DeepThought Documentation

Aug 16, 2021

User Documentation

The new Flinder 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 1

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

CHAPTER 2

Table of Contents

2.1 Getting Access to DeepThought

This page will guide you on the steps to get access and then connect to the HPC on your operating system of choice.

2.1.1 Flinders Staff / HDR Students

1. Create a ServiceOne Ticket asking for Access to the HPC. Currently the request is under the following path: Research Services -> Undertake Research -> Access to DeepThought HPC.
2. Pick your Operating System (*Unix/Linux/MacOS* or *Windows*)
3. Read up on some basic *SLURM*

2.1.2 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 first and then contact the HPC Support Team via email.

2.1.3 Unix/Linux/MacOS

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.

The Windows Sub-System for Linux

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. Generally, if you are using the WSL, then following the Unix/Linux instructions.

Getting Connected

The simplest manner is to open up a terminal window and type in the following, substituting FAN for your FAN as needed.

- ssh FAN@deepthought.flinders.edu.au

If all is OK it will ask for your password. Enter the same password you use to login to OKTA.

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

Success

Upon a successful login, you should get a screen similar to this:

```
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 ~]$
```

If so, you are now connected and ready to start using the HPC!

SSH Keys

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

2.1.4 Windows

To connect to Deep Thought a SSH application 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.

