DeepThought HPC

Flinders University

Jan 16, 2024

USER DOCUMENTATION

The new Flinders University HPC is called DeepThought. This new HPC comprises of AMD EPYC based hardware and next-generation management software, allowing for a dynamic and agile HPC service.

Attention: This documentation is under active development, meaning that it can change over time as we improve it. Please email deepthought@flinders.edu.au if you require assistance. We also welcome user contributions to this documentation - contact the same email above so we can grant you access.

CHAPTER ONE

ATTRIBUTION

If you use the HPC to form a part of your research, you should attribute your usage. Flinders has minted a DOI that points to this documentation, specific for the HPC Service. It will also allow for tracking the research outputs that the HPC has contributed to.

1.1 Text Citation

The below text citation follows the standard Australian Research Data Commons (ARDC) format for attributing data and software. For more information on this type of attribution, visit the ARDC Data Citation page.

`Flinders University (2021). DeepThought (HPC). Retrieved from https://doi.org/10.25957/FLINDERS.HPC.DEEPTHOUGHT`

1.2 Reference Managers

The following files are provided for integration into your reference manager of choice. To save the file, 'Right Click -> Save Link As..'.

- BibTex
- EndNote
- RIS

1.3 How-To Guides

By popular demand, the HPC Team has started putting together some 101's for common pain-points that we spend time training people on. You can find them below, and please, let us know if you would like a guide for something!

Scroll down the page 'How-To' Table of Contents, head over to the User Guides Toc.

1.4 HPC Visual Dashboard

DeepThought has a time-series based visual statistics dashboard that be viewed via a web-browser while on campus.

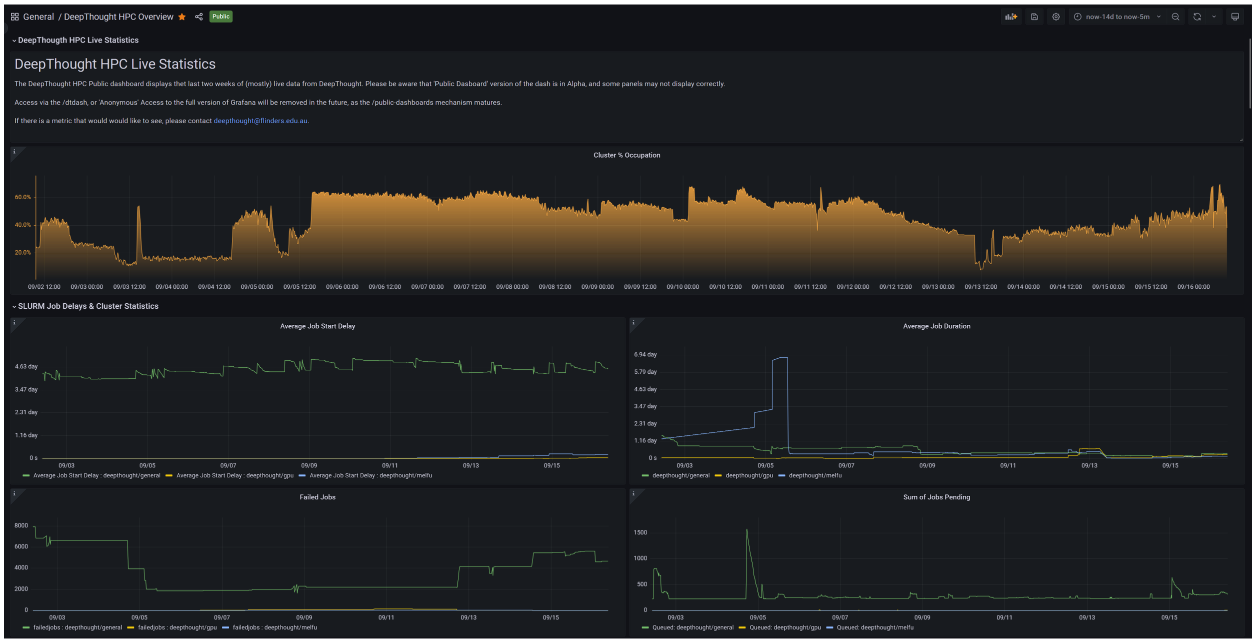
The ‘Default’ and ‘Public’ URL’s are using an *Alpha Release* feature set, and may not display correctly. In that case, please use the ‘Full Version’ link, and any display strangeness will be resolved.

The following URLS all link to the dashboard.

1. Default URL
2. Public Dashboard
3. Full Version

The ‘Full Version’ has the capability to change the time-period viewed, and will be removed when the ‘Public’ and ‘Default’ feature set reach maturity.

A screen shot of a small portion of the Dashboard is below.



DeepThought HPC Live Statistics

The DeepThought HPC Public dashboard displays thet last two weeks of (mostly) live data from DeepThought. Please be aware that 'Public Dasboard' version of the dash is in Alpha, and some panels may not display correctly.

Access via the /dtdash, or 'Anonymous' Access to the full version of Grafana will be removed in the future, as the /public-dashboards mechanism matures.

If there is a metric that would would like to see, please contact deepthought@flinders.edu.au.

CHAPTER TWO

TABLE OF CONTENTS

2.1 Access Requests for DeepThought HPC

Getting access to the HPC is quick and easy process. Follow the steps below to get up and running.

Attention: The HPC now has new URLs for access, specifically to the new web portals.

1. deepthought.flinders.edu.au: SSH / Command Line Access
2. https://deepweb.flinders.edu.au/: User Portal
3. https://deepweb.flinders.edu.au/jupyter: Jupyter Hub

2.1.1 VPN Requirements

When **off campus** you must be connected to the Flinders VPN to access the HPC.

Flinders Staff / HDR Students

1. Create a ServiceOne Ticket asking for Access to the HPC
    a. Currently the request is under the following path: Research Services -> Undertake Research -> Access to DeepThought HPC.
2. Pick your Operating System for additional guidance:
    a. Windows
    b. Windows Sub-System For Linux
    c. Unix Based Systems (Linux & MacOS/X)
3. The HPC Team will send you an on-boarding email once your HPC account has been provisioned

Undergraduates

Currently, the HPC is not open to Undergraduates on a general basis. Exceptions are made on a case-by-case basis - please talk to your project supervisor or Topic Coordinator first, then have them contact the HPC Support Team via Email.

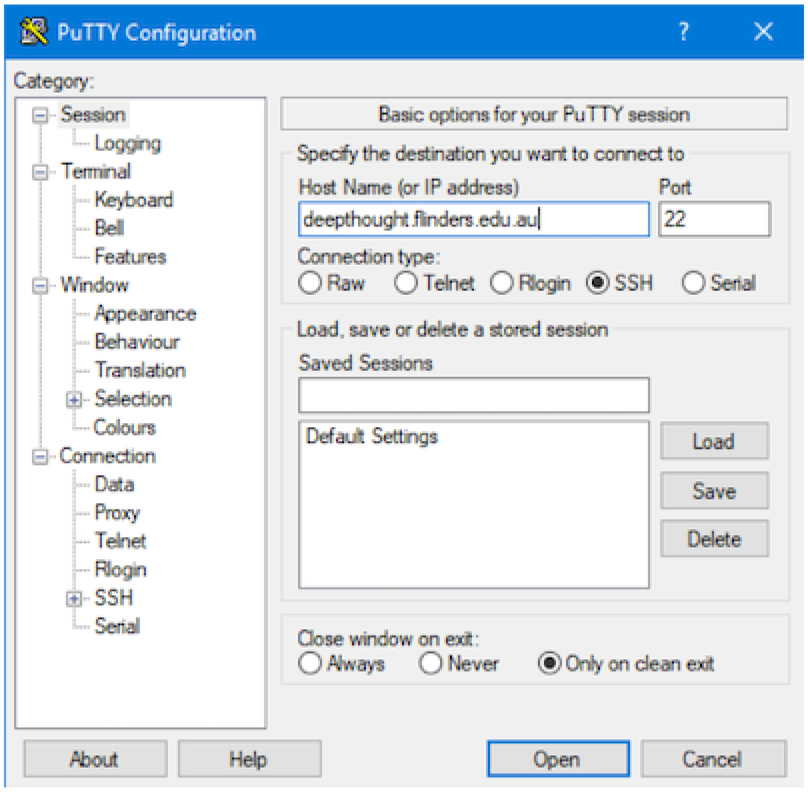
2.2 Windows Connection Guide

To connect to Deep Thought a SSH client such as PuTTy is required. Below is a short list of the possible programs you can use as a client to connect to the HPC. This guide will focus on Putty - but will be equally applicable to the other programs.

2.2.1 Windows SSH Client Options

2.2.2 Windows SSH Connection Guide

Open PuTTy, and you are presented with this screen:



Host Name (or IP address): deepthought.flinders.edu.au

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

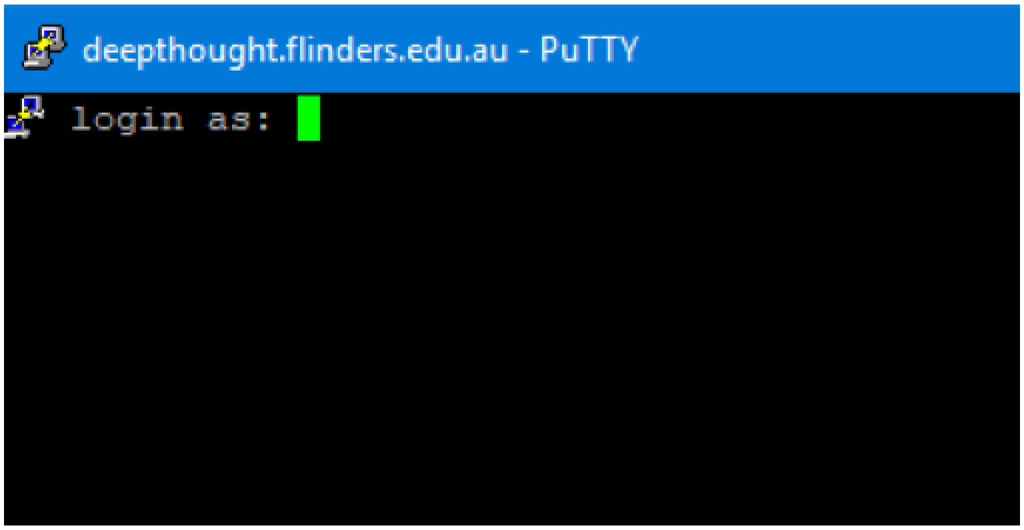
Close window on exit: Always Never [✓ Only on clean exit]

Fig. 1: PuTTY Connection Detail Screen

1. Fill in the hostname to be: deepthought.flinders.edu.au,
2. Change the Connection Type to SSH
3. Set the Port Number to 22
4. Click Open

Logging In on Windows

If all has gone well, you will be presented with a login prompt similar to the following image.



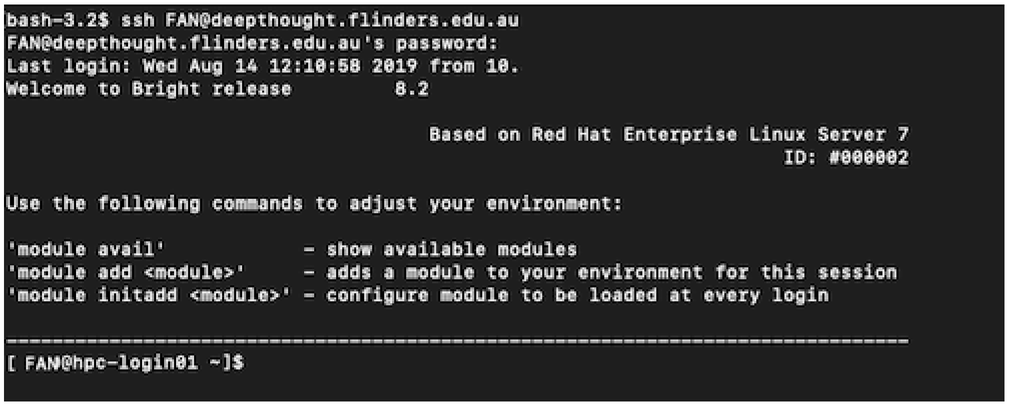
```
login as:
```

Fig. 2: PuTTY Login Prompt

1. Your Username is your FAN
2. Your Password is your FAN Password.

These are the same credentials you use to login to OKTA.

When successful, you will be logged into the system and presented with a screen similar to the image below:



```
bash-3.2$ ssh FAN@deepthought.flinders.edu.au
FAN@deepthought.flinders.edu.au's password:
Last login: Wed Aug 14 12:10:58 2019 from 10.
Welcome to Bright release         8.2

                                     Based on Red Hat Enterprise Linux Server 7
                                                                ID: #000002

Use the following commands to adjust your environment:

'module avail'            - show available modules
'module add <module>'     - adds a module to your environment for this session
'module initadd <module>' - configure module to be loaded at every login

-------------------------------------------------------------------------------
[ FAN@hpc-login01 ~]$
```

Fig. 3: Successful Login

You are now connected to the DeepThought HPC and ready to go.

SSH Keys on Windows

As with the Unix/Linux/MacOS system, you may also setup SSH Keys for password-less logins. Be sure to follow the specific instructions for your client, as they will differ.

2.3 Unix Connection Guide

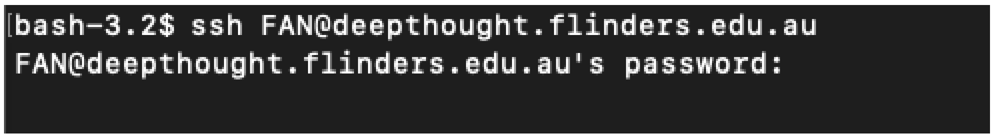
MacOS / MacOSX shares a similar procedure to Unix/BSD Based system. Unix/Linux & MacOS systems have native support for the SSH Protocol, used to connect to the HPC.

2.3.1 Windows Sub-System For Linux

The windows Subsystem for Linux (WSL) allows you to run a Linux Distribution as a sub-system inside the Windows Operating System. When following these instructions, a ‘terminal’ is the same as starting your WSL Distribution.

2.3.2 Logging In on Unix

The simplest manner is to open up a terminal windows and SSH into the HPC.

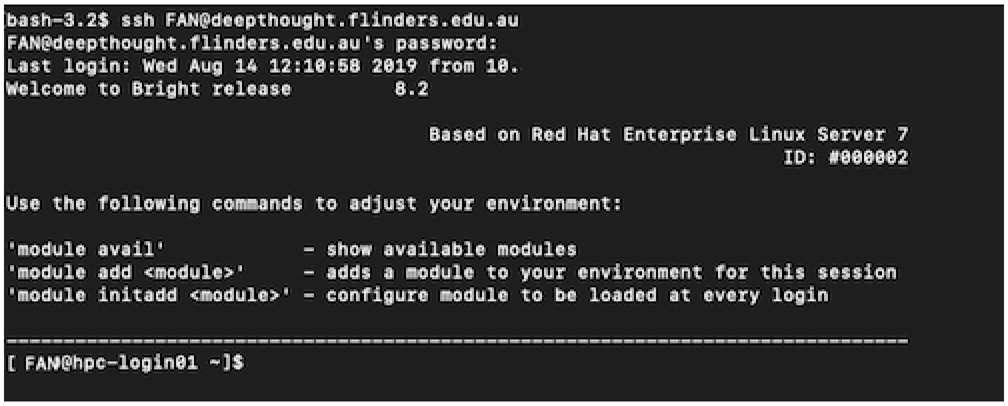


```
[bash-3.2$ ssh FAN@deepthought.flinders.edu.au
FAN@deepthought.flinders.edu.au's password:
```

Fig. 4: SSH Login Prompt

1. Your Username is your FAN
2. Your Password is your FAN Password.

These are the same credentials you use to login to OKTA. When successful, you will be logged into the system and presented with a screen similar to the image below:



```
bash-3.2$ ssh FAN@deepthought.flinders.edu.au
FAN@deepthought.flinders.edu.au's password:
Last login: Wed Aug 14 12:10:58 2019 from 10.
Welcome to Bright release         8.2

                                     Based on Red Hat Enterprise Linux Server 7
                                                                  ID: #000002

Use the following commands to adjust your environment:

'module avail'            - show available modules
'module add <module>'     - adds a module to your environment for this session
'module initadd <module>' - configure module to be loaded at every login

-------------------------------------------------------------------------------
[ FAN@hpc-login01 ~]$
```

Fig. 5: Successful Login

You are now connected to the DeepThought HPC and ready to go.

SSH Keys on Unix

If you wish to setup password-less login via SSH Keys, you may do so.

2.4 Storage Overview & Usage Guidelines

The HPC is a little different that your desktop at home when it comes to storage, not just computing power. It's a shared resource, so we cant store everybody's data for all time - there just isn't enough space. So, before we start putting files onto the HPC, its best you know where to put them in the first place.

On DeepThought, are three main storage tiers. Firstly our bulk storage is the 'Scratch' area - and is slower, spinning Hard-Disk Drives (HDD's). The next storage tier is the 'Cluster' parallel filesystem. For distributed jobs, or jobs that required lots of staged files, this filesystem is the standard high-speed and low-latency HPC filesystem. Be aware that it is smaller that /scratch, and as such, is a *transient* filesystem. Finally, there are hyper-fast NVMe Solid-State Drives (SSD's) located at /local. For the exact specifications and capacities, see the System Specifications.

We also integrate into R-Drive to allow you access to your Research Data Storage allocations. This is an automatic process for any shares you have access to.

There is a critical differences between these three locations, so please read this page carefully.

Attention: The HPC Job & Data Workflow, along with links to the new Data-Workflow Management Portal are under construction and will be linked here when completed.

2.4.1 Storage Accessibility Overview

As general guide, the following table presents the overall storage for the HPC.

Filesystem Location	Accessible From	Capacity
/scratch	All Nodes	~250TB
/cluster	All Nodes	~41TB
/home	All Nodes	~12TB
/local	Individual Compute Nodes	~1TB (400GB on Node019)
/RDrive/<Share Name>	Head Node | Share Dependant	

Attention: Due to an Upstream Kernel Bug, R-Drive integration is currently unavailable. Please use your file transfer program of choice from a wired desktop to move files from the HPC to R-Drive in a single operation. The HPC Team is activly working on rebuilding the integration.

Warning: The HPC is classed as **volatile** storage. Your research data and datasets that you wanted backed up MUST be moved to /RDrive, or off the HPC.

2.4.2 Usage Guidelines

The following sections will go over the individual storage location/mounts along with some general guidelines of what should be stored where.

/Scratch

Scratch is your working space and 'Large Dataset Storage' As you can see from the table above it have far more storage capacity than /home. If you need to store a large data-set, then it should live in here.

