Monash eResearch Centre

MonARCH Welcomes Students from the FIT3143 Parallel Computing Course

August 2020

Philip Chan
Monash HPC Team

Learning Outcomes

1. Better understanding of HPC
    a. how to access MonARCH system
    b. what do you need installed to access a compute cluster
    c. account information, initial password, and signing into MonARCH
2. Basic Linux and Job Submission
    a. navigation, file management, directory management
    b. $HOME and Project Folders on MonARCH
    c. running the text editor
    d. managing data transfers
3. How to use a compute cluster
    a. basic concepts
    b. SLURM commands
    c. submitting your first job, checking results
4. Further reading

Introduction to High-Performance Computing

- High-performance computing is about employing the best and cost-effective technologies, from processors, storage and networks, to provide aggregated computational capabilities beyond what is typically available to the end-user
- HPC versus HTC
  - high-performance -- running a single program as quickly as possible
  - high-throughput -- running many independent programs as quickly as possible
- HPC/HTC is predominantly achieved by one or more compute clusters
- HPC/HTC are enabling technologies for larger experiments, more complex data analyses, achieving higher accuracy in computational models

Introducing MonARCH

MonARCH (Monash Advanced Research Computing Hybrid)

- university cluster for high-performance/high-throughput computing
- open for access to Monash researchers and collaborators
- hosts courses in Chemistry, Engineering and FIT that require access to advanced tools or hardware, like lots of CPU cores, or GPU devices
- made up of over 83 servers connected via a fast network
- runs on top of the Monash internal cloud
- over 233 active users in the past 90 days
- 40-50 active users who together run 400-500 calculations at once

What you need to access MonARCH

On Windows you will need two applications:

- an **`ssh`** client
  - `putty`
    - https://the.earth.li/~sgtatham/putty/latest/w64/putty.exe
  - MobaXterm
    - https://download.mobatek.net/2032020060430358/MobaXterm_Portable_v20.3.zip
- a secure copy client (for data file transfers)
  - `WinSCP`
    - https://winscp.net/download/WinSCP-5.17.7-Portable.zip
  - `FileZilla` (choose the appropriate OS)
    - https://filezilla-project.org/download.php?show_all=1

On Linux and Mac OS systems, the commands **`ssh`** and **`scp`** are pre-installed
FileZilla is available on these platforms

Where to get info about my MonARCH Account

FIT3143 staff and students are assigned individual user accounts on MonARCH; and your user account is distinct from your Monash username.

Your Monash username is the word before the @ in your email address, e.g.:

```
abcd0001@student.monash.edu
```

Use **+** on MobaXterm => new connection => enter this as the address:

```
118.138.234.199
```

When prompted for a username, enter your Monash username; then enter you **MyMonash** password.   Read the message displayed.

Logging into the MonARCH cluster

Keep your password secure, do not share this with anyone.

By now, you should know your assigned MonARCH (HPC) username:

it is prefixed by “`fit`” followed by three decimal digits, e.g., `fit015`

On MobaXterm, create a new connection using this address:

`monarch.erc.monash.edu`

with your assigned HPC username (e.g., `fit015`) and then enter your given password.

On successful SSH access into MonARCH

You will see a “`Welcome to MonARCH`” MOTD (message-of-the-day)

If there are any updates on cluster availability, we will be posting the information on the MOTD.

Below the MOTD, you should see a command prompt, e.g.,

```
[fit015@monarch-login1 ~]$
```

awaiting your textual commands.

```
************************************************************
* Welcome to MonARCH                                       *
*                                                          *
* For assistance, please contact: mcc-help@monash.edu      *
* Documentation: https://docs.monarch.erc.monash.edu       *
************************************************************

  - Useful SLURM Commands:
     squeue
     SQ                       for extended info on jobs
     sbatch <slurm_script_file>
     scontrol show job <JOBID>
     scancel <JOBID>

  - M3/MonARCH User Scripts:
     show_job
     show_job <JOBID>
     show_cluster

************************************************************
Please limit CPU-intensive processes on this login node
to a maximum of 20 minutes. Please use the sbatch command
to submit longer processes to the queue. There is a short
partition for jobs that take a maximum of 24 hours.
************************************************************

NOTE: Your respective $HOME folders are currently backed-up.
The Lustre project & scratch folders are NOT backed-up.
Please ensure that you keep a copy of IMPORTANT data
outside of MonARCH. We will soon support Lustre backups.

************************************************************
```

First Commands on MonARCH

You are now within your "home folder", we call `$HOME`, also called `~`

First commands:

