RAIJIN EXERCISES

Exercise: Log in

Log in to raijin via ssh:

```
$ ssh username@raijin.nci.org.au
```

Use git to download the exercise material:???

Change into the downloaded directory:

```
$ cd Introductory-Supercomputing
```

List the contents of the directory:

```
$ ls
```

- What are the message of the day announcements?
- What directories and files are in the exercises?

Exercise: Modules

Try running the following module commands:

```
$ module avail
$ module load gcc
$ module list
$ module load hdf5
$ module show hdf5
$ module swap gcc intel-fc
$ module list
```

- What paths are set by the hdf5 module?

Exercise: Queues

Examine the queues and jobs on raijin:

```
$ qstat
```

or

```
$ nqstat -a
$ qstat -f jobid
```

- What is the largest job running at the moment?
- What is the largest job queued and when is it estimated to start?
- What is the most common reason for jobs not running?

Exercise: Hostname

Launch the job with `qsub`:

```
$ cd hostname
$ qsub hostname.pbs
```

Use `qstat` or `nqstat` to see if it is in the queue:

```
$ qstat -u username
```

It is a short job so it has probably run already.

If it is still in the queue can you work out why?

Examine the pbs-***jobid***.o file:

```
$ cat pbs-jobid.o
```

Which node did the job run on?

Serial python example

The following job submission script `hello-serial.pbs` will run a python job on a single core in normal queue for up to 5 minutes:

```
#!/bin/bash
#PBS -q express
#PBS -l walltime=00:05:00
#PBS -l ncpus=1
#PBS -l mem=1GB
#PBS -l jobfs=1GB
#PBS -l wd

# load modules
module load python3/3.6.2

# launch serial python script
python3 hello-serial.py
```

The script can be submitted to the scheduler with:

```
$ qsub hello-serial.pbs
```

OpenMP example

This will run 1 process with 16 threads on raijin, using 16 cores for up to 5 minutes:

```
#!/bin/bash
#PBS -q express
#PBS -l walltime=00:05:00
#PBS -l ncpus=16
#PBS -l mem=1GB
#PBS -l jobfs=1GB
#PBS -l wd

# set OpenMP environment variables
export OMP_NUM_THREADS=16
export OMP_PLACES=cores
export OMP_PROC_BIND=close

# launch OpenMP program
./hello-openmp-gcc
```

The program can be compiled and the script can be submitted to the scheduler with:

```
$ cd hello-openmp
$ make –f Makefile.gcc
$ qsub hello-openmp-gcc.pbs
```

The directory also contains the intel-equivalent files.

MPI example

This will run 28 MPI processes on 1 node on raijin:

```
#!/bin/bash
#PBS -q express
#PBS -l walltime=00:05:00
#PBS -l ncpus=16
#PBS -l mem=1GB
#PBS -l jobfs=1GB
#PBS -l wd

# prepare MPI environment
module load openmpi

# launch MPI program
mpirun ./hello-mpi
```

The script can be submitted to the scheduler with:

```
$ cd hello-mpi
$ module load openmpi
$ make
$ qsub hello-mpi.pbs
```

Launching parallel jobs

The following job submission script parallel-raijin.pbs will run a nwchem job on 32 MPI processes on 2 nodes in raijin normal queue for up to 5 minutes:

```
#!/bin/bash
#PBS -q express
#PBS -l ncpus=32
#PBS -l mem=64GB
#PBS -l jobfs=10GB
#PBS -l walltime=00:05:00
#PBS -l wd

# prepare MPI environment
module load nwchem/6.6

# launch MPI program
mpirun nwchem C60.nw > ${PBS_JOBID}.log
```

The script can be submitted to the scheduler with:

```
$ qsub parallel-raijin.pbs
```

Parallel Julia example

The following job submission script julia-mpi.pbs will run a Julia job on a single core in normal queue for up to 5 minutes:

```
#!/bin/bash
#PBS -q normal
#PBS -l ncpus=32
#PBS -l mem=64GB
#PBS -l jobfs=800GB
#PBS -l walltime=00:05:00
#PBS -l wd

# load modules
module load julia/0.6.0
```

```
$ cat example.jl
cpus = parse(Int64,ARGS[1])
fPath = ARGS[2]
using MPI
manager = MPIManager(np=ncpus)
addprocs(manager)
@everywhere using NetCDF
Y = @parallel (append!) for f in readdir(fPath)
  ni = ncinfo(joinpath(fPath,f))
  data = . . .
  ncclose(fname)
  data
end
using JLD
save("Temp.jld","temp",Y)
```

```
# launch julia script
julia example.jl ${PBS_NCPUS} ${INPUT_DIR}
```

The script can be submitted to the scheduler with:

```
$ qsub julia-mpi.pbs
```

Exercise: Run a job

Run hello-serial.py on a raijin compute node.

Move into the exercise directory:

```
$ cd hello-serial
```

View the submission script:

```
$ less hello-serial.pbs
```

Submit the script to the PBS scheduler:

```
$ qsub hello-serial.pbs
```

Check the queue:

```
$ qstat -u username
```

View the output:

```
$ less pbs-#jobid.o
```

Exercise: Interactive session

Run hello-serial.py interactively on a raijin compute node.

Start an interactive session (you may need to wait while it is in the queue):

```
$ qsub —I —q express —l wd
```

Prepare the environment:

```
$ module load python/3.6.3
```

Launch the program:

```
$ python3 hello-serial.py
```

Exit the interactive session:

```
$ exit
```

Exercise: Backing up

Practice data transfer by backing up the course material.

Use `scp` to copy via a data transfer node:

```
$ scp –r
username@r-dm.nci.org.au:/short/c25/username/ .
```