What to store in /scratch

Here is a rough guide as to what should live in your /scratch/$FAN directory. In general, anything large, bulky and only needed for a little while should go here.

- Job Working Data-sets
- Large Intermediate Files

/cluster

Cluster is the new, high speed parallel filesystem for DeepThought, deployed using BeeGFS. It is highly recommended that you take advantage of high speeds available to reduce the I/O times associated with /scratch - so **please read this section carefully**.

When performing parallel jobs, to prevent congestion, its best to have a random back-off to your job-scripts. This will take you stage-in times down from up to 30 *minutes* to less than 3 *minutes*. An example of this random backoff is below:

```
sleep $(echo $RANDOM%30 | bc)
```

The directories you can write to in /cluster are controlled by SLURM. When your job starts, SLURM sets multiple environment variables and creates directories for you to use on this filesystem. See the environment variables sections of the SLURM Guide for more information.

Once you job completes, is cancelled, or errors out, SLURM removes then entire directory of your job. That means, *if you do not move your data from the /cluster filesystem, you will lose all of it*. This is by design, and the HPC Team cannot recover any data lost this way.

When this quota is exceeded, files can still be written, but the HPC Team is notified of the user and their associated usage.

What to store in /cluster?

- Your working data sets
- Temporary job files
- Results, before you copy them back to /scratch

/Home

Your 'home' directories. This is a small amount of storage to store your small bits and pieces. This is the analogous to the Windows 'Documents' folder. At a command prompt, your home directory will be shortened to ~/.

What to store in /home

Here is a rough guide as to what should live in your /home/$FAN directory. In general, you want small, little things is here.

- SLURM Scripts
- 'Small' Results from Jobs
- 'Small' Data-Sets (<5GB)

/Local

Local is the per-node, high speed flash storage that is specific to each node.

What to Store in /local

Only *transient files* should live on /local. Anything that your job is currently working on can be on /local. Once your job has finished with these files, they should be copied (or moved) to /scratch. The directory you were working in on /local should then cleaned, removing all files from your job - is you use the automatic SLURM created directories, then this is done for you.

2.5 HPC Research Data Flow

The following diagram illustrates the overall location of the HPC in the Research Data Management flow. Depending on your view-screen size you may wish to Right Click -> Open Image in New Tab to view the diagram correctly.

Fig. 6: DeepThought HPC Research Data Flow Diagram

Note: This diagram will change as the Research Data Project continues. Please check back regularly.

2.6 Transferring Files to DeepThought

Transferring files to the HPC will change depending upon your OS. Thankfully, there are some excellent tools that take this from 'potentially-scary' to 'click a button or two'.

Before we start, ensure that you have read the Storage Overview & Usage Guidelines.

2.6.1 Transferring Files

All file-transfers are done via Secure File Transfer Protocol (SFTP), or Secure Copy Protocol (SCP). Other options, like the tool RSync are also usable. This guide will focus upon the GUI based tools, using SFTP.

Where is the old r_drive?

The old /r_drive/ mount points where a legacy implementation left over from the eRSA Project. All the data from these drives has been migrated to a /RDrive/ share with the same name, and will appear automatically.

2.6.2 Linux/Unix File Transfers

Linux / Unix based systems share native support for the SFTP Protocol. The Secure Copy Protocol (SCP) is also widely accepted, which can sometimes offer an edge in transfer speed. Tools such as RSYNC are also usable.

The Windows Sub-System for Linux

Since Windows 10 and Windows Server 2019, the windows Subsystem for Linux (WSL) allows you to run a Linux Distribution as a sub-system in windows. When following these instructions, a ‘terminal’ is the same as starting your WSL Distribution.

Transferring Files to the HPC

When using a *NIX based system, using the terminal is the fastest way to upload files to the HPC.

The Quick Version

Substitute your filename, FAN and Password, type scp FILENAME FAN@deepthought.flinders.edu.au:/home/FAN then hit enter. Enter your password when prompted. This will put the file in your home directory on DeepThought. It looks (when substituted accordingly) similar to:

```
scp /path/to/local/file fan@deepthought.flinders.edu.au:/path/on/deepthought/hpc/
```

The Longer Version

To download files from DeepThought, you simply need to invert that command to point to either:

- A name of a Computer that DeepThought ‘knows’ about.
- An IP Address that DeepThought can reach.

Transfers By Computer Name

If you know the hostname of the computer, you can substitute this to transfer files back to your machine. The command stays the same, mostly. You still follow the same idea, we just change where we are pointing it. This one assumed you are transferring it to a Linux/Unix based machine.

The command will take this form:

```
scp FILENAME USERNAME@COMPUTER_NAME:/home/username
```

Transfer By IP Address

If you don't know your computer IP, then the commands of:

- ip addr
- ifconfig

Will be your friend to figure out what it is. Just like above, we slightly change the command, and sub-in an IP instead of a host-name.

```
scp FILENAME USERNAME@COMPUTER_IP_ADDRESS:/home/username
```

2.6.3 Windows

Windows doesn't support the SFTP protocol in a native way. Thankfully, there are lots of clients written to do just this for us.

Sub-System for Linux

You can use the WSL for this - head on over to the *Linux* Guide.

Potential Client List

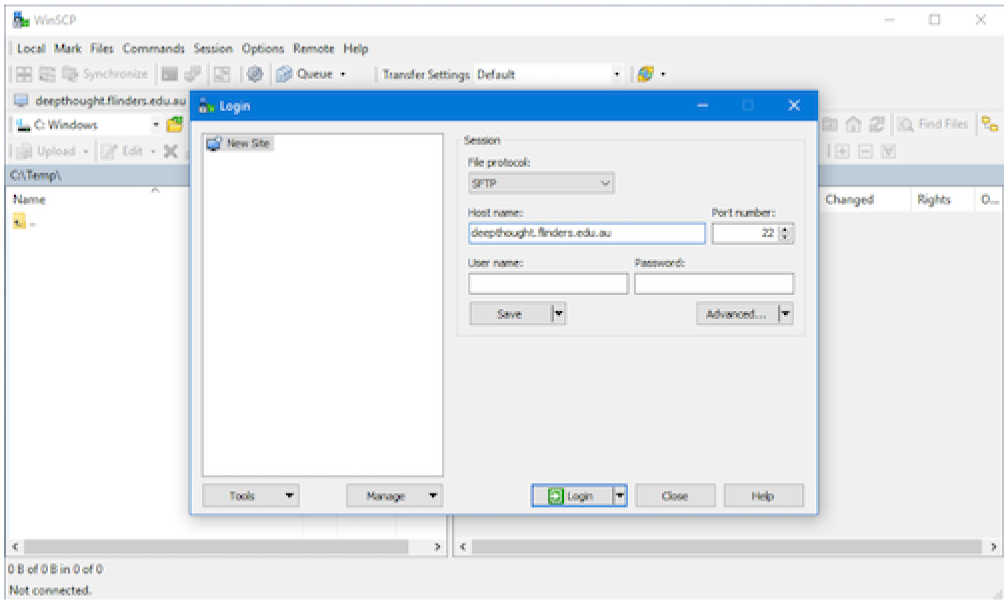
This is not an exhaustive list - feel free to use whatever you wish that supports the SFTP protocol.

- WinSCP
- FileZilla

This guide will focus on WinSCP.

Getting Connected with WinSCP

Open WinSCP, enter deepthought.flinders.edu.au as the host to connect to, and click Login. You should have a screen that looks like this.



Transfer Settings Default

File protocol: SFTP

Host name: deepthought.flinders.edu.au

Port number: 22

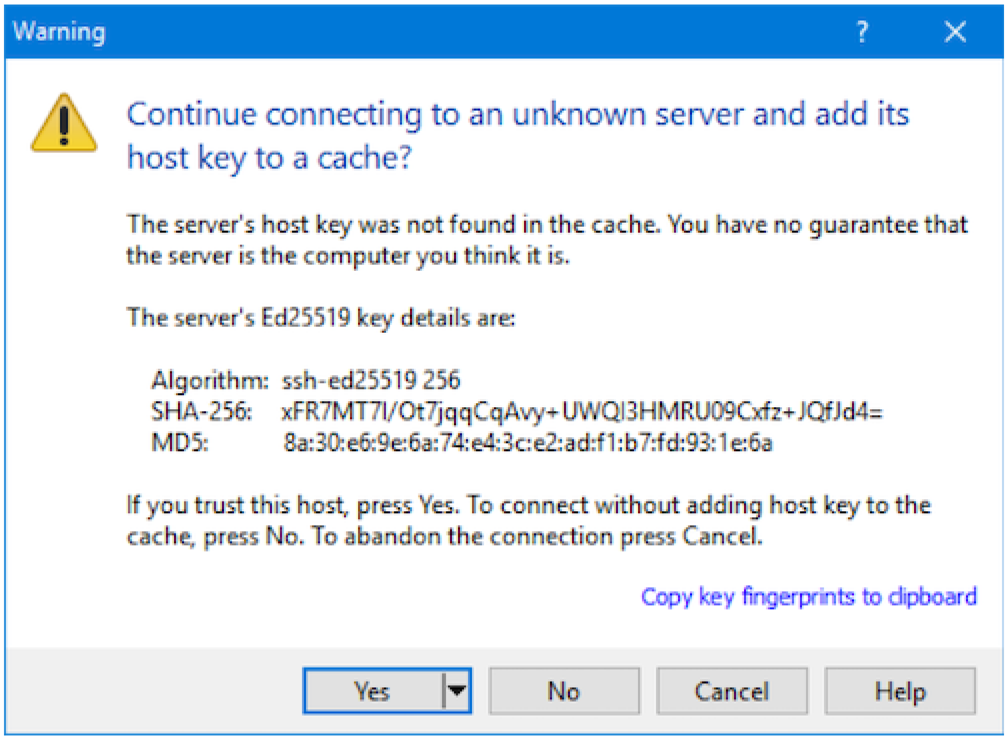
User name:

Password:

0 B of 0 B in 0 of 0

Not connected.

The first time you connect up you will get a warning - this is fine, just click YES to continue on.



Continue connecting to an unknown server and add its host key to a cache?

The server's host key was not found in the cache. You have no guarantee that the server is the computer you think it is.

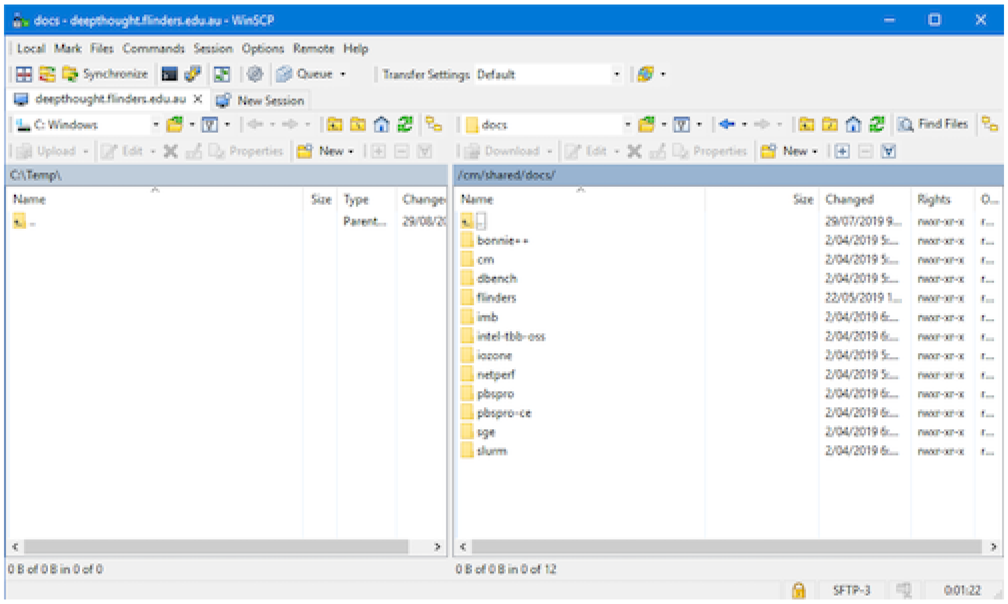
The server's Ed25519 key details are:

```
Algorithm:  ssh-ed25519 256
SHA-256:    xFR7MT7I/Ot7jqqCqAvy+UWQJ3HMRU09Cxfz+JQfJd4=
MD5:        8a:30:e6:9e:6a:74:e4:3c:e2:ad:f1:b7:fd:93:1e:6a
```

If you trust this host, press Yes. To connect without adding host key to the cache, press No. To abandon the connection press Cancel.

Copy key fingerprints to clipboard

A connection to Deep Thought will then be created. If all goes well, you will be treated to this screen:


docs - deepthought.flinders.edu.au - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
deepthought.flinders.edu.au
New Session
C: Windows
docs
Find Files
Upload
Edit
Properties
New
Download

C:\Temp\

Name	Size	Type	Changed
..		Parent...	29/08/20

0 B of 0 B in 0 of 0

/cm/shared/docs/

Name	Size	Changed	Rights	O...
..		29/07/2019 9...	rwxr-xr-x	r...
bonnie++		2/04/2019 5:...	rwxr-xr-x	r...
cm		2/04/2019 5:...	rwxr-xr-x	r...
dbench		2/04/2019 5:...	rwxr-xr-x	r...
flinders		22/05/2019 1...	rwxr-xr-x	r...
imb		2/04/2019 6:...	rwxr-xr-x	r...
intel-tbb-oss		2/04/2019 6:...	rwxr-xr-x	r...
iozone		2/04/2019 5:...	rwxr-xr-x	r...
netperf		2/04/2019 5:...	rwxr-xr-x	r...
pbspro		2/04/2019 6:...	rwxr-xr-x	r...
pbspro-ce		2/04/2019 6:...	rwxr-xr-x	r...
sge		2/04/2019 6:...	rwxr-xr-x	r...
slurm		2/04/2019 6:...	rwxr-xr-x	r...

0 B of 0 B in 0 of 12

SFTP-3 0:01:22

You can now drag and drop files between your computer (Left-hand side) and DeepThought (Right-hand side).

2.7 Linux Command Line Interface

We try not to re-invent the wheel, and there is already a wonderful guide for basic linux commands, written by the excellent people at Harvard. Head on over to here and have a read!

2.7.1 Tips and Tricks

If you want more from your command line, they have a useful tips and tricks guide that should keep you satisfied.

2.7.2 Help! I have no idea

Stuck? No idea what Unix is? Linux? BSD? This all complete gibbersh-and-gobbldegook to you? If so, head on over to a Introduction to UNIX and the Command Line that will help get you up to speed on the interface of choice for High-Performance Computing.

2.8 HPC Job Data Flow

To run jobs in an efficient manner on the HPC, there is usually some additional preparation work required. The below diagram will aid you in ascertaining what steps to take to ensure that you run your jobs in the quickest manner.

Depending on your view-screen size you may wish to Right Click -> Open Image in New Tab to view the diagram correctly.

Fig. 7: DeepThought HPC Job Data Flow Diagram

Note: If you have any questions, please reach out to the support team at deepthought@flinders.edu.au

2.9 SLURM

Slurm (Simple Linux Usage Resource Manager) is used to configure, run and otherwise manage jobs on the HPC. From the Slurm quick start guide:

"Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. …

As a cluster workload manager, Slurm has three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work."

2.9.1 System Specifications

If you want to know the system specifications for DeepThought, head on over to *here*

2.9.2 SLURM on DeepThought

SLURM on DeepThought uses the 'Fairshare' work allocation algorithm. This works by tracking how many resource your job takes and adjusting your position in queue depending upon your usage. The following sections will break down a quick overview of how we calculate things, what some of the cool-off periods are and how it all slots together.

In a sentence, what does this mean for you?

- Greedy users have to wait to run jobs - but only if there are people ahead of them!

What will it mean in the future?

- Greedy users will have to wait to start jobs *and* if they are running a job with a 'non-greedy' person waiting, their job will be forcibly paused to let the other person run their job.

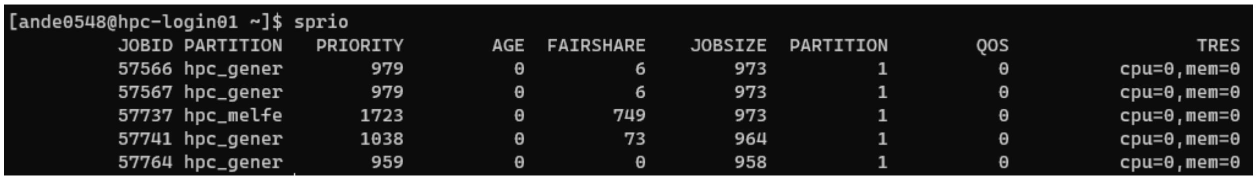
SLURM Priority

The 'Priority of a job is a number you get at the end of the Fairshare calculation. This is attached to your user and is inherited by accounts(colleges). For example, a system might have 1000 'Shares' - think of them as a arbitrary concept of 'some form of computation'. Then the individual 'College' Accounts would be divvied up as needed for all the users that are within that college.

At any point, you can check the 'live' priority scores of user accounts via the command:

```
sprio
```

Which will print something resembling this table:

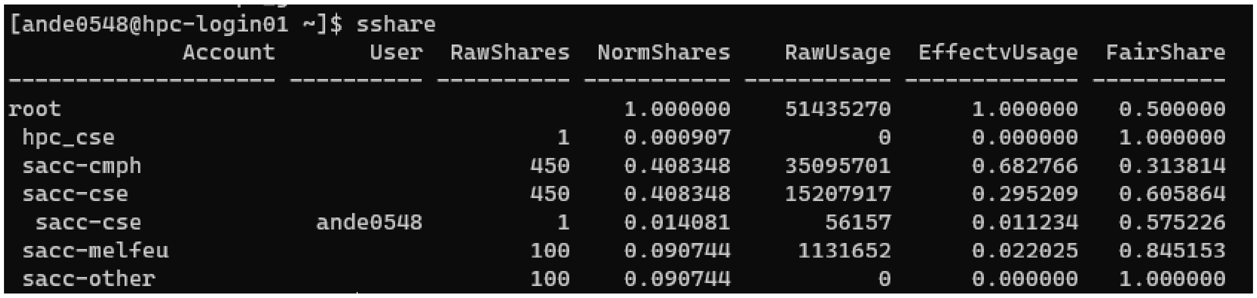


```
[ande0548@hpc-login01 ~]$ sprio
          JOBID PARTITION   PRIORITY        AGE  FAIRSHARE    JOBSIZE  PARTITION        QOS                 TRES
          57566 hpc_gener        979          0          6        973          1          0        cpu=0,mem=0
          57567 hpc_gener        979          0          6        973          1          0        cpu=0,mem=0
          57737 hpc_melfe       1723          0        749        973          1          0        cpu=0,mem=0
          57741 hpc_gener       1038          0         73        964          1          0        cpu=0,mem=0
          57764 hpc_gener        959          0          0        958          1          0        cpu=0,mem=0
```

If you want to interrogate even more data, you can explore the command:

```
sshare
```

Which will allow for greater details of how your score was calculated.



```
[ande0548@hpc-login01 ~]$ sshare
             Account       User  RawShares  NormShares    RawUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
root                                          1.000000    51435270      1.000000   0.500000
 hpc_cse                                 1    0.000907           0      0.000000   1.000000
 sacc-cmph                             450    0.408348    35095701      0.682766   0.313814
 sacc-cse                              450    0.408348    15207917      0.295209   0.605864
  sacc-cse            ande0548           1    0.014081       56157      0.011234   0.575226
 sacc-melfeu                           100    0.090744     1131652      0.022025   0.845153
 sacc-other                            100    0.090744           0      0.000000   1.000000
```

Calculating Priority

SLURM tracks 'Resources'. This can be nearly anything on the HPC - CPU's, Power, GPU's, Memory, Storage, Licenses, anything that the HPC needs to track usage and allocation.