`pwd`	-	displays the working directory (where you are)
`echo $HOME`	-	try this command and compare its output with above
`date`	-	display the system date and time
`cal 2019`	-	show the calendar for 2019
`fortune`	-	try this
`clear`	-	clears the terminal window
`passwd`	-	change your MonARCH password (do this soon)

Introducing MonARCH

- “CPU-only” nodes (three configurations)
- “GPU” nodes (two configurations)
- 1,476 physical CPU cores
- over 400 TB of Lustre storage
- compute nodes are virtual machines on the Monash Research Cloud

Login node: `monarch.erc.monash.edu`

`monarch-login1.erc.monash.edu`

`monarch-login2.erc.monash.edu`

Data Transfer: `monarch-dtn.erc.monash.edu`

Docs: `https://docs.monarch.erc.monash.edu`

```
show_cluster
```

```
gf01          K80      gpu    18   206   3   Running
gp01          P100     gpu     0   153   0      Busy
gp02          P100     gpu     0   142   0      Busy
gp03          P100     gpu     0   166   0      Busy
hs03          CPU    short     0     1   0      Busy
hs04          CPU    short     0    34   0      Busy
hs05          CPU    short     0    34   0      Busy
hs06          CPU     comp     0    54   0      Busy
hs12          CPU     comp     6    63   0   Running
hs13          CPU     comp     0    59   0      Busy
hs14          CPU     comp     0    65   0      Busy
hs16          CPU     comp     0    62   0      Busy
hs17          CPU     comp     0    49   0      Busy
mi00          CPU     comp     4    43   0   Running
mi01          CPU     comp     3    28   0   Running
mi02          CPU     comp     7    10   0   Running
mi05          CPU     comp     4    27   0   Running
mi06          CPU     comp     1    68   0   Running
mi07          CPU     comp     0    11   0      Busy
mi08          CPU     comp     0    26   0      Busy
mi09          CPU     comp     0    21   0      Busy
mi11          CPU     comp     2     7   0   Running
mi12          CPU     comp     6    68   0   Running
mi14          CPU     comp     0    12   0      Busy
mi15          CPU     comp     4    30   0   Running

                         Summary:
+------------+------------+------------+------------+------------+
|            | CPUS       | Nodes      | K80 GPUs   | P100 GPUs  |
|------------+------------+------------+------------+------------|
| Available  | 89   ( 6%) | 0    ( 0%) | 6    (25%) | 0    ( 0%) |
| In Use     | 1255 (85%) | 53   (91%) | 2    ( 8%) | 6    (50%) |
| Down       | 132  ( 9%) | 5    ( 9%) | 16   (67%) | 6    (50%) |
| Reserved   | 0    ( 0%) | 0    ( 0%) | 0    ( 0%) | 0    ( 0%) |
| ---------- | ---------- | ---------- | ---------- | ---------- |
| Total      | 1476       | 58         | 24         | 12         |
+------------+------------+------------+------------+------------+
```

Files, Directories and Paths

In Unix/Linux, everything is a file; even a directory is just a special file.

Each file on the system is identifiable by its absolute path.

A path is a string that specifies the name & location of a file (or directory); it consists of characters separated by zero or more / ("forward slash" character)

Examples of absolute paths:

are there relative paths?

```
/home/fit014
/usr/local/dos2unix/7.4.0/bin/dos2unix
/mnt/lustre/projects/fs19/tutorial/rawimages
```

Further reading: https://en.wikipedia.org/wiki/Path_(computing)

Files, Directories and Paths (relative paths)

A relative path is one that does not have a leading / ("forward slash" character)

Each file on the system is identifiable by its absolute path. Any absolute path can also be specified as a relative path.

Examples of relative paths:

```
python           ?≡?  ./python
bin/dos2unix      ≡   ./bin/dos2unix
./fs19/tutorial/demo.zip
```

two special relative paths:

.	-	this directory
..	-	the parent directory

Use of relative paths in your program **can break your access to data** if the program is run from another folder on the system.

