FIT3142 student use of the MCC Production HPC Cluster October 2016

The purpose of this document is to introduce you to the Monash Campus Cluster, the university's production HPC cluster. Operating since 2005, the Monash Campus Cluster (MCC) is available for access to its researchers (from professors, post-docs, PhD students and even honour students doing research work). Over the years, it has contributed towards advancing research in various fields including: Engineering, Physics, Chemistry, and IT. Further information on MCC may be found from:

https://confluence.apps.monash.edu/display/MCC/The+Monash+Campus+Cluster

IMPORTANT! You have been granted access to the MCC for the purposes of FIT3142, and your usage is subject to **fair use**. This implies (a) you may use your account for work relating to this unit only; (b) you will limit the number of running jobs to four; and (c) you are to limit your file storage to 1024 MB. Failure to comply will entail loss of access and may lead to appropriate disciplinary action. As the MCC is a university facility, your access and usage is subject to the IT policies.

http://www.policy.monash.edu/policy-bank/management/its/acceptable-use-of-information-technology-facilities-by-students-policy.html

In particular, you are never to share your password and your MCC account with another person. Your tutor and lecturer will never need your password to access your account, as we have admin access to check your work, if this is necessary.

Your MCC accounts will be disabled and archived shortly after the end of the semester.

A. How to access the Monash Campus Cluster

Your individual MCC account is accessible using your Monash ID, i.e., your **authcate** username and password. The hostname of the login node is:

```
msgln6.its.monash.edu.au
```

On Windows, you will need to install the SSH client **`putty`**. From OSX or Linux, you only need a terminal and use the pre-installed **`ssh`** command. For uploading and downloading files, you will need an appropriate secure copy or secure FTP client. Further information may be found at:

https://confluence.apps.monash.edu/display/MCC/Access

This server is available for access AOE (anywhere on earth); and is online 24x7, unless there is an outage or scheduled maintenance.

PLEASE NOTE: The login node is intended for interactive tasks such as file and directory management, editing your code and other files, compiling, and running short test runs; as well as uploading/download files. Please refrain from running parallel jobs that longer than 15 minutes on this login node, as it will impact other users.

B. Compiling your MPI programs

There are several version of OpenMPI installed on the system. From your secure shell session on the login node, select the recommended version of OpenMPI using the module load command, as demonstrated below (The text in **boldface** shows what you need to type into the session. The actual prompt will be slightly different on your account) :

```
[jsmith@msgln6 ~]$ module load openmpi/1.8.4-gcc-4.8.0

[jsmith@msgln6 ~]$ which mpicc
/opt/sw/openmpi-1.8.4-gcc-4.8.0/bin/mpicc

[jsmith@msgln6 ~]$ which mpirun
/opt/sw/openmpi-1.8.4-gcc-4.8.0/bin/mpirun
```

Consider this simple **MPI_hello.c** program:

```
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

int main(int argc, char **argv)
{
  int rank, nprocs, len;
  char name[MPI_MAX_PROCESSOR_NAME];

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&nprocs);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  MPI_Get_processor_name(name, &len);

  printf("Hello from rank %d of %d on node %s\n", rank, nprocs, name);
  fflush(stdout);

  MPI_Finalize();
  return 0;
}
```

For your convenience, we have prepared a folder "**hello**" containing this file along with other files you will need to try out the cluster. Please follow these steps to copy this folder and compile the program.

```
[jsmith@msgln6 ~]$ cd SCRATCH
[jsmith@msgln6 ~/SCRATCH]$ cp -Rp /nfs/fit3142/2016/Examples/hello .
[jsmith@msgln6 ~/SCRATCH]$ cd hello
[jsmith@msgln6 ~/SCRATCH/hello]$ module load openmpi/1.8.4-gcc-4.8.0
[jsmith@msgln6 ~/SCRATCH/hello]$ mpicc -o mpihello MPI_hello.c
[jsmith@msgln6 ~/SCRATCH/hello]$ mpirun -np 2 mpihello
Hello from rank 0 of 2 on node msgln6.its.monash.edu.au
Hello from rank 1 of 2 on node msgln6.its.monash.edu.au
```

C. Preparing your jobs to run on the cluster

The **msgln6** server is but the "front-desk" node of the entire cluster. Although it has sizeable CPU and RAM resources, real calculations are not to be executed directly on this server.

The full capability of the cluster sits on over a hundred compute servers (or nodes) that form its "back-end". These nodes are provided to run your jobs. However, you will first need to prepare a **job script**, which is essentially a booking request for resources to run your MPI job.