The basic premise is - you have:

- Weight
- Factor
- Unit(s)

Then you multiple all three together to get your end priority. So, lets say you ask for 2 GPU's (The current max you can ask for)

A GPU on DeepThought (when this was written) is set to have these parameters:

- Weight: 5
- Factor: 1000

So your final equation is basically:

```
2 x (5 x 1000) = 10,000
```

There is a then a lot of math to normalize your score against capacity, current cluster utilization, other users comparative usage, how big you job is, favor status, shares available, account shares percentages and a lot more. There are *a lot* of moving parts and its quite complicated! But why are the number so big? So that we have finer-grained priority and we can avoid 'collisions' where two individual users have the same priority number.

Requested Vs. Utilized

This 'Fairshare' score is part of the reason why you must tailor your SLURM scripts or risk your priority hitting rock bottom. If you ask for and entire standard node (64 CPU's + 256GB RAM) you will be 'charged' for that amount - even if you are only using a tiny percentage of those actual resources. This is a huge amount of resources, and will very quickly drop your priority!

To give you an idea of the *initial* score you would get for consuming an entire node, which is then influences by other factors:

CPU: `64 * 1 * 1000 = 64,000` (Measure Per CPU Core)

RAM: `256 * 0.25 * 1000 = 64,000` (Measured Per GB)

Total: `128,000`

So, its stacks up very quickly, and you really want to write your job to ask for what it needs, and not much more! This is not the number you see and should only be taken as an example. If you want to read up on exactly how Fairshare works, then head on over to here.

2.9.3 SLURM: The Basics

These are some of the basic commands. The Slurm Quick-Start guide is also very helpful to acquaint yourself with the most used commands.

Slurm has also produced the Rosetta Stone - a document that shows equivalents between several workload managers.

Job submission

Once the Slurm script is modified and ready to run, go to the location you have saved it and run the command:

```
sbatch <name of script>.sh
```

To test your job use:

```
sbatch --test-only <name of script>.sh
```

This does not actually submit anything. Useful for testing new scripts for errors.

Job information

List all current jobs for a user:

```
squeue -u <username>
```

List all running jobs for a user:

```
squeue -u <username> -t RUNNING
```

List all pending jobs for a user:

```
squeue -u <username> -t PENDING
```

List all current jobs in the shared partition for a user:

```
squeue -u <username> -p shared
```

List detailed information for a job (useful for troubleshooting):

```
scontrol show jobid -dd <jobid>
```

Cancelling a job

To cancel a job based on the jobid, run the command:

```
scancel <jobid of the job to cancel>
```

To cancel a job based on the user, run the command:

```
scancel -u <username of job to cancel>
```

To cancel all the pending jobs for a user:

```
scancel -t PENDING -u <username>
```

To cancel one or more jobs by name:

```
scancel --name myJobName
```

To hold a particular job from being scheduled:

```
scontrol hold <jobid>
```

To release a particular job to be scheduled:

```
scontrol release <jobid>
```

To requeue (cancel and rerun) a particular job:

```
scontrol requeue <jobid>
```

2.9.4 SLURM: Advanced

Job Arrays

Job arrays is a popular strategy to process large numbers of the same workflow repetitively in one go, often reducing analytical time significantly. Job arrays are also often refereed as embarrassingly/pleasingly parallel processes. For more information, see SLURM Job Arrays.

To cancel an indexed job in a job array:

```
scancel <jobid>_<index>
```

To find the original submit time for your job array:

```
sacct -j 32532756 -o submit -X --noheader | uniq
```

The following are good for both jobs and job arrays. Commands can be combined to allow efficiency and flexibility.

Suspend all running jobs for a user (takes into account job arrays):

```
squeue -ho %A -t R | xargs -n 1 scontrol suspend
```

Resume all suspended jobs for a user:

```
squeue -o "%.18A %.18t" -u <username> | awk '{if ($2 =="S"){print $1}}' | xargs -n 1␣
↪scontrol resume
```

After resuming, check if any are still suspended:

```
squeue -ho %A -u $USER -t S | wc -l
```

The following is useful if your group has its own queue and you want to quickly see usage:

```
lsload |grep 'Hostname \|<partition>'
```

Environmental Variables

There are environment variables set by both SLURM and the HPC to manipulate jobs and their execution.

SLURM Specific Environment Variables

The following variables are set per job, and can be access from your SLURM Scripts if needed.

DeepThought Set Environment Variables

The DeepThought HPC will set some additional environment variables to manipulate some of the Operating system functions. These directories are set at job creation time and then are removed when a job completes, crashes or otherwise exists.

This means that if you leave anything in $TMP, $BGFS or $SHM directories it will be *removed when your job finishes*.

To make that abundantly clear. If the Job creates `/cluster/jobs/$SLURM_JOB_USER/$SLURM_JOB_ID` (the $BGFS location) it will also **delete that entire directory when the job completes**. Ensure that your last step in any job creation is to *move any data you want to keep to /scratch or /home*.

$TMPDIR and SLURM Job-Arrays

```
[ande0548@hpcdev-head01 slurm]$ sbatch --array=0-5 tmpdir_increment.sh
Submitted batch job 502

[ande0548@hpcdev-head01 slurm]$ squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
502_0   general tmpdir_.s ande0548  R       0:00      1 hpcdev-node001
502_1   general tmpdir_.s ande0548  R       0:00      1 hpcdev-node001
502_2   general tmpdir_.s ande0548  R       0:00      1 hpcdev-node001
502_3   general tmpdir_.s ande0548  R       0:00      1 hpcdev-node001
502_4   general tmpdir_.s ande0548  R       0:00      1 hpcdev-node002
502_5   general tmpdir_.s ande0548  R       0:00      1 hpcdev-node002

[ande0548@hpcdev-head01 slurm]$ ls
slurm-502_0.out  slurm-502_1.out  slurm-502_2.out  slurm-502_3.out  slurm-502_4.out ␣
↪slurm-502_5.out  tmpdir_increment.sh
```

(continues on next page)

(continued from previous page)

```
[ande0548@hpcdev-head01 slurm]$ cat slurm-502_0.out
TMP: /local/jobs/ande0548/503
TMPDIR: /local/jobs/ande0548/503
JOB ID: 503
TASK ID: 0

[ande0548@hpcdev-head01 slurm]$ cat slurm-502_1.out
TMP: /local/jobs/ande0548/504
TMPDIR: /local/jobs/ande0548/504
JOB ID: 504
TASK ID: 1
```

Notice that the $TMP directories are different for every step in the array? This ensures that each job will never collide with another jobs $TMP, even if they are running on the same node.

To reiterate the warning above - if you leave anything in the $TMP or $SHM Directories, SLURM will delete it at the end of the job, so make sure you move any results out to /scratch or /home.

Filename Patterns

Some commands will take a filename. The following modifiers will allow you to generate files that are substituted with different variables controlled by SLURM.

2.9.5 SLURM: Extras

Here is an assortment of resources that have been passed on to the Support Team as 'Useful to me'. Your mileage may vary on how useful you find them.

Slurm batch scripting

Tasks, jobs & parallel scripting

Besides useful commands and ideas, this FAQ has been the best explanation of 'going parallel' and the different types of parallel jobs, as well as a clear definition for what is considered a task.

An excellent guide to submitting jobs.

2.9.6 SLURM: Script Template

```
#!/bin/bash
# Please note that you need to adapt this script to your job
# Submitting as is will fail cause the job to fail
# The keyword command for SLURM is #SBATCH --option
# Anything starting with a # is a comment and will be ignored
# ##SBATCH is a commented-out #SBATCH command
# SBATCH and sbatch are identical, SLURM is not case-sensitive
##################################################################
# Change FAN to your fan account name
# Change JOBNAME to what you want to call the job
# This is what is shows when attempting to Monitor / interrogate the job,
# So make sure it is something pertinent!
#
```

(continues on next page)

(continued from previous page)

```
#SBATCH --job-name=FAN_JOBNAME
#
##################################################################
# If you want email updates form SLURM for your job.
# Change MYEMAIL to your email address
#SBATCH --mail-user=MYEMAIL@flinders.edu.au
#SBATCH --mail-type=ALL
#
# Valid 'points of notification are':
# BEGIN, END, FAIL, REQUEUE.
# ALL means all of these
##################################################################
# Tell SLURM where to put the Job 'Output Log' text file.
# This will aid you in debugging crashed or stalled jobs.
# You can capture both Standard Error and Standard Out
# %j will append the 'Job ID' from SLURM.
# %x will append the 'Job Name' from SLURM
# %
#SBATCH --output=/home/$FAN/%x-%j.out.txt
#SBATCH --error=/home/$FAN/%x-%j.err.txt
##################################################################
# The default partition is 'general'.
# Valid partitions are general, gpu and melfu
##SBATCH --partition=PARTITIONNAME
#
##################################################################
# Tell SLURM how long your job should run for as a hard limit.
# My setting a shorter time limit, it is more likely that your
# job will be scheduled when attempting to backfill jobs.
#
# The current cluster-wide limit is 14 Days from Start of Execution.
# The timer is only active while your job runs, so if you suspend
# or pause the job, it will stop the timer.
#
# The command format is as follows: #SBATCH --time=DAYS-HOURS
# There are many ways to specify time, see the SchedMD Slurm
# manual pages for more.
#SBATCH --time=14-0
#
##################################################################
# How many tasks is your job going to run?
# Unless you are running something that is Parallel / Modular or
# pipelined, leave this as 1. Think of each task as a 'bucket of
# resources' that stand alone. Without MPI / IPC you can't talk to
# another bucket!
#
#SBATCH --ntasks=1
#
# If each task will need more that a single CPU, then alter this
# value. Remember, this is multiplicative, so if you ask for
# 4 Tasks and 4 CPU's per Task, you will be allocated 16 CPU's
#SBATCH --cpus-per-task=1
```

(continues on next page)

(continued from previous page)

```
###################################################################
# Set the memory requirements for the job in MB. Your job will be
# allocated exclusive access to that amount of RAM. In the case it
# overuses that amount, Slurm will kill the job. The default value is
# around 2GB per CPU you ask for.
#
# Note that the lower the requested memory, the higher the
# chances to get scheduled to 'fill in the gaps' between other
# jobs. Pick ONE of the below options. They are Mutually Exclusive.
# You can ask for X Amount of RAM per CPU (MB by default).
# Slurm understands K/M/G/T For Kilo/Mega/Giga/Tera Bytes.
#
#SBATCH --mem-per-cpu=4000
# Or, you can ask for a 'total amount of RAM'. If you have multiple
# tasks and ask for a 'total amount' like below, then SLURM will
# split the total amount to each task evenly for you.
##SBATCH --mem=12G
###################################################################
# Change the number of GPU's required for you job. The most GPU's that can be
# requested is 2 per node. As there are limited GPU slots, they are heavily
# weighted against for Fairshare Score calculations.
# You can request either a 'gpu:telsa_v100:X' or a 'gpu:x'
#
# You can either request 0, or omit this line entirely if you
# a GPU is not needed.
#
#SBATCH --gres="gpu:0"
###################################################################
# Load any modules that are required. This is exactly the same as
# loading them manually, with a space-separated list, or you can
# write multiple lines.
# You will need to uncomment these.
#module load miniconda/3.0 cuda10.0/toolkit/10.0.130
#module load miniconda/3.0
#module load cuda10.0/toolkit/10.0.130

###################################################################
# This example script assumes that you have already moved your
# dataset to /scratch as part of your HPC Pre-Job preparations.
# Its best to use the $TMP/$TMPDIR setup for you here
# to allow for the HPC to auto-clean anything you
# leave behind by accident.
# If you have a job-array and need a shared directory for
# data on /local, you will need to manually cleanup that
# directory as a part of your job script.

# Example using the SLURM $BGFS Variable (the Parallel Filesystem)
cd $BGFS
cp /scratch/user/<FAN>/dataset ./

###################################################################
# Enter the command-line arguments that you job needs to run.
```

(continues on next page)

(continued from previous page)

```
##################################################################
# Once you job has finished its processing, copy back your results
# and ONLY the results to /scratch, then clean-up the temporary
# working directory
# This command assumes that the destination exists

cp -r /$BGFS/<OUTPUT_FOLDER> /scratch/user/<FAN>/<JOB_RESULT_FOLDER>

# No need to cleanup $BGFS, SLURM handles the cleanup for you.
# Just dont forget to copy out your results, or you will lose them!

##################################################################
```

2.10 The Module System

One of such challenges of running a HPC is effectively managing a complex suite of applications that span the research areas of the entire university. To offset this issue the Flinders DeepThought HPC uses the LMod (Load MODules) system to load/unload applications in the command line. Any modules you use frequently can be loaded on login using your .bash_profile file; modules required for a job should be automated in your SLURM script.

Best way to think of Module is a singular program version and all of its associated dependencies to run correctly.

2.10.1 Additional Software & Modules

Generally speaking, we can install almost all Linux/CentOS bounded software/applications on HPC, but we don't always need to go thorough the effort to install things 'globally' for everybody.

1. Are people other than just me going to use this software?
2. If yes, create a ServiceOne Ticket, and the HPC Support Team will assess the request.

Otherwise, there is nothing stopping you installing the program locally for yourself! If you run into issues installing software then open a ServiceOne ticket, or contact the HPC Support team at their email.

2.10.2 How Do I Install Software?

There are multiple ways to install software on the HPC. Below is an expansion on some of the common ones. The short and sweet version is that, if you compile/install it yourself to your /home or a Virtual Environment of some kind, you are free to do whatever you want! Just be mindful of disk space. If its a common tool that your whole research lab will be using, consider putting in a support request so the HPC Team can make it a global module instead of everybody having their own copy.

You may also utilise the same tooling as the HPC Support Team - EasyBuild. EasyBuild is a management tool allowing for repeatable installation of a specific piece of software to aid in reproducibility of results. You can load this the same way as any other module on the HPC. By default, this tool will install to your home directory, and more information can be read here.

The HPC support team will need to action your request if you need something big and complicated like ANSYS, GNU Octave, a new version of R or other similar large and complicated programs.

Python / Conda

The HPC Team allows you to install your own packages by using the inbuilt package manager tools, like Pythons 'pip', or Conda when using a virtual environment - you cannot install modules globally on the HPC using these tools.

As an example, you can create a Conda Virtual Environment - this is under your complete control and you may install, remove or alter it as you wish. This is also the same for Pythons 'venv', which functions in much the same way.

The Conda Guide is located at: Conda Guide

The Python Guide is located at: Python Guide

EasyBuild

The HPC Support Team use EasyBuild to manage most of the software on the HPC (Not all, things like ANSYS and MATLAB are too complicated for such a tool). It is also open for users to self install software using the same tooling. As with all things HPC, it can get complicated - the documentation is situated here.

Compile Your Own

The HPC uses the FOSS Toolchain, as detailed in the Fair Usage Policy. Should you wish to compile and use your own software, simply load the associated module (eg, foss-2020a) which will load up the associated tools and libraries.

My Toolchain isn't Listed

Should you require a different Toolchain, like LLVM or Go and it is not listed under the `module avail` list, you can either:

1.) Bootstrap the compiler + libraries yourself in your /home directory

2.) Contact the HPC Support Team, either via Email or ServiceOne

2.10.3 Module Format

As Software requirements for research can be very specific, the modules follow suit as well. Newer modules managed with the EasyBuild Management tooling will have the following syntax:

- Program/Version-Toolchain-Version-Interpreter-Version

An example EasyBuild module is: `BioPerl/1.7.2-GCCcore-8.2.0-Perl-5.28.1`.

Manually installed software will always be a shorter format of:

- Program/Version

An example of a manual module is: matlab/r2020b

The following sections will break down the longer string of `BioPerl/1.7.2-GCCcore-8.2.0-Perl-5.28.1`.

Program-Version

This is program that was installed — in this case it's BioPerl, version 1.7.2

Toolchain-Version

The Compiler Toolchain that was used to compile the program — in this case it's GCCcore, version 8.2.0.

Interpreter-Version

Some programs have a dependence on another interpreter, like Perl or Python. In this case it's Perl, version 5.28.1. This means, it will also load the module for Perl, Version 5.28.1.