Your personal and project spaces on MonARCH

`${HOME}`	-	your personal $HOME directory
		good for source codes, configuration files, etc.
		not good for HPC with lots of data
`${HOME}/fs19`	-	shortcut to the project folder for FIT3143
	-	a symbolic link to absolute path `/mnt/lustre/fs19`
`${HOME}/fs19/${USER}`	-	your personal project space
	-	prepare and submit your jobs within this folder
	-	accessible only to you & your lecturer/tutors
	-	Note: `/mnt/lustre` sits on the fast parallel FS

Basic Navigation and Directory Management

`ls`	-	list the file names in current directory
`mkdir` *path*	-	create an empty directory
`rmdir` *path*	-	removes an empty directory
`cd` *path*	-	change into the given path/directory
`cd ..`	-	change into the enclosing directory
`cd`	-	change into your `$HOME` directory
`tree`	-	visualise the directory structure

Or try this sequence:

```
cd; mkdir A; cd A; mkdir a b c; cd; tree
```

Something to try:

```
cd
mkdir 2019
ls
cd 2019
mkdir jan feb mar
ls
cd jan
mkdir wk1
cd ../mar
mkdir wk1
cd ../feb
mkdir wk1 wk2
cd
tree
```

Navigating your way within MonARCH

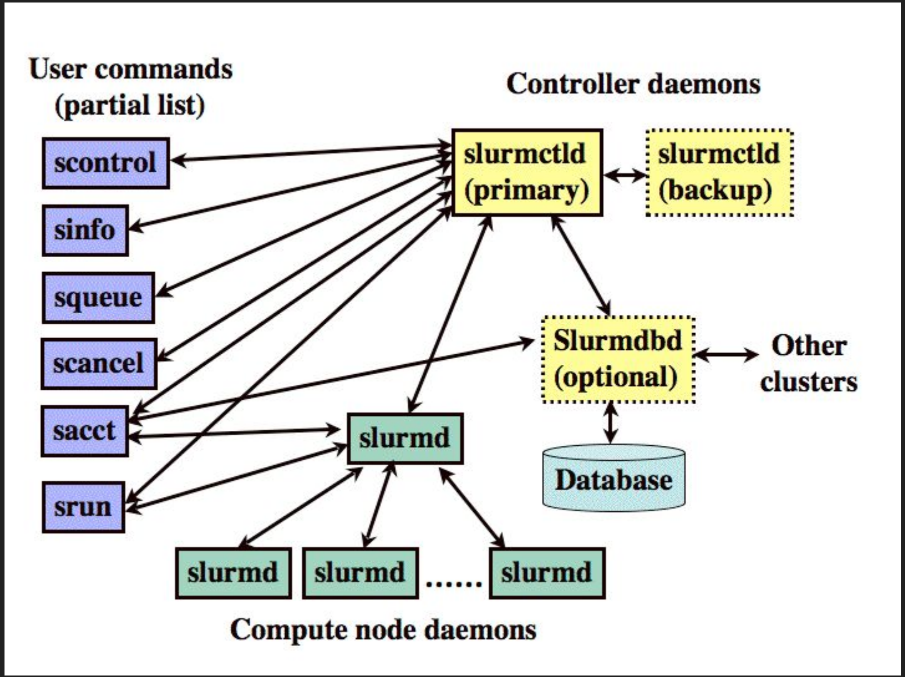
Key commands:

`cat` *path*	-	displays the contents of *path* (try this on a folder?)
		be careful running this on a binary file
`cp` *src dest*	-	makes a copy of *src* into *dest* (both are paths)
`mv` *src dest*	-	moves or renames files (or folders)
`rm` *path*	-	permanently deletes the given file
`ls -l`	-	long directory listing (equivalent shortcut `ll`)
`cd`	-	go back `$HOME`
`cd` *path*	-	change into the given path / folder

SLURM Scheduler

MonARCH uses SLURM as its resource manager and job scheduler:

- fair share scheduling policy
- no fixed CPU allocation, jobs are prioritised based on history of usage



Source: https://slurm.schedmd.com/overview.html

Interacting with the SLURM Job Scheduler

`sbatch jobscript.sh`	- to submit a job for batch execution; returns the unique JobID if successful
`squeue -u ${USER}`	- to display the status of my jobs on the queue
`scontrol show job` *{jobid}*	- to check the status of a job and other details
`sacct -j` *{jobid}*	- to check the accounting information of a given job
`scancel` *{jobid}*	- to delete a job that you own given its unique JobID
`scancel -u ${USER}`	- to delete all your jobs on the cluster

Commands unique to MonARCH:

`sq`	- this is a convenient shortcut to “**`squeue -u ${USER}`**”
`SQ`	- detailed listing of all user jobs (useful to determine job position in the queue)
`show_cluster`	- to see the status of all nodes on the cluster

