6.3
[aaa777@r73 ~]$
[aaa777@r73 ~]$
qsub: job 215984.r-man2 completed
```

# Exercise 5

1) Compile and run the following OpenMP code

```
ifort -O3 -openmp matmul_omp.f -o matmul_omp
export OMP_NUM_THREADS=2
time matmul_omp
export OMP_NUM_THREADS=4
time matmul_omp
```

# Exercise 5 (cont)

2) Compile and run the following MPI code

```
module list
mpif90 mpiexample1.f -o mpiexample.exe
mpirun -np 4 mpiexample.exe
```

or for a more complicated example:

```
mpif90 mpiexample2.f -o mpiexample.exe
mpirun -n 4 mpiexample.exe
```

mpirun is the usual instruction to start an MPI program.

```
man mpirun
```

for further details on usage.

C code simple example:

```
mpicc mpiexample3.c -o mpiexample.exe
mpirun -np 4 mpiexample.exe
```

and for a more complicated code:

```
mpicc mpiexample4.c -o mpiexample.exe
mpirun -n 4 mpiexample.exe
```

# Profiling

On a system level use: top, iostat, vmstat
On PBS level use: qstat, qstat -f, qps etc.

Lightweight: IPM
Heavyweight: Vampir

General profiling:

```
$ ifort -p -o prog.exe jacobi_serial.f
$ ./prog.exe < input.1
$ gprof ./prog.exe gmon.out
```

For the GNU compilers do

```
$ gfortran -pg -o prog.exe jacobi_serial.f
$ ./prog.exe < input.1
$ gprof ./prog.exe gmon.out
```

# Example:use of PBS_JOBFS

```
#!/bin/bash
#PBS -q express
#PBS -l walltime=2:00
#PBS -l jobfs=10mb
#PBS -l mem=30mb
cd $PBS_JOBFS
echo "Moving files from short directory to the local
  directory"
cp /short/$PROJECT/$USER/input.1 .
cp /short/$PROJECT/$USER/jacobs .
# Run program and write an output file to the local disk.
time ./jacobs <input.1 > output$PBS_JOBID 2>&1
# Move output data to /short space.
echo "The output files are now on my /short space."
mv output$PBS_JOBID /short/$PROJECT/$USER
# Archive to MDSS using netcp
cd /short/$PROJECT/$USER
netcp -N save_data output$PBS_JOBID $USER/output$PBS_JOBID
```

# Acknowledgement

This course is based on a course created by Margaret Kahn/NCI and presented to the BOM. Margaret and the NCI staff is greatly acknowledged for help, support and discussions.

# Thanks for attending!

Please complete our course survey at:

http://svy.mk/18c8dHa

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

INTERSECT