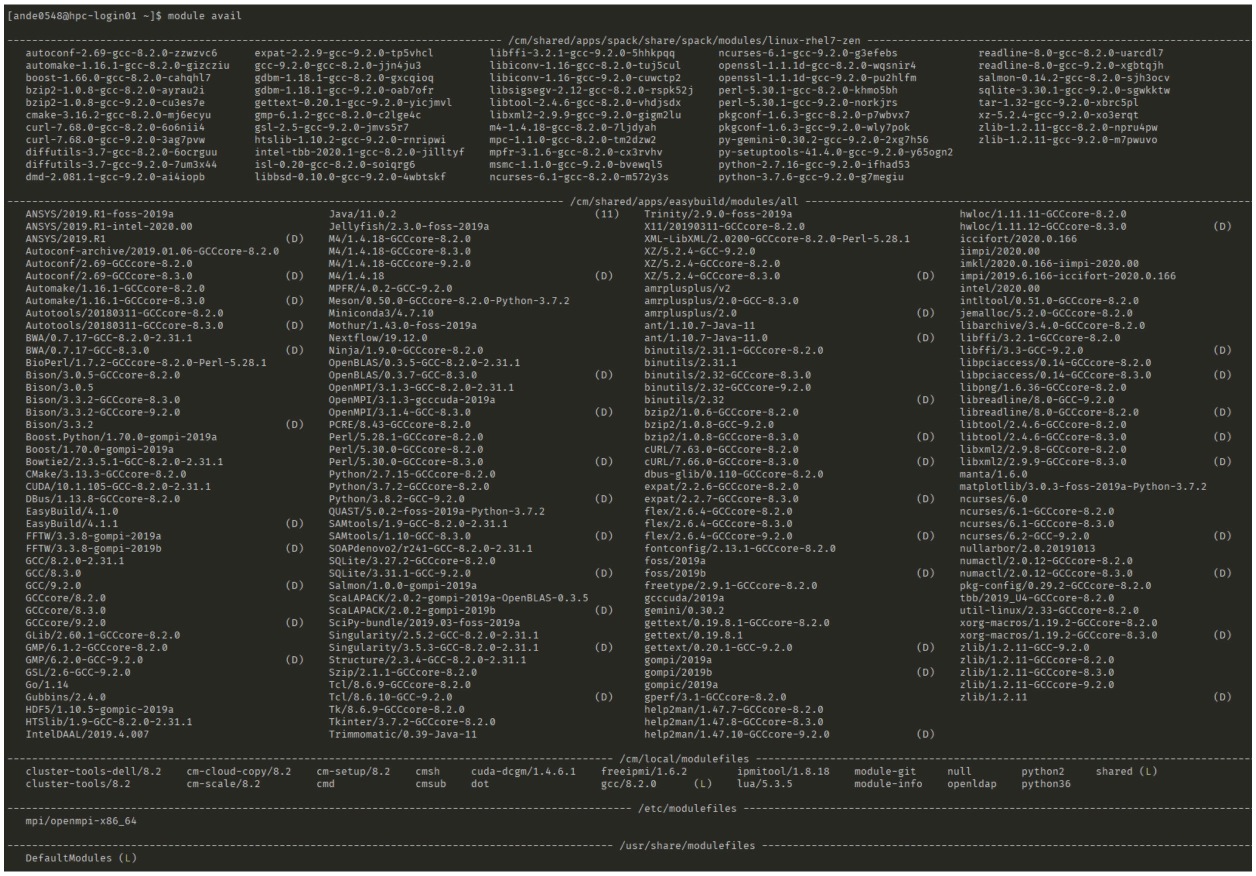
2.10.4 Useful Commands

Below are some common module commands to load, unload and reset your module system.

Available Modules

```
module avail
```

Will get you a list of something similar to this - a list of every single available module on the HPC. You can scroll by your 'Up' and 'Down' arrows and 'q' will exit the scroll if you don't want to scroll all the way to then end. The screenshot below is not an exhaustive list of all modules, just an example!

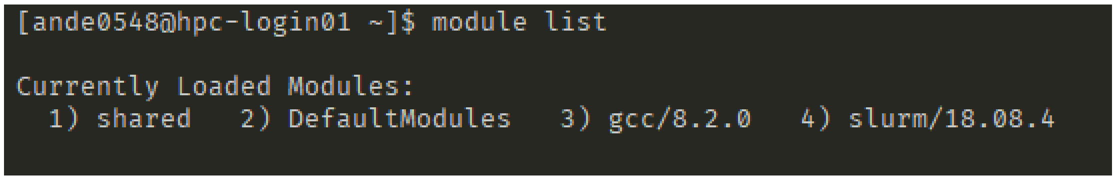


```
[ande0548@hpc-login01 ~]$ module avail

------------------------------------------ /cm/shared/apps/spack/share/spack/modules/linux-rhel7-zen ------------------------------------------
   autoconf-2.69-gcc-8.2.0-zzwzvc6      expat-2.2.9-gcc-9.2.0-tp5vhcl         libffi-3.2.1-gcc-9.2.0-5hhkpqq       ncurses-6.1-gcc-9.2.0-g3efebs             readline-8.0-gcc-8.2.0-uarcdl7
   automake-1.16.1-gcc-8.2.0-gizcziu    gcc-9.2.0-gcc-8.2.0-jjn4ju3           libiconv-1.16-gcc-8.2.0-tuj5cul      openssl-1.1.1d-gcc-8.2.0-wqsnir4          readline-8.0-gcc-9.2.0-xgbtqjh
   boost-1.66.0-gcc-8.2.0-cahqhl7       gdbm-1.18.1-gcc-8.2.0-gxcqioq         libiconv-1.16-gcc-9.2.0-cuwctp2      openssl-1.1.1d-gcc-9.2.0-pu2hlfm          salmon-0.14.2-gcc-8.2.0-sjh3ocv
   bzip2-1.0.8-gcc-8.2.0-ayrau2i        gdbm-1.18.1-gcc-9.2.0-oab7ofr         libsigsegv-2.12-gcc-8.2.0-rspk52j    perl-5.30.1-gcc-8.2.0-khmo5bh             sqlite-3.30.1-gcc-9.2.0-sgwkktw
   bzip2-1.0.8-gcc-9.2.0-cu3es7e        gettext-0.20.1-gcc-9.2.0-yicjmvl      libtool-2.4.6-gcc-8.2.0-vhdjsdx      perl-5.30.1-gcc-9.2.0-norkjrs             tar-1.32-gcc-9.2.0-xbrc5pl
   cmake-3.16.2-gcc-8.2.0-mj6ecyu       gmp-6.1.2-gcc-8.2.0-c2lge4c           libxml2-2.9.9-gcc-9.2.0-gigm2lu      pkgconf-1.6.3-gcc-8.2.0-p7wbvx7           xz-5.2.4-gcc-9.2.0-xo3erqt
   curl-7.68.0-gcc-8.2.0-6o6nii4        gsl-2.5-gcc-9.2.0-jmvs5r7             m4-1.4.18-gcc-8.2.0-7ljdyah          pkgconf-1.6.3-gcc-9.2.0-wly7pok           zlib-1.2.11-gcc-8.2.0-npru4pw
   curl-7.68.0-gcc-9.2.0-3ag7pvw        htslib-1.10.2-gcc-9.2.0-rnripwi       mpc-1.1.0-gcc-8.2.0-tm2dzw2          py-gemini-0.30.2-gcc-9.2.0-2xg7h56        zlib-1.2.11-gcc-9.2.0-m7pwuvo
   diffutils-3.7-gcc-8.2.0-6ocrguu      intel-tbb-2020.1-gcc-8.2.0-jilltyf    mpfr-3.1.6-gcc-8.2.0-cx3rvhv         py-setuptools-41.4.0-gcc-9.2.0-y65ogn2
   diffutils-3.7-gcc-9.2.0-7um3x44      isl-0.20-gcc-8.2.0-soiqrg6            msmc-1.1.0-gcc-9.2.0-bvewql5         python-2.7.16-gcc-9.2.0-ifhad53
   dmd-2.081.1-gcc-9.2.0-ai4iopb        libbsd-0.10.0-gcc-9.2.0-4wbtskf       ncurses-6.1-gcc-8.2.0-m572y3s        python-3.7.6-gcc-9.2.0-g7megiu

---------------------------------------------------- /cm/shared/apps/easybuild/modules/all ----------------------------------------------------
   ANSYS/2019.R1-foss-2019a                        Java/11.0.2                                  (11)  Trinity/2.9.0-foss-2019a                           hwloc/1.11.11-GCCcore-8.2.0
   ANSYS/2019.R1-intel-2020.00                     Jellyfish/2.3.0-foss-2019a                         X11/20190311-GCCcore-8.2.0                         hwloc/1.11.12-GCCcore-8.3.0                (D)
   ANSYS/2019.R1                              (D)  M4/1.4.18-GCCcore-8.2.0                            XML-LibXML/2.0200-GCCcore-8.2.0-Perl-5.28.1         iccifort/2020.0.166
   Autoconf-archive/2019.01.06-GCCcore-8.2.0       M4/1.4.18-GCCcore-8.3.0                            XZ/5.2.4-GCC-9.2.0                                 iimpi/2020.00
   Autoconf/2.69-GCCcore-8.2.0                     M4/1.4.18-GCCcore-9.2.0                            XZ/5.2.4-GCCcore-8.2.0                             imkl/2020.0.166-iimpi-2020.00
   Autoconf/2.69-GCCcore-8.3.0                (D)  M4/1.4.18                                    (D)   XZ/5.2.4-GCCcore-8.3.0                        (D)  impi/2019.6.166-iccifort-2020.0.166
   Automake/1.16.1-GCCcore-8.2.0                   MPFR/4.0.2-GCC-9.2.0                               amrplusplus/v2                                     intel/2020.00
   Automake/1.16.1-GCCcore-8.3.0              (D)  Meson/0.50.0-GCCcore-8.2.0-Python-3.7.2            amrplusplus/2.0-GCC-8.3.0                          intltool/0.51.0-GCCcore-8.2.0
   Autotools/20180311-GCCcore-8.2.0                Miniconda3/4.7.10                                  amrplusplus/2.0                               (D)  jemalloc/5.2.0-GCCcore-8.2.0
   Autotools/20180311-GCCcore-8.3.0           (D)  Mothur/1.43.0-foss-2019a                           ant/1.10.7-Java-11                                 libarchive/3.4.0-GCCcore-8.2.0
   BWA/0.7.17-GCC-8.2.0-2.31.1                     Nextflow/19.12.0                                   ant/1.10.7-Java-11.0                          (D)  libffi/3.2.1-GCCcore-8.2.0
   BWA/0.7.17-GCC-8.3.0                       (D)  Ninja/1.9.0-GCCcore-8.2.0                          binutils/2.31.1-GCCcore-8.2.0                      libffi/3.3-GCC-9.2.0                       (D)
   BioPerl/1.7.2-GCCcore-8.2.0-Perl-5.28.1         OpenBLAS/0.3.5-GCC-8.2.0-2.31.1                    binutils/2.31.1                                    libpciaccess/0.14-GCCcore-8.2.0
   Bison/3.0.5-GCCcore-8.2.0                       OpenBLAS/0.3.7-GCC-8.3.0                     (D)   binutils/2.32-GCCcore-8.3.0                        libpciaccess/0.14-GCCcore-8.3.0            (D)
   Bison/3.0.5                                     OpenMPI/3.1.3-GCC-8.2.0-2.31.1                     binutils/2.32-GCCcore-9.2.0                        libpng/1.6.36-GCCcore-8.2.0
   Bison/3.3.2-GCCcore-8.3.0                       OpenMPI/3.1.3-gcccuda-2019a                        binutils/2.32                                 (D)  libreadline/8.0-GCC-9.2.0
   Bison/3.3.2-GCCcore-9.2.0                       OpenMPI/3.1.4-GCC-8.3.0                      (D)   bzip2/1.0.6-GCCcore-8.2.0                          libreadline/8.0-GCCcore-8.2.0              (D)
   Bison/3.3.2                                (D)  PCRE/8.43-GCCcore-8.2.0                            bzip2/1.0.8-GCC-9.2.0                              libtool/2.4.6-GCCcore-8.2.0
   Boost.Python/1.70.0-gompi-2019a                 Perl/5.28.1-GCCcore-8.2.0                          bzip2/1.0.8-GCCcore-8.3.0                     (D)  libtool/2.4.6-GCCcore-8.3.0                (D)
   Boost/1.70.0-gompi-2019a                        Perl/5.30.0-GCCcore-8.2.0                          cURL/7.63.0-GCCcore-8.2.0                          libxml2/2.9.8-GCCcore-8.2.0
   Bowtie2/2.3.5.1-GCC-8.2.0-2.31.1                Perl/5.30.0-GCCcore-8.3.0                    (D)   cURL/7.66.0-GCCcore-8.3.0                     (D)  libxml2/2.9.9-GCCcore-8.3.0                (D)
   CMake/3.13.3-GCCcore-8.2.0                      Python/2.7.15-GCCcore-8.2.0                        dbus-glib/0.110-GCCcore-8.2.0                      manta/1.6.0
   CUDA/10.1.105-GCC-8.2.0-2.31.1                  Python/3.7.2-GCCcore-8.2.0                         expat/2.2.6-GCCcore-8.2.0                          matplotlib/3.0.3-foss-2019a-Python-3.7.2
   DBus/1.13.8-GCCcore-8.2.0                       Python/3.8.2-GCC-9.2.0                       (D)   expat/2.2.7-GCCcore-8.3.0                     (D)  ncurses/6.0
   EasyBuild/4.1.0                                 QUAST/5.0.2-foss-2019a-Python-3.7.2                flex/2.6.4-GCCcore-8.2.0                           ncurses/6.1-GCCcore-8.2.0
   EasyBuild/4.1.1                            (D)  SAMtools/1.9-GCC-8.2.0-2.31.1                      flex/2.6.4-GCCcore-8.3.0                           ncurses/6.1-GCCcore-8.3.0
   FFTW/3.3.8-gompi-2019a                          SAMtools/1.10-GCC-8.3.0                      (D)   flex/2.6.4-GCCcore-9.2.0                      (D)  ncurses/6.2-GCC-9.2.0                      (D)
   FFTW/3.3.8-gompi-2019b                     (D)  SOAPdenovo2/r241-GCC-8.2.0-2.31.1                  fontconfig/2.13.1-GCCcore-8.2.0                    nullarbor/2.0.20191013
   GCC/8.2.0-2.31.1                                SQLite/3.27.2-GCCcore-8.2.0                        foss/2019a                                         numactl/2.0.12-GCCcore-8.2.0
   GCC/8.3.0                                       SQLite/3.31.1-GCC-9.2.0                      (D)   foss/2019b                                    (D)  numactl/2.0.12-GCCcore-8.3.0               (D)
   GCC/9.2.0                                  (D)  Salmon/1.0.0-gompi-2019a                           freetype/2.9.1-GCCcore-8.2.0                       pkg-config/0.29.2-GCCcore-8.2.0
   GCCcore/8.2.0                                   ScaLAPACK/2.0.2-gompi-2019a-OpenBLAS-0.3.5         gcccuda/2019a                                      tbb/2019_U4-GCCcore-8.2.0
   GCCcore/8.3.0                                   ScaLAPACK/2.0.2-gompi-2019b                  (D)   gemini/0.30.2                                      util-linux/2.33-GCCcore-8.2.0
   GCCcore/9.2.0                              (D)  SciPy-bundle/2019.03-foss-2019a                    gettext/0.19.8.1-GCCcore-8.2.0                     xorg-macros/1.19.2-GCCcore-8.2.0
   GLib/2.60.1-GCCcore-8.2.0                       Singularity/2.5.2-GCC-8.2.0-2.31.1                 gettext/0.19.8.1                                   xorg-macros/1.19.2-GCCcore-8.3.0           (D)
   GMP/6.1.2-GCCcore-8.2.0                         Singularity/3.5.3-GCC-8.2.0-2.31.1           (D)   gettext/0.20.1-GCC-9.2.0                      (D)  zlib/1.2.11-GCC-9.2.0
   GMP/6.2.0-GCC-9.2.0                        (D)  Structure/2.3.4-GCC-8.2.0-2.31.1                   gompi/2019a                                        zlib/1.2.11-GCCcore-8.2.0
   GSL/2.6-GCC-9.2.0                               Szip/2.1.1-GCCcore-8.2.0                           gompi/2019b                                   (D)  zlib/1.2.11-GCCcore-8.3.0
   Go/1.14                                         Tcl/8.6.9-GCCcore-8.2.0                            gompic/2019a                                       zlib/1.2.11-GCCcore-9.2.0
   Gubbins/2.4.0                                   Tcl/8.6.10-GCC-9.2.0                         (D)   gperf/3.1-GCCcore-8.2.0                            zlib/1.2.11                                (D)
   HDF5/1.10.5-gompic-2019a                        Tk/8.6.9-GCCcore-8.2.0                             help2man/1.47.7-GCCcore-8.2.0
   HTSlib/1.9-GCC-8.2.0-2.31.1                     Tkinter/3.7.2-GCCcore-8.2.0                        help2man/1.47.8-GCCcore-8.3.0
   IntelDAAL/2019.4.007                            Trimmomatic/0.39-Java-11                           help2man/1.47.10-GCCcore-9.2.0                (D)

---------------------------------------------------------- /cm/local/modulefiles ----------------------------------------------------------
   cluster-tools-dell/8.2    cm-cloud-copy/8.2    cm-setup/8.2    cmsh     cuda-dcgm/1.4.6.1    freeipmi/1.6.2        ipmitool/1.8.18    module-git     null        python2     shared (L)
   cluster-tools/8.2         cm-scale/8.2         cmd             cmsub    dot                  gcc/8.2.0      (L)    lua/5.3.5          module-info    openldap    python36

----------------------------------------------------------- /etc/modulefiles -----------------------------------------------------------
   mpi/openmpi-x86_64

-------------------------------------------------------- /usr/share/modulefiles --------------------------------------------------------
   DefaultModules (L)
```

Loaded Modules

```
module list
```

Will get you a list of your currently loaded modules.



```
[ande0548@hpc-login01 ~]$ module list

Currently Loaded Modules:
  1) shared   2) DefaultModules   3) gcc/8.2.0   4) slurm/18.08.4
```

Loading Modules

There are three main ways to load a module. For most of the time, they are functionally equivalent. For more information, head on over to LMod Loading Types and read up on how they differ.

- A good tip! Typing out a partial name and double-tapping 'tab' will attempt to complete the name of what you are typing. This means you don't have to worry about typing the very-long name of a module.

Module Load

```
module load BioPerl/1.7.2-GCCcore-8.2.0-Perl-5.28.1.
```

s There is also a nice shortcut, that you can use:

```
ml BioPerl/1.7.2-GCCcore-8.2.0-Perl-5.28.1.
```

'ml' is just short for 'module load'. You can also use it as shorthand for `module` commands, like `ml spider bioperl`.

An Important Note

The software must in all cases be appropriately licensed.

2.10.5 Currently Installed Modules

The latest list is always available by running the `module avail` command on the HPC. Its broken into several segments, reflecting the manual vs. tool-managed modules.

Writing Your Own Modules

You can write your own module files if you want, although this should be a last resort. It is far less error prone to use a Python/Conda environment to bypass the need for modules entirely or the EasyBuild tool to handle this for you.

2.11 User Guides

The HPC Team does its best to write as much as possible on how to use the HPC, we welcome anything that has been written by our users. This is the landing page where we keep track of who has contributed guide directly, or graciously allowed us to tweak existing content.

2.11.1 Compression Overview

Compressing data is basically a required action nowadays. Keeping your dataset around in its raw state would be nigh impossible for most data storage capabilities. So we compress data, for the sake of any SAN administrators and Sys-Admins fighting to keep their storage space from collapsing under their collective weight of research datasets.