This is a corresponding job script **hello.job** to run the MPI_hello program. It is a shell script with additional markups.

```
#!/bin/sh

#$ -N hellompi
#$ -l h_vmem=1G
#$ -l h_rt=1:00:00
#$ -pe orte_adv 4
#$ -cwd

module load openmpi/1.8.4-gcc-4.8.0
mpirun mpihello
```

The MCC uses the Sun Grid Engine[1] for its batch queueing system. There are other systems in use on other facilities; like PBS Pro[2] and SLURM[3]. Each system uses its own distinct job script syntax. In the case of SGE/OGS, the lines that start with **#$** are treated in a special way.

Setting	Description
`#$ -N jobname`	Sets a user-specified jobname (no spaces in the name please) Useful if you have several jobs and you want to name them appropriately
`#$ -l h_vmem=nG`	Requests for specific amount of memory for this job in n gigibytes
`#$ -l h_rt=hh:mm:ss`	Requests for hh:mm:ss time on the node or nodes to run this job
`#$ -pe orte_adv ncpus`	Requests ncpus (cores) to be allocated to this job ncpus is referred to as the # of **slots** requested by the job
`#$ -cwd`	Use the current working directory for the job

The example job script "**hello.job**" has requested the name of the job as "**hellompi**", memory requested at **1 GB**, for **one hour**, and **4 CPU cores or slots**.

Note that there is no need to fill the **-np** argument for the `mpirun` command. This is because the queueing system will provide this information automatically to `mpirun`. When the `hello.job` runs, `mpirun` will launch four processes, possibly on several nodes, depending on what is available.

[1] Now called the OGS (http://gridscheduler.sourceforge.net/)

[2] http://www.pbspro.org/

[3] http://slurm.schedmd.com/

D. Submitting and monitoring jobs to the cluster

Once a jobscript has been prepared, the only step to do is to submit it to the queue. Assuming that you have copied the "**hello**" folder into SCRATCH, and compiled the MPI_hello.c program to **mpihello**, submitting the job script "**hello.job**" is easy:

```
[jsmith@msgln6 ~]$ cd SCRATCH/hello

### confirm that the executable mpihello and the job file are present
[jsmith@msgln6 ~/SCRATCH/hello]$ ls hello.job mpihello
hello.job mpihello

### submit the job to the queue
[jsmith@msgln6 ~/SCRATCH/hello]$ qsub hello.job
Your job 6012345 ("hellompi") has been submitted

### check the status of the job, it appears to be queued and waiting
[jsmith@msgln6 ~/SCRATCH/hello]$ qstat
job-ID  prior   name       user         state submit/start at     queue        slots ja-task-ID
-----------------------------------------------------------------------------------------------
6012345 0.00000 hellompi   jsmith       qw    10/12/2016 17:47:14                4

### after some time, the job should start running
### and it is running on gn223 (one of the compute nodes)
[jsmith@msgln6 ~/SCRATCH/hello]$ qstat
job-ID  prior   name       user         state submit/start at     queue        slots ja-task-ID
-----------------------------------------------------------------------------------------------
6012345 0.00000 hellompi   jsmith       r     10/12/2016 17:48:02 aqu8@gn223...  4

### qstat returns nothing if all jobs have finished or failed
[jsmith@msgln6 ~/SCRATCH/hello]$ qstat

### check the output file
[jsmith@msgln6 ~/SCRATCH/hello]$ cat hellompi.o6012345
Warning: no access to tty (Bad file descriptor).
Thus no job control in this shell.
Hello from rank 2 of 4 on node gn223.its.monash.edu.au
Hello from rank 1 of 4 on node gn223.its.monash.edu.au
Hello from rank 3 of 4 on node gn223.its.monash.edu.au
Hello from rank 0 of 4 on node gn223.its.monash.edu.au
```

Finally, if you happened to have submitted a job by mistake, you can use the "**qdel**" command to cancel it. This command deletes the job, whether it is in the queued-waiting (qw) or running (r) states. Cancelling a job is a non-recoverable operation. Given jobid, the 7-digit unique identifier of the job, cancel it by:

```
qdel jobid
```

The MCC is a production cluster with typically 50-100 users running jobs. If you want to see other users' jobs, use this command:

```
qstat -u \*
```

What this means is that if you request over 32 cores/slots for your jobs, it may take some time before it starts running.

For any questions and issues, please email the MCC Help Desk at: mcc-help@monash.edu