Submitting your first sample job

1. Change to your project folder:

```
cd ~/fs19/${USER}
pwd
```

2. Make a copy of the demo.zip:

```
cp ../tutorial/demo.zip .
ll
```

3. Unzip the file and cd into `demo`:

```
unzip demo.zip
ll
cd demo
```

4. Submit the “demo.job”:

```
sbatch demo.job
sq
```

5. Check for output:

```
ll
tree
cat slurm*.out
```

6. View any one of the output JPEGs:

```
cd blackwhite
display filename
cd ../sharpened
```

MPI job script template *(when running all 8 mpi processes on one node)*

```
#!/bin/bash
#SBATCH --job-name=np_8_job       ### name your job
#SBATCH --time=00:30:00           ### hh:mm:ss or dd-hh:mm:ss
#SBATCH --mem=32G                 ### set this to 8 ntasks x 4 GB
#SBATCH --ntasks=8                ### launch one python process
#SBATCH --cpus-per-task=1         ### single-threaded processes
#SBATCH --ntasks-per-node=8       ### set this to ntasks if ntasks <= 16
#SBATCH --account=fit3143         ### SLURM setting
#SBATCH --reservation=fit3143     ### reserved nodes

module load openmpi/1.10.7-mlx
srun path-to-mpi-executable
```

1/2 hours max per job

1 cpu max per task

Active as of 27/08/2020

one running max per user

MPI job script template *(same 8 processes but on two nodes)*

```
#!/bin/bash
#SBATCH --job-name=np_32_job      ### name your job
#SBATCH --time=00:30:00           ### hh:mm:ss or dd-hh:mm:ss
#SBATCH --mem=32G                 ### memory setting is max @ 4 GB per core
#SBATCH --ntasks=8                ### launch one python process
#SBATCH --cpus-per-task=1         ### single-threaded processes
#SBATCH --ntasks-per-node=4       ### four per node
#SBATCH --account=fit3143         ### SLURM setting
#SBATCH --reservation=fit3143     ### reserved nodes

module load openmpi/1.10.7-mlx
srun path-to-mpi-executable
```

32 tasks is the current max per job

1/2 hours max per job

8 tasks for this job

1 cpu max per task

4 tasks per node, two nodes

Active as of 27/08/2020

one running max per user

MPI job script template *(beyond 16 cores, you need two nodes)*

32 tasks is the current max per job

```
#!/bin/bash
#SBATCH --job-name=np_32_job      ### name your job
#SBATCH --time=00:30:00           ### hh:mm:ss or dd-hh:mm:ss
#SBATCH --mem=32G                 ### memory setting is max @ 4 GB per core
#SBATCH --ntasks=32               ### launch one python process
#SBATCH --cpus-per-task=1         ### single-threaded processes
#SBATCH --ntasks-per-node=16      ### this is the max as each node has 16 cores only
#SBATCH --account=fit3143         ### SLURM setting
#SBATCH --reservation=fit3143     ### reserved nodes

module load openmpi/1.10.7-mlx
srun path-to-mpi-executable
```

1/2 hours max per job

32 tasks max per job

1 cpu max per task

Active as of 27/08/2020

one running max per user

MPI + OpenMP job script template *(4 MPI processes on two nodes)*

16 cpu cores allocated to this job, 8 cores per node

```
#!/bin/bash
#SBATCH --job-name=mpi_omp_4x4_job    ### name your job
#SBATCH --time=00:30:00               ### hh:mm:ss or dd-hh:mm:ss
#SBATCH --mem=32G                     ### memory setting is max @ 4 GB per core
#SBATCH --ntasks=4                    ### launch one python process
#SBATCH --cpus-per-task=4             ### multi-threaded processes
#SBATCH --ntasks-per-node=2           ### two processes per node
#SBATCH --account=fit3143             ### SLURM setting
#SBATCH --reservation=fit3143         ### reserved nodes

module load openmpi/1.10.7-mlx
srun path-to-mpi-executable
```

1/2 hours max per job

4 tasks for this job

4 cpu per mt process

2 tasks per node, two nodes

Active as of 27/08/2020

one running max per user

MPI Demo Hello World Job

Self study and further reading

Install WinSCP or FileZilla and download/upload files into MonARCH

Running a text editor (nedit or nano). Alternatively, you may edit a file from Windows, upload to MonARCH, but will need to run: `dos2unix` to clean up the file.

Please read through the Getting-Started document:

https://docs.google.com/document/d/1Ewa49D00zYk1GK3Lg__ZxR-w0tUepW69U536w-owDZs

How to pick a secure but easy-to-remember password:

https://xkcd.com/936/

Further help with Unix/Linux:

http://www.ee.surrey.ac.uk/Teaching/Unix/

Thank you for your time.

Your questions welcome.

First point of contact ⇒ your tutor/lecturer

For help with cluster-related issues, email:

`mcc-help@monash.edu`

On the subject, please include this tag: `[FIT3143]`