Broadly speaking, you get have two main types of things we want to compress in some way:

1. Text
2. Video

Lets not get into Images/TIFFs, as i've never had good results with attempting to compress multi-layer Geo-Tiff's. Just treat images as a 'Text' file, but expect the compression ratio to be somewhere between 'meh' and 'abysmal'.

Also, yes I *know* the correct term for *video compression* is *re-encoding*, but lets not argue semantics.

Compression Trade-Offs 101

If you are compressing something, then something has to give. It usually boils down to a few points that you get to slide between.

1. Computation Time
2. Size of Output / Amount of Reduction
3. Quality of Output

Its (almost) the old adage of ‘Quick, Fast, Cheap’, pick one - the only change is usually, you can pick two *especially for video*. So it plays out, **in very general terms**, somewhat like so:

1. No Computation time, keep the quality? You get a massive file size
2. Lots of Computation time, keep the quality? You get small file size
3. Lots of Computation time, drop some quality? teeny tiny file size

So you get the idea - spend more time to do a thing, and it will, on average, be smaller at the end of it.

2.11.2 Video Compression

Starting with video, as video is *complicated*. This guide is very much **not** a guide on filters, codec differences, hardware vs. software encoding, HDR, CMYK, 4:4:2 vs 4:4:4 chroma or anything of that. So, to be very clear, this will *not be covering*:

1. Chroma Encoding / Gamut / Colour spaces / etc.
2. VBR vs CBR
3. Filters of *any sort*
4. Debate about FOSS/Open vs. Closed vs. Semi-Closed Codec
5. Upscaling / Super-Resolution Style Enhancements
6. X-Special-Presets for Y-Tooling
7. Matroska vs. Mpeg vs. WebM vs. Whatever-Container-You-Champion

What we will cover is going to meet the following criteria:

1. Simple, User Friendly, Cross-Platform and preferably with a GUI
2. A quick list of Key-Terms,
3. A very rough idea of how to get ‘Acceptable’ results

Before we dive too deep. Video encoding will generally take a *very, very long time*. So best you have access to a nvidia GPU of some description, or have a decent workstation that you can throw this at a queue of jobs and just leave it alone for as long as it takes to crunch. You’ll also need disk space a plenty - SSD’s will make sure you don’t get held of by spinning rust.

The NVidia hardware encoder is widely supported, and called NVENC. Funnily enough, the decoder is NVDEC.

Tooling Choice

Handbrake wins hands down for tooling. Its cross platform, supports GPU Acceleration, and can just about convert anything to anything for video. So grab it via your preferred installation method. Its been around for almost as long as the dinosaurs, and does support all the fancy features, like HDR. Which we don't actually care about at this stage, but now you know the tooling has the chops for production level things.

Its also loaded with a bunch of presets that make your life *much* easier, as they have already done all the hard work around dialling in the settings and tweaking the million dials for you.

Video Compressions: The Moving Parts

Video compression is mainly governed by the codec used. Codec is short-hand for enCOder-DECer: CO-DEC. So its made of both the encoder, to create the video file, and the decoder, to play it back. As with all things, there are proprietary and open codecs. Generally, the newer the better, but newer also usually means 'more compute time'. Generally, the best-to-worst is as follows for file-size:

1. AV1
2. VP9
3. H265 (HEVC)
4. H264 (AVC)

Then we have a container of some description. The container is just 'how do we hold and describe the video, audio and subtitle channel' type thing. You put a video stream, some audio streams, and maybe subtitles into a container. That is it, job done.

So, to summarise, you have control over:

1. The Codec
2. The Container

Resolutions and You

While usually, video will list both the pixel resolution and the short-hand friendly name, but not always. Here is a list for you, in the format of <Short Name, Length x Width in Pixels, Acronym>

1. 1080P, 1920x1080, Full-HD / HD
2. 2160P, 2560x1440, QWHD
3. 3160P, 3840x2160, UHD

Compressing Video: The Quick Guide

Handbrake is very easy to use. You can point it at a folder to convert *everything* in that folder, or a single file at a time. The steps are the same, so we will work with a single file for now. And use a preset, to make life easier all around.

1. Open Handbrake
2. Click 'Open Source', then select the Video File you want to compress.
3. Wait for Handbrake to open and parse the file
4. At the top the screen, there is a "Preset" Drop-Down list.
5. Select the preset you would like.

6. In the ‘Format’ Drop-Down, change the container if you like. MKV is generally a little more forgiving for playback than mp4, anecdotally.
7. Hit ‘Start’ Encode
8. Wait

That’s it. The longer video encoding version will be written if and when I get enough interest in pick-everything-manually-process.

Now, onwards! Text compression is must less complicated for the quick version.

2.11.3 Text Compression

Text (or just general files) is usually simpler than video compression. Again, its governed by the algorithm used, and the universal trade-off - but this time, its just the two dimensions of file size and computation time. So, we will start with the algorithms, them move on to tooling.

File Compression Algorithms 101

Generally, the newer algorithms have better compression. However, the highest-is-not-always-the-absolute-best, although we are talking low single-digit percentage difference in final file-size. The usual difference is the exponential difference in computation time. So in VERY rough best-to-worst-on-vague-average, we can talk about the following algorithms. For all that is good, this is just a **rough** estimate. I did say it (file compression) was simpler than video, but compression is still complicated - so this list *is not always truth*. Do not treat it as such!

1. LZMA2 and LZMA (V2 is better at multi-threaded compression)
2. BZIP
3. ZSTD
4. GZIP/ZIP

File Compression Tooling: GUI

Compression tools with a GUI has multiple options, but this guide assumes 7zip. It’s got support for most of the algorithms listed above, is cross-platform and has a nice user interface. Using 7zip is the same as any other compression program - add files to your archive as you want. For best results, you want to avoid .zip , the compression ration is usually hot garbage compared to newer algorithms - only use .zip if you must have an absolutely *ancient* system decompress the file - this includes GNU zip/.gzip/.gz files.

1. AVOID using .zip or .gzip
2. Use the .7z, .xz or .bzip2 archive types - they will use a decent compression algorithm by default
3. Ramp up the ‘Compression Level’ to your wanted level of time-to-compress vs. file-size
4. Hit OK, and wait for you shiny new archive

As a rough ‘real-world’ example, a 1.2GB of python, bash and CSV data-files was compressed, using 7zip, with compression level ‘Ultra’ to about 45MB. Yes, that’s 1200MB down to 45MB. Text will compress down to almost nothing. Only took a few minutes to compress on a decently powerful desktop.

File Compression Tooling: Somewhat-Modern CLI

For any unix-like system, you will almost certainly have access to the program `tar`. There are multiple versions and flavours of this program with slightly different options. For best usage I strongly suggest the RTFM (read the *friendly* manual) approach available via `man tar`. This guide assumes GNU Tar. When using `tar`, older version do not support the direct creation of archives, so the command is slightly different. And, there are two ways of giving it options to control what `tar` does. You have `tar -optionslist` and `tar optionlist`. I.e., with and without the `-` character. *There is a difference**. If in doubt, RTFM.

When using the `-` the `f` for 'Filename' **must come last**. The below commands assume that you can use a somewhat recent vertsion of `tar` that supported direct archive creation.

1. `tar -cJf archivename.xz /path/to/thing/to/compress/`

When not using the `-`, the it doesn't matter the order.

1. `tar fcJ archivename.xz /path/to/compress/`

Now, onwards, to a quick list of options. If its a single letter, you can use it with or with the -. If it has `--option`, you need the –. **Case. Is. IMPORTANT**. Let that be your first and only reminder/warning.

GNU Tar: Quick CLI Options List

A quick list of CLI options. Again, RTFM via `man tar` if want to fine-tune it.

Option Letter	What it does
J	Create a XZ Archive
j	Create a BZIP2 Archive
a	Guess the arhive program to use, by looking at the file suffix, eg `archive.xz` is the same as using `J`.
–lzma	Create a lzma Archive

As a rough real-world example for you: Using XZ, with level 9e, and 12 threads, it took about a day to compress 1.2TB of assorted Geo-Tiff and CSV data file down to about 700MB.

2.11.4 Centre Contributors

The Flinders Accelerator for Microbiome Exploration has some an excellent tutorial series on conda and microbiology python packages. The original content can be found on the FAME homepage.

2.12 Software Suites on DeepThought

DeepThought has several enterprise grade software suites either installed or under active development. These integrations run the gamut from Jupyter Notebooks for quality of life, to desktop integrations to the wider university software ecosystem. This page give a brief overview on what integrations or suites are installed and their associated access requirements.

More software will be added here as development and testing are finished.

2.12.1 List of Enterprise Software Suites

1. ANSYS
2. Delft 3D
3. GROMACS
4. Guassian16
5. Jupyter Hub
6. LAMMPS
7. MATLAB
8. Singularity Containers
9. VASP
10. Open Data Cube

2.13 ANSYS Engineering Suite

2.13.1 ANSYS Status

ANSYS 2021R2 is the current version of the ANSYS Suite installed on the HPC. Both Single-Node (-smp) and Multi-Node (-dis) execution is supported as well as GPU acceleration.

2.13.2 Before You Start

APDL, Fluent and the EM Suite / HFSS all have a common requirement. All script, journal or run files **must** have been generated in *TUI* Mode. This means that all commands must been created, and saved via the programs' Command Line Interface. An example of this is - in Fluent, you **cannot** use the "Journal Recording" function via the GUI. When running the job, the GUI will not be available and Fluent **will fail**.

2.13.3 ANSYS APDL Quickstart Command Line Guide

To run a job with ANSYS APDL on the HPC you will need the following:

- An ANSYS Script file
- Any reference file(s) (eg, a .db file)

Ensure that the paths to anything in the script file reflect where it lives on the HPC, not your local machine. When running with the `-dis` option, you must use a distributed filesystem like /scratch or /cluster, as all nodes will need to the the files, and /local is *not* visible between individual nodes. Below are some example command-line examples to get you started.

Replace all <OPTIONS> to suit your requirements. You can omit the > PATH_TO_OUTPUT_FILE, and SLURM will capture the ANSYS output and write it to your `#SBATCH --output=/path/to/file.out`.

1. Shared-Memory Parallel (Single-Node)

```
ansys212 -smp -np $SLURM_NTASKS -b -s < PATH_TO_SCRIPT_FILE > PATH_TO_OUTPUT_FILE
```

2. Distributed Mode (Multi-Node)

```
ansys212 -dis -mpi openmpi -np $SLURM_NTASKS -b -s < PATH_TO_SCRIPT_FILE >
PATH_TO_OUTPUT_FILE
```

3. Hybrid Distributed Mode (Multi-Node Shared-Memory Parallel)

```
ansys212 -dis -mpi openmpi -np $SLURM_NTASKS -nt $SLURM_CPUS_PER_TASK <SLURM Memory
Allocation> -b -s < PATH_TO_SCRIPT_FILE > PATH_TO_OUTPUT_FILE
```

4. GPU Hybrid Distributed Mode (Multi-Node Shared-Memory Parallel with GPU Acceleration)

```
ansys212 -dis -mpi openmpi -np $SLURM_NTASKS -nt $SLURM_CPUS_PER_TASK -acc nvidia -na
<GPU_COUNT_PER_NODE> -b -s < PATH_TO_SCRIPT_FILE > PATH_TO_OUTPUT_FILE
```

ANSYS APDL CLI Quick List

CLI Option	Description
-acc nvidia	Enable GPU Compute Acceleration
-na value	The number GPU per Compute Node ANSYS should use. Current Max: 2
-dis	Run ANSYS in Distributed (Multi-Node) Mode
-np value	In SMP mode, specify the number of CPU's to use. In Distributed/Hybrid mode, specify the number of Tasks/Processes
-nt value	Specify the number of threads per process in Hybrid mode
-smp	Run ANSYS in Single-Node Mode
-g	Start the Graphical User interface
-b	Enable ANSYS Batch mode. Needs -s.
-i	Full Input file path
-o	Full Output File Path
-s	Read the Ansys Start-up Script
-dir /path	The Working Directory of ANSYS
-db	Initial Allocation for the ANSYS .db file. Can be omitted.
-m value	RAM Allocation for ANSYS, in MB. Can be omitted.
< /path/	Script file Path for batch Mode
> /path/file.out	Path to store ANSYS Output to file

2.13.4 ANSYS Fluent Quickstart Command Line Guide

To run a job using ANSYS Fluent you will need:

- Journal File(s) (TUI Mode *only*)
- Shape Files
- Mesh Files

As with APDL, ensure that the paths to anything in the script file reflect where it lives on the HPC, not your local machine. When running with the -dis option, you must use a distributed filesystem like /scratch (or $BGFS) as all nodes will need to the the files, and /local is not visible between individual nodes. Below are some example command-line examples to get you started.

There are less modes of operation than ANSYS APDL. You must specify the solver type along with any options.

1. Single-Node Parallel, 3-Dimensional, Double-Precision Solver, No GUI or Graphics, Hidden Mesh, 64 Processes, CPU Only

```
fluent 3ddp -g -nm -t 64 -mpi=openmpi -i /path/to/TUI/journal/file
```

2. Single-Node Parallel, 3-Dimensional, Single-Precision Solver, No GUI or Graphics, Hidden Mesh, 64 Processes, 2 GPU's per Compute Node

```
fluent 3d -g -nm -t 64 -mpi=openmpi -gpgpu=2 -i /path/to/TUI/journal/file
```

Important Note: **gpugpu=X** must be divisible evenly with **-t <X>**, or GPU Acceleration will be **disabled**.

2.13.5 SLURM Task Layout and CLI Requirements

When designing your SLURM scripts, you must follow the these guidelines, due to how ANSYS Fluent structures its MPI Calls. If you do no, your Fluent run will not run with the expected core-count!

1. Do **NOT** force a node constraint on SLURM via #SBATCH -N 1 or #SBATCH –nodes=1, unless you know *exactly* what you are doing

```
GPU Acceleration may require this, contact the HPC Team for assitance if you are unsure
```

2. You **MUST** use #SBATCH –ntasks=X and #SBATCH –cpus-per-task=1 to allocate CPUS, *not* #SBATCH –ntasks=1 –cpus-per-task=X

```
#SBATCH --ntasks=64 and #SBATCH --cpus-per-task=1 will get you 64 CPUS, in 64 Tasks
```

3. You **MUST** use #SBATCH –mem-per-cpu= instead of #SBATCH –mem= to prevent unintended memory allocation layouts

```
#SBATCH --mem-per-cpu=3G
```

4. You **MUST** use the -mpi=openmpi flag when running fluent. The Default IntelMPI *will not work*, and hangs indefinitely
5. When recording your Journal File, you **MUST** input the commands via the Command-Line or it *will not work*.
6. You **MUST** alter all paths in the TUI Journal File to use `/` instead of `\`, or it *will not work*
7. You **MUST** alter all paths in the TUI Journal File to be `/absolute/path/to/your/file`, or it *will not work*
8. You **MUST** replace the final command of `/close-fluent` with `/exit y` or your *job will hang until it times out and SLURM kills it*

ANSYS Fluent CLI Solver List

You must match the solver mode to the mode you created the mesh with. If you attempt to solve a Double-Precision Mesh with a Single-Precision solver, Fluent will crash.

CLI Option	Description
3ddp	3-Dimensional, Double Precision Solver
3d	3-Dimensional, Single Precision Solver
2ddp	2-Dimensional, Double Precision Solver
2d	2-Dimensional, Single Precision Solver

ANSYS Fluent CLI Quick List

CLI Option	Description
-aas	Start Fluent in 'Server' Mode
-affinity=<core or sock or off>	Override the automatic process affinity settings
-app=	Load the specified app
-cflush	Clear the System RAM by asking the OS to flush the File-Buffers
-driver=<opengl or x11 or null>	Override the automatic driver detection for Graphics
-g	Run without GUI or Graphics
-gpgpu=<X>	Specify the Number of GPU's per Node (Max 2)
-gr	Run without Graphics
-gu	Run without the GUI
-i /path/to/journal/file	Read and Execute the specified Journal File
-mpi=openmpi	Must be set to OpenMPI. IntelMPI will not work
-nm	Do not display mesh after reading
-post	Run post-processing only
-prepost	Run pre-post only
-r	List all releases/program
-r<V>	Use the specified version
-t <X>	Specify the number of Processors
-tm	Specify the number of Processes for Meshing
-cnf	A MPI Hosts file for Distributed Shared Memory Parallel

Fluent Shared-Memory Parallel / Distributed Mode

Fluent MPI / Distributed Mode needs a 'Hosts' file to tell it what machines to use. Use the below snippet generate a file that can be used as the `-cnf=` parameter to ensure the Distributed shared-memory parallel works correctly.

# Write Number of Procs/Host to Temp File in /home/<USER>/ HOST_FILE=~/$SLURM_JOBID.fluent.hosts.txt T_HOSTS=~./T_HOSTS mpirun hostname | sort | uniq -c > $T_HOSTS

# reformat to a 'HOSTS' file format that FLUENT likes cat ~/T_HOSTS | while read line; do

echo $line | awk '{print $2":"$1}' >> $HOST_FILE

done

You can then invoke Fluent like so, assuming you use the snippet:

```
fluent 3ddp -g -nm -cnf=$HOST_FILE -mpi=openmpi -i /path/to/TUI/journal/file
```

ANSYS CLI Program Quick List

The following table lists the Global ANSYS programs and their associated CLI command.

Program	Name
Mechanical ADPL	ansys212, ansys2021R2
ANSYS Workbench	runwb2
CFX	cfx5
FLUENT	fluent
ICEM CFD	icemcfd
POLYFLOW	polyman
CFD-Post	cfdpost
Icepak	icepak
TurboGrid	cfxtg
AUTODYN	autodyn212

2.13.6 ANSYS EDT (Formerly, HFSS) Quickstart Command Line Guide

To run a job with ANSYS Electronics Desktop (Formerly, HFSS) on the HPC you will need the following:

- Your .aedt File

Ensure that the paths to anything in the script file reflect where it lives on the HPC, not your local machine. Below are some example command-line examples to get you started.

1. The general format of a ANSYS EDT Command is:

```
ansysedt <options> <run command> <project name/script name>
```

2. Single-Node Execution, No GPU

```
ansysedt -ng -batchsolve -Distributed -machinelist list="$SLURM_NODELIST:$SLURM_NTASKS:$SLURM_CPUS_PER_
-monitor /path/to/project.aedt"
```

3. Single-Node Execution, GPU Enabled

```
ansysedt -ng -batchsolve -Distributed --machinelist list="$SLURM_NODELIST:$SLURM_NTASKS:$SLURM_CPUS_PER
-monitor -batchoptions "EnbleGPU=1" /path/to/project.aedt
```

1. Multi-Node

```
Under Testing
```

ANSYS EDT CLI Quick List

All of these options have expanded options for specific use cases. If you need the options, please contact the HPC Team.

CLI Option	Description
-batchsolve	Enable Batch Solving
-ng	Disable GUI, Required for SLURM Jobs
-Local / -Remote / -Distributed	Solver distribution Type, prefer -Distributed
-machinelist list=”host:tasks:cpus”	Define the Machine List for Distributed tasks. Required for -Distributed
-machinelist file=”/path/to/file”	Instead of a CLI option, define a file to use as the Machine list for -Distributed
-batchoptions=”option1=value,options’	Specific batch options. A useful one: EnableGPU=1
-monitor	Print progress to STDOUT

2.13.7 ANSYS RSM: Remote Solve Manager

Last, but certainly not least - the GUI based, fire-and-forget Remote Solve Manger.

There are a few hoops that you must jump through to get it all setup to work with DeepThought. RSM is hard-locked per version of the application.

Currently, only 2021R2 is supported.

You will need to the following installed:

1. ANSYS RSM Configuration 2021R2
2. ANSYS Workbench
3. Fluent / Mechanical / Electrical App

All of the above assumes that already have access to DeepThought. If not, then you will need to ask for access to the HPC.

Setting Up ANSYS RSM 2021R2 for DeepThought

Below is a checklist for getting ANSYS RSM online. 99% of this checklist is initial setup, and after its all set, you *do not* need to do it again!

1. Open ANSY RSM Configuration
2. Add a 'New HPC Resource'
3. Set the 'Name' field to a name you want to refer to this configuration as
4. Set the 'HPC Type' to to 'SLURM'
5. Set the 'Submit Host' to hpc-head01.deept.flinders.edu.au
6. Remember the SLURM 'Job Submission Arguments'. Take a stab at how much RAM per CPU you want, and add –mem-per-cpu=3G etc, to this line. If submitting to the *high-capacity* queue, you will also need to add –qos=hc-concurrent-jobs to this field as well!
7. Ensure that 'User SSH protocol for Inter and intra-node communications is Ticked'
8. 'How Does the Client communicate with SLURM is 'Able to Directly Submit Jobs'
9. Click to the 'File Management' Tab
10. 'Client to HPC File Management' is 'RSM Internal File Transfer Mechanism'
11. 'Staging Directory on the HPC Cluster', is a directory of your choice on /scratch. For example `/scratch/user/<FAN>/ANSYS_STAGE_IN/`
12. 'HPC Side File-Management' is set to 'Scratch directory local to the execution node(s)'
13. 'HPC Scratch directory' is set to `$TEMPDIR`. This is a HPC Admin team managed variable, so it will *just work*.
14. Click to the 'Queues' Tab
15. Use the Autodiscover Options to get RSM to fill out the Queues for you.
16. Use your FAN + Password to allow RSM to log into the HPC on your behalf
17. Close the RSM Configuration for now. Open your Project in your ANSYS App (Mechanical/Fluent etc.)
18. Find the 'Solve Process Settings', and open it.
19. Create a new entry, and point it at the RSM Queue you want to use. RSM Queues are the same as the HPC Queues.

20. Set this new RSM as the 'Default' for your program if you *always* want to use RSM via the HPC to solve things.
21. Open Workbench. For each project that you want to use RSM for, select the 'Solution Properties'
22. Set the 'Solution Process' to 'Submit to Remote Solve Manager'
23. Set the 'Solve Process Setting' to the new setting you just made in Mechanical/Fluent etc.
24. Leave Workbench open for now, SSH into the HPC.
25. Edit your ~/.bashrc file. Right at the end, add the following `module load intel-compilers`. ANSYS has a nice hidden call to `ifort` for user-added functions, so you MUST auto-load the intel-compilers for your user account, or it will fail.
26. Back to Workbench, and now we can hit 'Solve'.
27. Watch the HPC queue, you should see job appearing
28. You can also open the RSM Job Monitor program, and watch your jobs there.
29. DONE! The only alterations would be to the 'Job Submission Arguments' to tweak your –mem-per-cpu= parameter, or if submitting to the high-capacity queue, adding –qos=hc-concurrent-jobs

2.14 CST Studio Suite

2.14.1 CST Studio Suite Status

CST Studio 2022 is the current version of the CST Studio Suite installed on the HPC. Currently, Single-Node Module and GPU acceleration are tested and confirmed to be working via a manual CLI call.

The `cst_` job-submission scripts will work, however they *do not* allocate resources in an effective manner for DeepThought. Use at your own risk, as a manually created SLURM job will give you much better results.

Below are the modes of operation and their status.

Run Mode	Status
GPU Accelerated	Working/Released
Single-Node Parallel	Working/Released
Multi-Node Parallel	Testing/Unreleased

2.14.2 CST Studio Suite Quickstart Command Line Guide

To run a CST Studio job on the HPC you will need your `.cst` file, and the associated SLURM script. Modify any <OPTIONS> to your requirements.

1. Single-Node, GPU Accelerated

   ```
   cst_design_environment -defaultacc -with-gpu=<NUMBER_OF_GPUS_REQUESTED> -m -r
   -num-threads=$SLURM_CPUS_PER_TASK -project-file /path/to/projet.cst
   ```

2. Single-Node, CPU Only

   ```
   cst_design_environment -m -r -num-threads=$SLURM_CPUS_PER_TASK -project-file /path/
   to/project.cst
   ```

3. Multi-Node GPU Accelerated

   ```
   Coming Soon, when MPI is released
   ```

4. Multi-Node, CPU Only

   ```
   Coming Soon, when MPI is released
   ```

CST Studio Suite CLI Quick List

CLI Option	Description
-defaultacc	Enable Compute Acceleration Detection.
-with-gpu=<NUM_GPU>	Per Node, the number of GPU's (Max 2). Requires -defaultacc.
-m -r	Enable 'Batch' Mode
-num-threads=<COUNT>	Number of threads (CPU's) to use
-num-cpudevices=<COUNT>	MPI Only. Use only if you know what NUMA is
-withmpi	Enable MPI
-machinefile=/path/to/file	Path to the CST Machine File. Required for MPI.
-autompi	Automatically detect the MPI Type. Required for MPI.
-shared-dir	A shared space for CST. Defaults to $TMPDIR
-project-file /path/to/file.cst	Path to .CST Project FIle

2.15 Delft3D

2.15.1 Delft3D Status

Delft3D 4, Revision 65936 is installed and available for use on the HPC.

2.15.2 Delft3D Overview

From Delft3D Home:

Delft3D is Open Source Software and facilitates the hydrodynamic (Delft3D-FLOW module), morphodynamic (Delft3D-MOR module), waves (Delft3D-WAVE module), water quality (Delft3D-WAQ module including the DELWAQ kernel) and particle (Delft3D-PART module) modelling

2.15.3 Delft3D Known Issues

1. Delft3D does **not** currently support Multi-Node Execution. The binary swan_mpi.exe will *not* work and immediately crash with errors.
2. The `.mdw` file **CANNOT** start with a capital letter, or `wave` will crash with the error below:

```
terminate called after throwing an instance of 'char const*'

Program received signal SIGABRT: Process abort signal.

Backtrace for this error:
#0  0x15555344cb1f in ???
#1  0x15555344ca9f in ???
#2  0x15555341fe04 in ???
#3  0x155553d8edb4 in _ZN9__gnu_cxx27__verbose_terminate_handlerEv
    at ../../../../libstdc++-v3/libsupc++/vterminate.cc:95
```

(continues on next page)

(continued from previous page)

```
#4  0x155553d8cb85 in _ZN10__cxxabiv111__terminateEPFvvE
    at ../../../../libstdc++-v3/libsupc++/eh_terminate.cc:47
#5  0x155553d8cbd0 in _ZSt9terminatev
    at ../../../../libstdc++-v3/libsupc++/eh_terminate.cc:57
#6  0x155553d8ce13 in __cxa_throw
    at ../../../../libstdc++-v3/libsupc++/eh_throw.cc:93
#7  0x4ee0b8 in throwexception_
    at /local/delf3d/delft3d4_65936/src/utils_lgpl/deltares_common/packages/deltares_
↪common_c/src/throwexception.cpp:35
#8  0x411889 in wave_init_
    at /local/delf3d/delft3d4_65936/src/engines_gpl/wave/packages/kernel/src/wave_init.
↪f90:66
#9  0x404f6d in waves_main
    at /local/delf3d/delft3d4_65936/src/engines_gpl/wave/packages/wave/src/wave_exe.
↪f90:130
#10  0x404a7c in main
    at /local/delf3d/delft3d4_65936/src/engines_gpl/wave/packages/wave/src/wave_exe.
↪f90:35
/cm/local/apps/slurm/var/spool/job1447072/slurm_script: line 104: 811018 Aborted      ␣
↪         (core dumped) wave S3.mdw
```

Delft3D Program Quick List

Below are two main binaries that are used as part of the Delft3D Suite

Program	Description
wave	The WAVE module
swan_omp.exe	The SWAN Module with Single-Node parallelism

Ignore the .exe - ending, it is valid linux binary. Due a transision state between CMake and Make for the Delft3D source-code, the tools and scripts rely on the binary name ending in .exe.

2.16 GROMACS

2.16.1 GROMACS Status

GROMACS version 2021.5 is installed and available for use on the HPC.

2.16.2 GROMACS Overview

From GROMACS:

GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles.

It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a lot of complicated bonded interactions, but since GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers.

GROMACS supports all the usual algorithms you expect from a modern molecular dynamics implementation.

2.16.3 GROMACS Quickstart Command Line Guide

When running on a single node (–ntasks=1 in SLURM)

- `gmx make_ndx <OPTIONS>`
- `gmx grompp <OPTIONS>`
- `gmx mdrun <OPTIONS>`

When running on more than one node (–ntasks>1 in SLURM)

- `mpirun -np $SLURM_NTASKS gmx_mpi make_ndx`
- `mpirun -np $SLURM_NTASKS gmx_mpi grompp`
- `mpirun -np $SLURM_NTASKS gmx_mpi mdrun`

GROMACS Program Quick List

Below is a quick reference list of the different programs that make up the GROMACS suite.

CLI Option	Description
gmx	The Single-Node Only Binary
gmx_mpi	The main MPI Enabled GROMACS binary

2.17 Gaussian

2.17.1 Gaussian Status

Gaussian 16 is installed and available for use on the HPC.

2.17.2 Gaussian Overview

From Gaussian16 Home:

Gaussian 16 is the latest in the Gaussian series of programs. It provides state-of-the-art capabilities for electronic structure modelling. Gaussian 16 is licensed for a wide variety of computer systems. All versions of Gaussian 16 contain every scientific/modelling feature, and none imposes any artificial limitations on calculations other than your computing resources and patience.

Gaussian Program Quick List

The main binary for Gaussian is `g16`.

Running Gaussian SLURM Jobs

Gaussian can be somewhat tricky to ensure that it runs in parallel under SLURM. Below are some starting points for you to fine-tune your Guassian runs. Please remember that not all Gaussian algorithms scale well across *multiple nodes*. There are also several variables to you will want to set, or your job will almost certainly crash!

The below bash function will setup G16 to run correctly, no matter if you asked for GPU or CPU on a single node. There are several ways you can use this script:

1. Copy and past the method body (everything between { and }) `inline` *before* you start `g16`.
2. Copy the entire snippet, then call the setup method as `setup_g16_env` *before* you start `g16`
3. Store it somewhere, and call it as either `. /path/to/env/setup/script.sh` or `source /path/to/env/setup/script`, again, *before* you start `g16`.

The script is as follows:

```
#!/bin/bash
# Author: Scott Anderson
# Version: 1.0
# Reason: Gaussian16 just... doesn't do auto-detection. This script sets up the CDEF,␣
↪GDEF, MDEF, MDisk and SCRDir correctly to allow for gaussian to function in parallel␣
↪without stomping all over SLURM.
# Notes: Abuses the fact that SLURM will quite nicely set affinity for us, so we can do␣
↪a limited amount of parsing
#        We do split out the working CPU set, just for 'verification', but G16 still␣
↪needs the 'GPU Controller' CPU's in its main 'CPU' List
#        Or it will complain about 'CPUID not bound to thread' and then die in a fire.␣
↪Hence, the stange dance of lazy awk sub() calls, but not actually
#        using the list that gets created as the CDEF= option.
#
#        A known limitation: Only up to 2 GPUs. Thats the max we have per node on␣
↪DeepThought, so thats what we can deal with.

setup_g16_env() {
        # Do the memory first.
        ALLOWED_MEM=`cat /sys/fs/cgroup/memory/slurm/uid_$UID/job_$SLURM_JOB_ID/memory.
↪limit_in_bytes | numfmt --to-unit=M`
        BUFFER_MEM=`echo $(( $ALLOWED_MEM - 512 ))`
        export GAUSS_MDEF=`echo $BUFFER_MEM"MB"`
        echo "GAUSS_MDEF set to $GAUSS_MDEF, from a SLURM Maximum of $ALLOWED_MEM"
        # Scratch Disk & Max Disk Read/Write
        export GAUSS_SCRDIR=$BGFS
        export GAUSS_RDEF="Maxdisk=-1"
        export CPUS_ALLOCATED=`cat /sys/fs/cgroup/cpuset/slurm/uid_$UID/job_$SLURM_JOB_
↪ID/cpuset.cpus`
        # Check to see if we got a list or a run of cpus, e.g., 12,13,15,61 vs. 64-72.
        # FYI needs further testing to handle the possibility of 1,2,3,5-22 in the list.
        if [[ "$CPUS_ALLOCATED" == *"-"* ]]; then
                echo "Found Sequential CPU list of $CPUS_ALLOCATED, converting to␣
↪individual core ID's...."
                START_SEQ=`echo $CPUS_ALLOCATED | awk '{split($0,a,"-");print a[1]}'`
                END_SEQ=`echo $CPUS_ALLOCATED | awk '{split($0,a,"-");print a[2]}'`
                echo "Start: $START_SEQ End: $END_SEQ"
                if [[ $START_SEQ && $END_SEQ ]]; then
```

(continues on next page)

(continued from previous page)

```
                        export CPUS_ALLOCATED=`seq -s ',' $START_SEQ $END_SEQ`
                        echo "CPUs set to $CPUS_ALLOCATED..."
                else
                        echo "FOUND SEQUENCE, BUT UNABLE TO FIGURE OUT START AND END OF␣
↪LISTS. CANNOT SETUP GAUSSIAN, JOB EXITING!"
                        exit 99
                fi
        fi
        if [[ -z $CUDA_VISIBLE_DEVICES ]]; then
                echo "No Allocated or visible GPU devices. Not exporting GAUS_GDEF="
                export GAUSS_CDEF=$CPUS_ALLOCATED
                echo "GAUSS_CDEF set to $CPUS_ALLOCATED"
                return
        else
                echo "Found CUDA Visible Devices: $CUDA_VISIBLE_DEVICES"
                # Are we >1 GPU?
                IFS="," read -r -a array <<< $CUDA_VISIBLE_DEVICES;
                if [[ ${#array[@]} -eq 2 ]]; then
                        echo "Found 2 GPUs..."
                        export C_CPU_1=`echo $CPUS_ALLOCATED | awk '{split($0,a,",");␣
↪print a[1]}'`
                        export C_CPU_2=`echo $CPUS_ALLOCATED | awk '{split($0,a,",");␣
↪print a[2]}'`
                        echo "GPU Control CPUs: $C_CPU_1 $C_CPU_2"
                        export WORK_CPUS=`echo $CPUS_ALLOCATED | awk '{split($0,a,",");␣
↪sub(a[1]",", "", $0); sub(a[2]",","",$0); print $0}'`
                        echo "Work CPUS: $WORK_CPUS"
                        export GAUSS_GDEF="$CUDA_VISIBLE_DEVICES=$C_CPU_1,$C_CPU_2"
                        echo "GAUSS_GDEF set to: $GAUSS_GDEF"
                        export GAUSS_CDEF=$CPUS_ALLOCATED
                        echo "GAUSS_CDEF set to: $CPUS_ALLOCATED"
                        return
                elif [[ ${#array[@]} -eq 1 ]]; then
                        echo "Found a Single GPU..."
                        export C_CPU_1=`echo $CPUS_ALLOCATED | awk '{split($0,a,",");␣
↪print a[1]}'`
                        export WORK_CPUS=`echo $CPUS_ALLOCATED | awk '{split($0,a,",");␣
↪sub(a[1]",", "", $0);  print $0}'`
                        echo "Work CPUS: $WORK_CPUS"
                        export GAUSS_GDEF="$CUDA_VISIBLE_DEVICES=$C_CPU_1"
                        echo "GAUSS_GDEF set to: $GAUSS_GDEF"
                        export GAUSS_CDEF=$CPUS_ALLOCTED
                        echo "GAUSS_CDEF set to: $CPUS_ALLOCATED"
                        return

                else
                        echo "CRITICAL ERROR! WE CAN SEE $CUDA_VISIBLE_DEVICES as NOT␣
↪EMPTY, BUT UNABLE TO EXTRA GPU ID's NEEDED"
                        echo "SETTING GAUSSING TO CPU ONLY MODE!"
                        export GAUSS_CDEF=$CPUS_ALLOCATED
                        echo "GAUSS_CDEF set to $CPUS_ALLOCATED"
                        return
```

(continues on next page)

(continued from previous page)

```
                fi
        fi
    }
# Call method as named
setup_g16_env
```

Multi-Node Parallelism: Linda Workers

Linda is the mechanism for Multi-Node parallelism for Gaussian. As **not all algorithms in Gaussian scale well beyond a single node**, each job will need testing to identify if MPI-Enabled jobs gain you any significant speedup.

Do not attempt to mix this with the above script for single-node setup, as it WILL almost certainly fail.

The following SLURM Snippet is a starting point for Multi-Node execution of Gaussian16:

```
## Guassian16 may benefit from >1 CPU/Task.
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=<Memery needed per CPU>
#SBATCH --ntasks=<Total Number of Tasks Wanted"
# 1 Day Runtime, alter as needed
#SBATCH --time=1-0
# Will write to the $SLURM_SUBMIT_DIR, which is wherever you call sbatch <file>
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.err

module load gaussian
#SLURM sets this for us, but just in case.
export OMP_NUM_THREADS=1
# Make sure LINDA Workers are more talkative
export GAUSS_LFLAGS="-v"
# G16 Scratch is in /cluster/jobs/<FAN>/JOBID/
export GAUSS_SCRDIR=$BGFS
# Generate the Linda Parallel hosts file
# If you need >1 CPU/Task, this will need alteration, to append each-nodes CPUSet as␣
↪well as the node-name
for n in `scontrol show hostname | sort -u`; do
echo ${n}
done | paste -s -d, > g.nodes.$SLURM_JOBID

# Datestamp Start of G16 run in the SLURM Output file
date
g16 -w=`cat g.nodes.$SLURM_JOBID` <INPUT FILE>
# Datestamp End of G16 run in the SLURM Output file
date

# remove the par-nodes file
rm g.nodes.$SLURM_JOBID
```

2.18 Jupyter Hub

2.18.1 Jupyter Status

Released and accessible to all HPC Users at the correct URLs.

2.18.2 Jupyter Overview

The Jupyter Enterprise Gateway is a multi-user environment for Jupyter Notebooks. DeepThought has integrated the Jupyter Gateway to allow users to run jobs on the cluster via the native Web Interface.

If you have access to the HPC, you automatically have access to the Jupyter Lab. You can the JupyterLab Instance via the following Jupyter URL or manually via https://deepweb.flinders.edu.au/jupyter. Your credentials are the the same as the HPC, your FAN and password.

If you are a *student* with access to the HPC, the above URLs may work - the URL http://deepteachweb.flinders.edu.au/jupyter is guaranteed to work correctly.

2.18.3 Using Conda Environments in Jupyter Hub

You can use your own custom environment in a Jupyter Kernel. There are some restrictions on python versions that must be adhered to for this integration to work correctly. You will also need to install some specific packages. This process of enabling a Conda environment to work with the HPC, SLURM and Jupyter Hub is detailed below.

Python Version Restrictions

- Python Version <= 3.9.

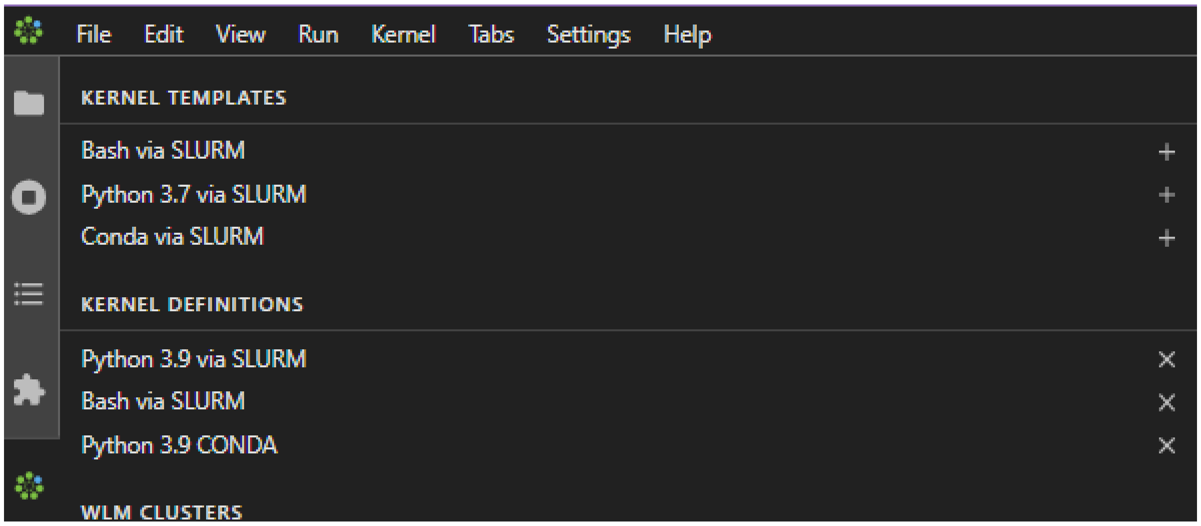
Python 3.10 *will* **not** *work*, due to dependency incompatibility. Specifically, there was a change in the PyCrypto and HTML packages.

Conda Environment Preparation

1. If you have not performed an initial `conda init bash`, issue `module load Miniconda3` followed by `conda init bash`, the log-out and log-in the HPC.
2. Ensure you have a named Conda environment that is loadable by `conda activate <ENVIRONMENT_NAME>`
3. Ensure that the python version used by the environment is Python 3.9 or less.'
4. Log onto the HPC via SSH and activate your Conda environment `conda activate <ENVIRONMENT_NAME>` (You can also use the JupyterHub 'Terminal' to do this)
5. **Execute the following commands:**
   a. `python3 -m pip install cm-jupyter-eg-kernel-wlm bash-kernel`
   b. `conda install ipython ipython_genutils`
6. Log into Jupyter Hub with your FAN and password at the Jupyter URL
7. Using the bar on the Left-hand side, select small green symbol
8. Create a new Kernel based on the 'CONDA via SLURM' template. a. Ensure you select any additional modules you need, like GDAL b. Ensure that you select the correct Conda environment to initialise

9. Use the Kernel in your Jupyter Notebooks

For Reference, the below image shows the Kernel Template Screen.



KERNEL TEMPLATES

Bash via SLURM +
Python 3.7 via SLURM +
Conda via SLURM +

KERNEL DEFINITIONS

Python 3.9 via SLURM ×
Bash via SLURM ×
Python 3.9 CONDA ×

WLM CLUSTERS

Additional Troubleshooting Steps

If you get it wrong, the integration will be mangled for *all* of your Conda Environments, and **will not work correctly**.

If you still get a HTTP 599 or 500 errors on job launch, then perform the following:

1. Activate your conda environment
2. **`python --version` and `python3 --version`**
    a. They MUST be 3.9 or less.
3. **Start a Python Interpreter Session via `python3`**
    a. Attempt to load the Bash-kernel module
    b. `from bash_kernel.kernel import BashKernel`
    c. No Error Messages
4. **Force a re-installation of the Bash Kerlel Layer**
    a. `python3 -m pip install --user --upgrade --force-reinstall bash-kernel`
5. **If you are Submitting to short, and *did not change the max runtime of the Kernel* it wil also not work. This is usually pin-pointanable by an IMMEDIATE error, without any type of delay.**
    a. The Default Runtime is set to `1-0`, which is 24 Hours. Short has a **maxiumum** of 12 Hours
    b. To Fix this, *delete* the Kernel Template, and recreate it wil a *valid runtime* (and resource specification). `12:00:00` is the *maximum amount possibe against short*

Tensorflow / GPU Usage via Jupyter Hub

Conda Environment & Tensorflow Install

It possible to use the GPU nodes and access the GPU's via the JupyterHub Interface. In this example, Tensorflow is used as the GPU-Enabled package of choice. To access the GPUs from your Conda environment and use Tensorflow, perform the following steps:

1. Follow the 'Conda Environment Preparation' steps up to step 4.
2. **Run the following commands to install Tensorflow. DO NOT use `conda install tensorflow`, as it has issues with GPU detection. You *must* use pip.**

    a. `python3 -m pip install tensorflow`

There are a few known issues that can occur here, however the main on is solved below:

1. **An error about HTML has not attribute 'parser' or similar. This is usually resolved by upgrading pip.**

    a. `python3 -m pip install --upgrade pip`

Try and re-install Tensorflow as above. If it still fails, upgrade the rest of the installation tooling for python:

1. `python3 -m pip install --upgrade setuptools`
2. `python3 -m pip install --upgrade distlib`

And once again attempt to Install Tensorflow. If you are still having issues, then reach out to deepthought@flinders.edu.au

Now that the environment is up and running, create a new Kernel Templace from the JupyterHub GUI. For CUDA access you must load the following modules. Alter the version numbers as needed.

- cuda1X.X/toolkit
- cuDNN

If you do not add these modules to the modules list, then you will not be able to use CUDA processing, even if you allocate yourself GPUs on the cluster.

All done! You should now be able to run the following commands from your Jupyter notebook and have Tensorflow return back GPU's in its visible devices list and return a Tensor.

- `python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"`
- `python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"`

2.19 LAMMPS

2.19.1 LAMMPS Status

LAMMPS was installed from the Development Branch on 7th Jan, 2022.

There are two versions of LAMMPS installed on DeepThought, each with their own modules:

1. A CPU only version, with the program called called lmp
2. A GPU only version, with the program called lmp_gpu

You cannot run the GPU enabled version without access to a GPU, as it will cause errors.

2.19.2 LAMMPS Overview

From LAMMPS:

LAMMPS is a classical molecular dynamics code with a focus on materials modelling. It's an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator. LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale.

LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. Many of its models have versions that provide accelerated performance on CPUs, GPUs, and Intel Xeon Phis. The code is designed to be easy to modify or extend with new functionality. LAMMPS is distributed as an open source code under the terms of the GPLv2. The current version can be downloaded here. Links are also included to older versions. All LAMMPS development is done via GitHub, so all versions can also be accessed there. Periodic releases are also posted to SourceForge.

2.19.3 LAMMPS Installed Packages

The following is an extract from the `lmp -h` option, showing the enabled packages and capabilities of the LAMMPS installation.

> ASPHERE ATC AWPMD BOCS BODY BROWNIAN CG-DNA CG-SDK CLASS2 COLLOID COLVARS COMPRESS CORESHELL DIELECTRIC DIFFRACTION DIPOLE DPD-BASIC DPD-MESO DPD-REACT DPD-SMOOTH DRUDE EFF EXTRA-COMPUTE EXTRA-DUMP EXTRA-FIX EXTRA-MOLECULE EXTRA-PAIR FEP GPU GRANULAR H5MD INTERLAYER KIM KSPACE LATBOLTZ LATTE MACHDYN MANIFOLD MANYBODY MC MDI MEAM MESONT MESSAGE MGPT MISC ML-HDNNP ML-IAP ML-PACE ML-QUIP ML-RANN ML-SNAP MOFFF MOLECULE MOLFILE MPIIO MSCG NETCDF OPENMP OPT ORIENT PERI PHONON PLUGIN PLUMED POEMS PTM PYTHON QEQ QMMM QTB REACTION REAXFF REPLICA RIGID SCAFACOS SHOCK SMTBQ SPH SPIN SRD TALLY UEF VORONOI VTK YAFF

2.19.4 LAMMPS Quickstart Command Line Guide

LAMMPS uses UCX and will require a custom mpirun invocation. The module system will warn you of this when you load the module. The following is a known good starting point:

```
mpirun -mca pml ucx --mca btl ^vader,tcp,uct -x UCX_NET_DEVICES=bond0 <program> <options>
```

2.20 Mathworks MATLAB

2.20.1 MATLAB Status

Matlab 2020b is the currently installed version on the HPC.

Flinders has also licenced MATLAB for usage at NCI on Gadi. You will need to join the **matlab_flinders** group via 'My NCI'. Once membership is approved by the HPC Team, then you will be able to run MATLAB at GADI. Please see the NCI Wiki for any help running MATLAB on NCI computing resources.

The HPC Team has updated the licencing and capabilities of MATLAB on DeepThought. You can now use Parallel Server via SLURM and the Parallel Computing Toolbox to run MATLAB in a distributed manner on DeepThought.

Please see the MATLAB help as a starting point to use MATLAB Parallel Server via SLURM or the Parallel Computing Toolkit MATLAB Running Code on Clusters and Clouds is a good starting point.

2.20.2 MATLAB Overview

Mathworks MATLAB is a leading mathematical analysis suite. Coupled with their Parallel Server, the new MATLAB Integration will enable the full power of DeepThought and a greatly enhanced user experience.

2.20.3 MATLAB Quickstart Command Line Guide

To run a job with MATLAB on the HPC you will need the following:

- A MATLAB Script file
- Any Datasets (eg, a .csv)

Ensure that the paths to anything in the script file reflect where it lives on the HPC, not your local machine.

You can then invoke MATLAB (after loading the MATLAB Module) like so:

- matlab -r _PATH_TO_SCRIPT_FILE

2.21 Singularity Containers

2.21.1 Singularity Status

The Singularity Container Engine is available for use on the HPC.

2.21.2 Singularity Overview

Singularity is the most common container engine used for HPC. With the capability to run most container formats, including Docker, DeepThought now allows for containerised workflows via the singularity container engine. More information and a user guide is available here, or at https://singularity.hpcng.org/user-docs/master/.

2.22 VASP

2.22.1 VASP Status

VASP: The Vienna Ab initio Simulation Package version 6.2.0 is operational with OpenACC GPU support on the HPC for the Standard, Gamma Ponit and Non-Collinear versions.

2.22.2 VASP Overview

From VASP:

The Vienna Ab initio Simulation Package (VASP) is a computer program for atomic scale materials modelling, e.g. electronic structure calculations and quantum-mechanical molecular dynamics, from first principles.

VASP computes an approximate solution to the many-body Schrödinger equation, either within density functional theory (DFT), solving the Kohn-Sham equations, or within the Hartree-Fock (HF) approximation, solving the Roothaan equations. Hybrid functionals that mix the Hartree-Fock approach with density functional theory are implemented as well. Furthermore, Green's functions methods (GW quasiparticles, and ACFDT-RPA) and many-body perturbation theory (2nd-order Møller-Plesset) are available in VASP.

2.22.3 VAPS Quickstart Command Line Guide

2.22.4 VASP Quickstart Command Line Guide

VASP must be started via the MPI wrapper script. If your SLURM Script has requested a GPU, VASP will autodetect and use the GPU for all supported operations. Substitute the VASP binary of your requirements.

```
mpirun <MPI OPTIONS> vasp_std <OPTIONS>
```

VASP Program Quick List

Below is a quick reference list of the different programs that make up the VASP suite.

CLI Option	Description
vasp_std	Standard version of VASP
vasp_std_debug	Debug, and slower version of the standard VASP binary. Unless specifically asked, you should use vasp_std
vasp_gam	Gamma Ponit version of VASP
vasp_gam_debu	Debug, and slower version of the Gamma Ponit VASP binary. Unless specifically asked, you should use vasp_gam
vasp_nlc	Non Collinear version of VASP
vasp_ncl_debug	Debug, and slower version of the Non Collinear VASP binary. Unless specifically asked, you should use vasp_ncl

2.23 Open Data Cube

The Open Data Cube provides an integrated gridded data analysis environment for decades of analysis ready Earth observation satellite and related data from multiple sources.

2.23.1 Open Data Cube Status

Released for General Usage. You will need to raise a ServiceOne Request asking for ‘Access to the Open Data Cube’.

2.23.2 Open Data Cube Current Products

The following products have a known index in the Open Data Cube. If the product or date-range you wish to analyse is not currently present in the Data Cube, please raise a ServiceOne request for ‘Add of Modify open Data Cube Datasets’.

ODC Product Name	Description
ga_lst5_ard_3	Geoscience Australia Landsat 5 Thematic Mapper Analysis Ready Data Collection 3
ga_ls7e_ard_3	Geoscience Australia Landsat 7 Enhanced Thematic Mapper Plus Analysis Ready Data Collection 3
ga_ls8c_ard_3	Geoscience Australia Landsat 8 Operational Land Imager and Thermal Infra-Red Scanner Analysis Ready Data Collection 3
ga_s2am_ard_provisio	Geoscience Australia Sentinel 2a MSI Analysis Ready Data Collection 3 (provisional)
ga_s2bm_ard_provisio	Geoscience Australia Sentinel 2b MSI Analysis Ready Data Collection 3 (provisional)
ls5_nbart_geomedial_	Surface Reflectance Geometric Median 25 metre, 100km tile, Australian Albers Equal Area projection (EPSG:3577)
ls7_nbart_geomedial_	Surface Reflectance Geometric Median 25 metre, 100km tile, Australian Albers Equal Area projection (EPSG:3577)
ls8_nbart_geomedian_	Surface Reflectance Geometric Median 25 metre, 100km tile, Australian Albers Equal Area projection (EPSG:3577)
s2a_ard_granule	Sentinel-2A MSI Definitive ARD - NBART and Pixel Quality
s2b_ard_granule	Sentinel-2B MSI Definitive ARD - NBART and Pixel Quality

2.23.3 Open Data Cube Installation Guide

The Open Data Cube is best installed into a Conda Environment, allowing you to update and manipulate it as needed.

1. Follow the Jupyter Conda Setup, if you wish to use the ODC via the JupyterHub instance
2. `conda install datacube`. If this step hangs for more that 5 minutes, then follow the 'Advanced Installation Guide' further down this page to get split the install into smaller parts
3. Raise a ServiceOne request for 'Access to the Open Data Cube'
4. Place the provided configuration file in the following location on the HPC: `/home/<FAN>/.datacube.conf`
5. Activate your Conda environment, and run `datacube system check` to verify that your ODC connects correctly

Open Data Cube Advanced Installation Guide

As the Open Data Cube is quite complicated, sometimes Conda can have trouble resolving the dependencies. In this case, we can manually install many of them, before we install the Data Cube itself.

1. 1. If you have not performed an initial `conda init bash`, issue `module load Miniconda3` followed by `conda init bash`, then log-out and log back in to the HPC.
2. `conda create --name=odc python=3.9`
3. `conda activate odc`
4. `conda config --add channels conda-forge`
5. `conda install affine pyproj shapely cachetools click`
6. `conda install cloudpickle dask[array] distributed jsonschema netcdf4`
7. `conda install numpy psycopg2 lark-parser pandas python-dateutil`
8. `conda install pyyaml rasterio sqlalchemy toolz xarray`
9. `conda install datacube`

10. Raise a ServiceOne request for 'Access to the Open Data Cube'
11. Place the provided configuration file in the following location on the HPC /home/<FAN>/.datacube.conf
12. Activate your Conda environment, and run `datacube system check` to verify that your ODC connects correctly

2.24 Deepthought: System Specifications

Deepthought is the brand new HPC for Flinders University SA, and available for Flinders Colleges to utilise. The following details the system specifications and node types that are present within the HPC.

2.24.1 Partition Layout

The SLURM Scheduler as the notion of 'Job Queue' or 'Partitions'. These manage resource allocations and job scheduling independent from on-another. At this time, there is a single job-queue for general usage. As resource usage patterns are identified, this may change.

2.24.2 Storage Layout

Scratch: ~240TB of scratch disk, mounted on all nodes

Cluster: 41TB of High-Speed Paralllel Fileystem Storage, mounted on all nodes

Per node /local: ~400GB to 1TB, depending on node layout

2.24.3 Node Breakdown

- 20 Compute Nodes, with ~1800 Cores and ~10TB of RAM total
- 5 V100 Nvidia TESLA GPU's with 32GB VRAM per GPU

General Nodes

There are 17 General Purpose nodes, each with:

- CPU:
  - 1 x AMD EPYC 7551 @2.55Ghz with 32 Cores / 64 Threads
- RAM:
  - 256GB DDR4 @ 2666Mhz
- Local Storage
  - 1TB of NVMe SSD's

GPU Nodes

There are 3 dedicated GPU nodes. They comprise of two ‘Standard’ and One ‘Light’ Node:

Standard GPU Nodes

- CPU:
  - 1 x AMD EPYC 7551 @2.55Gz with 32 Cores / 64 Threads
- RAM:
  - 256GB DDR4 @ 2666Mhz
- GPU:
  - 2 x TESLA V100 w/ 32GB VRAM
- Local Storage
  - 1TB of NVMe

Light GPU Node

- CPU:
  - 1 x AMD EPYC 7551 @2.55Gz with 32 Cores / 64 Threads
- RAM:
  - 128GB DDR4 @ 2666Mhz
- GPU:
  - 1 x TESLA V100 w/ 32GB VRAM
- Local Storage
  - 1TB of NVMe

High Capacity Node

There is are 3 High-Capacity nodes with:

- CPU:
  - 2 x AMD EPYC 7742 @2.25Ghz with 64 Cores / 128 Threads
- RAM:
  - 2TB (1.8TB) DDR4 @ 3200Mhz
- Local Storage
  - 1TB of NVMe

Private Nodes

The Flinders University Molecular Biology lab maintains two nodes for their exclusive usage. Access is strictly limited to the Molecular Biology Lab. For completeness, the two nodes are listed here as well.

Melfu General Node

- CPU:
  - 1 x AMD EPYC 7551 @2.55Gz with 32 Cores / 64 Threads
- RAM:
  - 512GB DDR4 @ 1966Mhz

Melfu High-Capacity Node

- CPU:
  - 2 x AMD EPYC 7551 @2.55Gz with 64 Cores / 128 Threads
- RAM:
  - 2TB (1.8TB) DDR4 @ 3200Mhz

2.25 FAQ

Below are some of the common steps that the team has been asked to resolve more than once, so we put them here to (hopefully) answer your questions before you have to wait in the Ticket Queue!

2.25.1 When Connecting, Host Not Found?

When attempting to connect to the HPC, you receive a message that says 'Could not find deepthought.flinders.edu.au'.

1. If you are on campus, contact ServiceDesk via ServiceOne or Phone.
2. If you are off campus or working remotely, connect to the VPN and retry.

2.25.2 What are the SLURM Partitions?

There are three at this point:

- general
- gpu
- melfu

You can omit the

- #SBATCH partition=<name> directive

as the sane-default for you is the general partition. If you need access to the GPU's you **must** use the gpu queue.

2.25.3 SLURM - Tasks & OpenMPI/MPI

When running jobs enabled with OpenMPI/MPI, there is some confusion around how it all works and what the correct settings for SLURM are. The biggest confusion is around what a 'Task' is, and when to use them.

Tasks

Think of a task as a 'Bucket of Resources' you ask for. It cannot talk to another bucket without some way to communicate - this is what OpenMPI/MPI does for you. It lets any number of buckets talk to each other.

When asking SLURM for resources, when you ask for N Tasks, you will get N tasks of X size that you asked for, and all are counted against your usage. For example

- -N12 –cpus-per-task=10 –mem-per-cpu=2G

Will get you a combined *total* of 120 CPUs and 240GB of RAM *spread across 12 individual little instances*.

Running the Job

There are several ways to correctly start OpenMPI/MPI based programs. SLURM does an excellent job of integrating with OpenMPI/MPI, so usually it will 'Just Work'. Its highly dependant upon how the program is structured and written. Here are some options that can help you boot things when they do not go to plan.

- mpirun - bootstraps a program under MPI. Best tested under a manual allocation via salloc.
- srun - Acts nearly the same as 'sbatch' but runs immediately via SLURM, instead of submitting the job for later execution.

OOM Killer

Remember, that each 'task' is its own little bucket - which means that SLURM tracks it individually! If a single task goes over its resource allocation, SLURM will kill it, and usually that causes a cascade failure of your program, as you suddenly have a process missing.

2.25.4 Installing ISoSeq3

IsoSeq3, from Pacific Bio Sciences has install instructions that won't get you all the way on DeepThought. There are some missing packages and some commands that must be altered to get you up and running. This guide will assume that you are starting from scratch, so feel free to skip any steps you have already performed.

The steps below will:

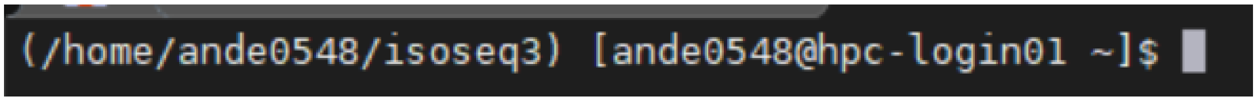
- Create a new Virtual Environment
- Install the dependencies for IsoSeq
- Install IsoSeq3
- Alter a SLURM script to play nice with Conda

Conda/Python Environment

Only thing you will need to decide is 'where you want to store my environment' you can store it in your /home directory if you like or in /scratch. Just put it someplace that is easy to remember. To get you up and running (anywhere it says FAN, please substitute yours):

- module load miniconda/3.0
- conda create -p /home/FAN/isoseq3 python=3.9
- source activate /home/FAN/isoseq3
- You may get a warning saying 'your shell is not setup to use conda/anaconda correctly' - let it do its auto-configuration. Then Issue
  - source ~/.bashrc

When all goes well, your prompt should read something similar to



```
(/home/ande0548/isoseq3) [ande0548@hpc-login01 ~]$ █
```

Notice the (/home/ande0548/isoseq3)? Thats a marker to tell you which Python/Conda Environment you have active at this point.

BX Python

The given bx-python version in the wiki doesn't install correctly, and if it *does* work, then it will fail on run. To get a working version, run the following.

- conda install -c conda-forge -c bioconda bx-python

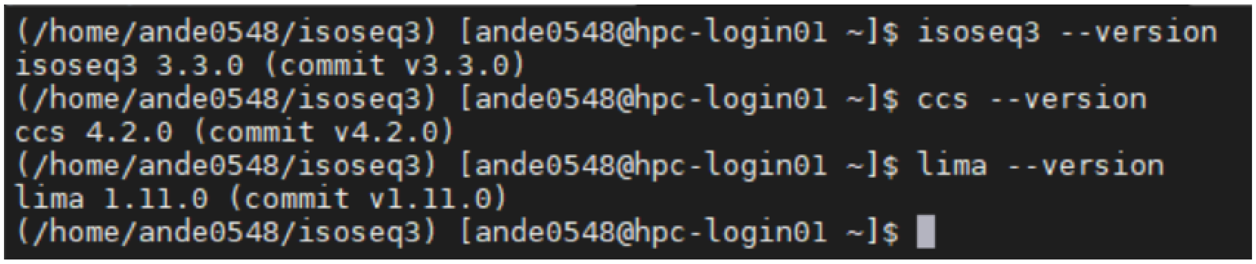
Which will get you a working version.

IsoSeq3

Finally, we can install IsoSeq3 and its dependencies.

- conda install -c bioconda isoseq3 pbccs pbcoretools bamtools pysam lima

Will get you all the tools installed into your virtual environment. To test this, you should be able to call the individual commands, like so.



```
(/home/ande0548/isoseq3) [ande0548@hpc-login01 ~]$ isoseq3 --version
isoseq3 3.3.0 (commit v3.3.0)
(/home/ande0548/isoseq3) [ande0548@hpc-login01 ~]$ ccs --version
ccs 4.2.0 (commit v4.2.0)
(/home/ande0548/isoseq3) [ande0548@hpc-login01 ~]$ lima --version
lima 1.11.0 (commit v1.11.0)
(/home/ande0548/isoseq3) [ande0548@hpc-login01 ~]$ █
```

SLURM Modifications

You may get an issue when you ask SLURM to run your job about CONDA not being initialised correctly. This is a very-brute-force hammer approach, but it will cover everything for you.

Right at the start of your script, add the following lines:

- module load miniconda/3.0
- conda init –all
- source /home/FAN/.bashrc
- conda activate /path/to/conda/environment

This will load conda, initialises (all of your) conda environment(s), force a shell refresh and load that new configuration, then finally load up your environment. Your job can now run without strange conda-based initialisation errors.

2.25.5 BX-Python

The given bx-python is a problematic module that appears in many of the BioScience packages in Conda, below will get you a working, Python 3 version. These steps are the same as the installation for IsoSeq3, but given how often this particular python package gives the support team issues, it gets its own section!

- conda install -c conda-forge -c bioconda bx-python

2.25.6 My Jupyter Kernel Times Out

This is usually caused by one of two things:

- HPC has allocated all its Resources
- Incorrect Conda Setup

HPC Is Busy

You job will time out when the HPC is busy, as your job cannot get an allocation within 30 seconds (or so). If you do not see a file like 'slurm-<NUMBER>.out' in your /home directory, then the HPC cannot fit your kernel's requested allocation as all resources are busy.

To solve the above, you can either:

- Recreate a Kernel with lower resource requirements
- Wait for the HPC to be less busy

A sneaky command from the HPC Admin Team: `sinfo -No "%17n %13C %10O %10e %30G"`. This gets you a layout like so:

```
HOSTNAMES          CPUS(A/I/O/T) CPU_LOAD   FREE_MEM   GRES
hpc-node001        0/64/0/64     0.46       241647     gpu:tesla_v100:2(S:2,6)
hpc-node002        0/64/0/64     1.86       250777     gpu:tesla_v100:2(S:2,6)
hpc-node003        64/0/0/64     20.44      240520     (null)
hpc-node004        64/0/0/64     19.46      244907     (null)
hpc-node005        64/0/0/64     18.59      241284     (null)
hpc-node006        64/0/0/64     17.37      244390     (null)
hpc-node007        64/0/0/64     14.50      221633     (null)
```

(continues on next page)

(continued from previous page)

```
hpc-node008       64/0/0/64     18.06     211002     (null)
hpc-node009       64/0/0/64     19.27     206833     (null)
hpc-node010       64/0/0/64     19.39     233411     (null)
hpc-node011       64/0/0/64     20.51     221966     (null)
hpc-node012       64/0/0/64     19.06     181808     (null)
hpc-node013       64/0/0/64     20.35     221835     (null)
hpc-node014       60/0/4/64     4.00      151584     (null)
hpc-node015       64/0/0/64     18.01     191874     (null)
hpc-node016       64/0/0/64     11.04     214227     (null)
hpc-node017       0/64/0/64     0.00      512825     (null)
hpc-node018       0/64/0/64     0.03      61170      (null)
hpc-node019       128/0/0/128   515.85    1929048    (null)
hpc-node020       128/0/0/128   30.31     1062956    (null)
hpc-node021       128/0/0/128   38.10     975893     (null)
hpc-node022       0/64/0/64     0.06      119681     gpu:tesla_v100:1(S:2)
```

What you want to look at is that first and second numbers in the CPUS Column. The first is 'Allocated' and the second is 'Available for Usage'. This above example shows that the GPU queue is empty (0/64) but the general queue is busy (64/0).

Incorrect Conda Environment Setup

The timeout error can also be caused by missing a required package for the custom WLM Integration to work correctly.

This means that the job started, but could not connect your Jupyter Notebook correctly. If you look in your home directory, you will see the previously mentioned 'slurm-<NUMBER>.out' file. Right at the very bottom of the file (its quite long, with lots of debugging information in it) you will see a message similar to:

- `command not found ipykernel-wlm`

To fix this type of 'command not found' error for ipykernel or similar - go back to the Jupyter Hub Conda Setup instructions, and double check that you have installed *all* of the needed packages.

2.26 Known Issues

Below are the 'known issues' with the documentation at this point. This is not an issue tracker for the HPC, so please don't try and lodge an issue with DeepThought here!

2.26.1 PDF Version

- Images are missing in some sections on the PDF Version
- Alt-Text appears when it should not in the PDF Version

2.26.2 EBPUB Version

- EPUB Version is missing some pages of content

2.26.3 Web / ReadTheDocs / Online Version

None at the moment.

2.27 Fair Usage Guidelines

The Deepthought HPC provides moderate use at no charge to Flinders University Colleges. This is enforced by the Fairshare System and is under constant tweaking and monitoring to ensure the best possible outcomes. The current split of resources between colleges is:

- 45 % for CSE
- 45 % for CMPH
- 10 % for General

For example, if the HPC had 1000 'shares' that represent its resources, the following would be demonstrative of how they are allocated:

- 450 'shares' for CSE
- 450 'shares' for CMPH
- 100 'shares' for General

2.28 Storage Usage Guidelines

See the page at: StorageGuidelines, for details on what storage is present and other details. A general reminder for HPC Storage:

- All storage is *volatile* and no guaranteed backup systems are in place
- Cleanup you /home and /scratch regularly
- Cleanup and /local storage you used at the end of each job

2.29 Software Support Guidelines

HPC Users are encouraged to compile and install their own software when they are comfortable to do so. This can be done freely on the *compute nodes*. The HPC Head-Node is a differnet hardware architecture to the compute nodes. You will most likely encounter **SIG4, Illegal Instruction** if you attempt to compile programs on the head-node for usage on the compute-nodes.

The HPC Support team cannot maintain and provide active support for every piece of software that users of the HPC may need. The following guidelines are an excerpt from our HPC Software Support Policy to summarise the key points. For more information on how the software can be loaded/unloaded on the HPC, head on over to the Module System.

2.29.1 Supported Software Categories

The following categories are how the HPC Team asses and manage the differing types of Software that are present on the HPC. For more information, each will have their own section.

- Core 'Most Used' Programs
- Licensed Software
- Libraries
- Toolchains
- Transient Packages
- Interpreters / Scripting Interfaces

Core 'Most Used' Programs

Holds the most used packages on the HPC. The HPC Team monitors the loading and unloading of modules, so we can manage the lifecycle of software on the HPC. As an example, some of the most used program on the HPC are:

- R
- Python 3.9
- RGDAL
- CUDA 11.2 Toolkit

While not an exhaustive list of the common software, it does allow the team to focus our efforts and provide more in-depth support for these programs. This means they are usually first to be updated and have a wider range of tooling attached to them by default.

Licensed Software

Licensed Software covers the massive packages like ANSYS (which, all installed is about 300 *gigabytes*) which are licensed either to the HPC specifically or obtained for usage via the University Site licenses. This covers things like:

- ANSYS Suite (Structures, Fluids, Electronics, PrepPost & Photonics)

Libraries

Generally, these libraries are required as dependencies for other software, however there are some core libraries that are used more than others. As an example, the Geometry Engine - Open Source (GEOS) Library is commonly used by other languages (R, Python) to allow for Geo-Spatial calculations. Some of the common libraries include:

- GEOS
- ZLib
- ImageMagik
- OpenSSL

Most of these are useful in a direct manner only for software compilation or runtime usage.

Unless compiling your own software, you can safely ignore this section - the Module System takes care of this for you.

Toolchains

Generally we support the FOSS (Free Open Source Software) Toolchain, comprising of:

- GNU C, C++ & Fortran Compilers (gcc, g++ and gfortran)
- GNU Bin Utils
- Open MPI Library
- Open BLAS, LAPACK & ScaLAPACK
- FFTW Library

Transient Packages

Listed here for completeness, these packages are install-time dependencies or other packages that do not fit within the above schema.

Scripting Languages

Interpreters like Python & R (Perl, Ruby, Scala, Lua etc.) are complex things and have their own entire ecosystems of packages, versioning and tools. These Scripting Interfaces (The technical term is 'Interpreters') are all managed as their own standalone aspect to the HPC.

Using Python as an example you have:

- The interpreter 'Python'
- The package manager 'Pip'
- The Meta-Manager 'Conda'/'Mini-Conda'
- The Virtual Environments Manager 'venv'

Each interacting in slightly different ways and causing other issues. To ensure that the HPC team can support a core set of modules the interpreters are only updated when:

- Security patches are needed
- A new *Major* Version is available
- A commonly requested feature requires an upgrade

Versioning Support

Most major packages will be supported in a Latest - 1 fashion. Below show an example when a package would be updated in the quarterly package upgrade cycle.

- Latest Version: 2020a
- Installed Version: 2019a
- Supported Version: 2019b

As not all software follows such clean release patterns, the HPC Team will hold final say on updating a piece of software in the global module lists.

2.30 Upgrade Cycles

The HPC Team does their best to adhere to the following cycle for upgrades for software and associated systems.

Software Category	Upgrade Cycle	Outage Required	Versioning Type
Core Supported Programs	Quarterly	No	N - 1
Core Licensed Programs	Bi-Yearly	No	N - 1
OS & Managerial Tools	Yearly	Yes	Latest
Software Images	Bi-Yearly	Partial	Latest
Scripting Interfaces	Quarterly	No	Major, Security & Feature Minor
Scripting Modules	Quarterly	No	Latest

2.31 HPC Etiquette

The HPC is a shared resource, and to help make sure everybody can continue to use the HPC together, the following provides some expected behaviour.

2.31.1 Head / Login / Management Nodes

1) The Head / Login / Management Nodes are to be used for light, single-threaded tasks only.
2) If it takes more than 5-10 minutes or > 2-3GB of RAM, do NOT run it on the head node.
3) Some acceptable tasks include:
    - Compiling software for your own use
    - Transferring / Decompressing Files
    - Light Pre/Post Processing
    - SLURM Job Management

2.31.2 General Cluster Rules

1) Use only the resources you need, remembering this is shared resource.
2) Clean up your disk usage in /scratch and /home regularly.
3) Do not attempt to bypass security controls.
4) Do not attempt to bypass job Scheduling.
5) Do not access the compute nodes directly.

2.31.3 Permissions & Access Levels

The HPC has the following capabilities. If a capability is NOT listed, then you are not permitted to perform the action. Security is approached via a list of allowed actions, not a list of denied actions.

General User

1) Full read & write permission to your own /scratch/FAN and /home/FAN locations
2) The ability to compile and run your own software, stored in your /home directory
3) The ability to run any module present in the module system
4) Manipulate your own jobs via SLURM

Group Admins

Trusted partners may be appointed 'Group Administrators' at the *sole discretion of the HPC Team* allowing them to:

1) Perform all actions of a general user
2) Manipulate SLURM actions of users under their remit

Permissions that are Never Granted

The following is a non-exhaustive list of permissions that are never, and will never, be granted to end-users. The HPC is a complicated system and, while the Support Team is asked for these permissions quite often, the potential inadvertent damage to systems means these permissions cannot be provided.

1) Root or Sudo access
2) Global 'Module' Software Installation
3) Elevated Access to the HPC System
4) Access to the Cluster Management System
5) Access to the Storage Analytics

2.31.4 If You Break These Rules

If you break these rules the HPC Team may take any or all of these actions:

1. Cancellation of running tasks/jobs
2. Removal problematic files and/or programs
3. Warning of expected behaviours
4. Revocation of HPC Access

2.32 Acknowledgements

We recognise the respect the trademarks of all third-party providers referenced in this documentation. Please see the respective EULAs for software packages used in configuring your own environment based on this knowledgebase.

2.33 License