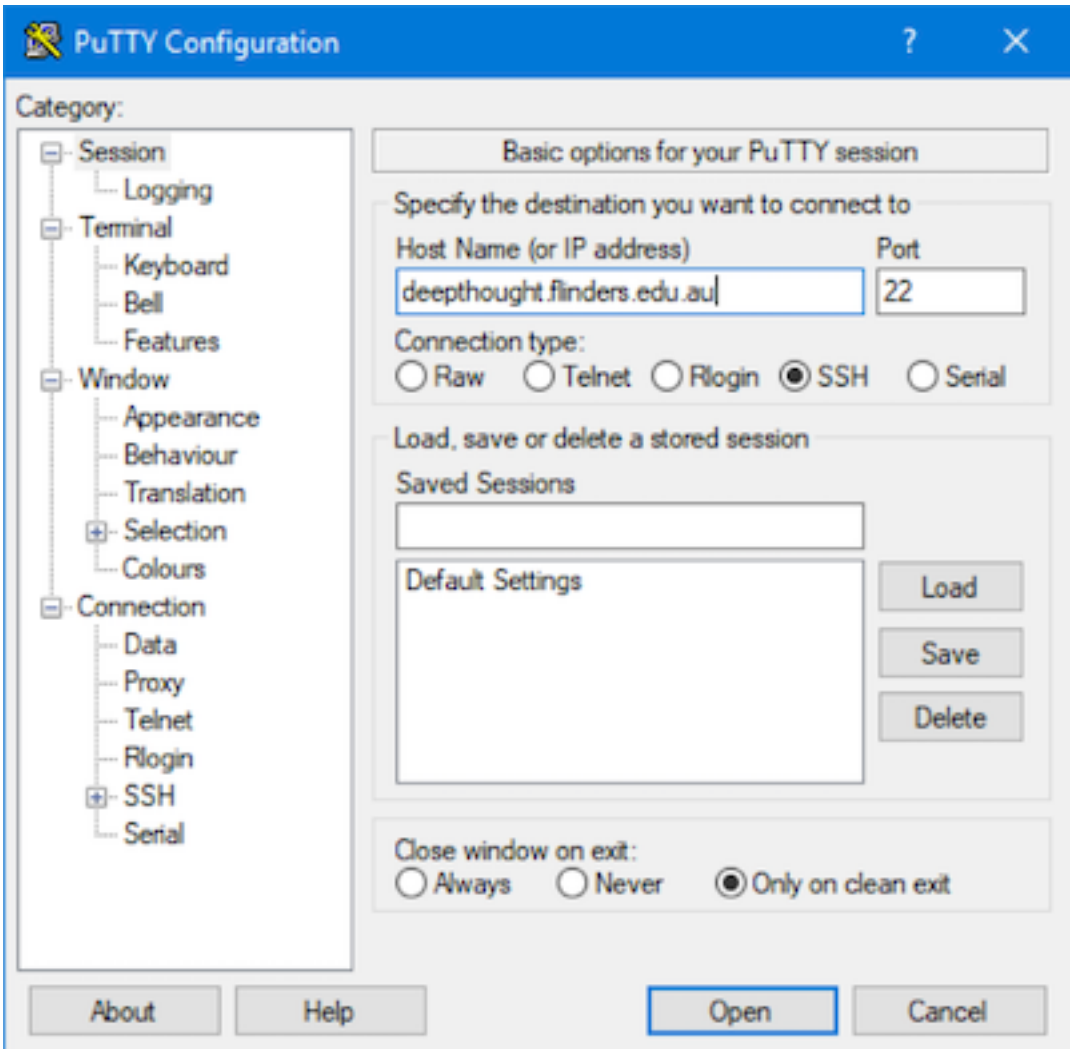
Client Options

- Putty
- KiTTY

- BitVise Client
- MobaXterm

Getting Connected on Windows

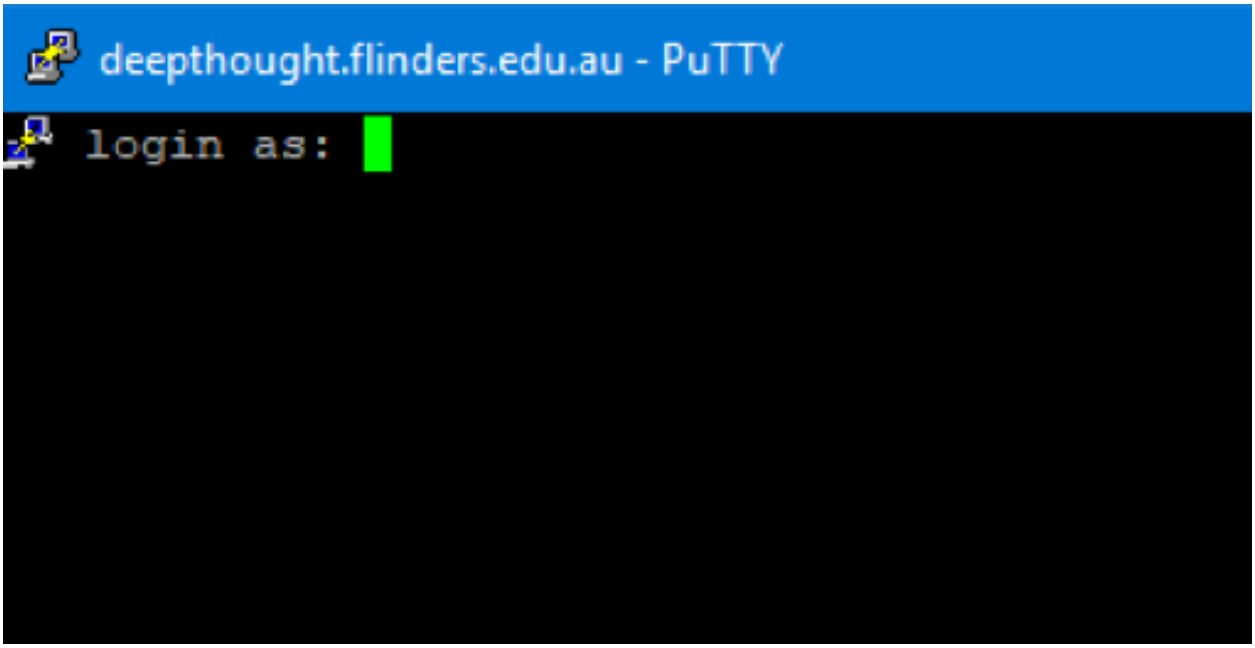
Open PuTTy, and you are presented with this screen:



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

Logging In

If all has gone well, you will be presented with this screen:



- Your Username is your FAN
- Your Password is your FAN Password.

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

Successful Login

Upon a successful login, you should get a screen similar to this:

```
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 ~]$
```

If so, you are now connected and ready to start using the HPC.

Windows SSH Keys

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.

2.2 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 two main storage tiers. Firstly our bulk storage is the 'Scratch' area - and is slower, spinning Hard-Disk Drives (HDD's). The smaller, hyper-fast NVMe Solid-State Drives (SSD's) are located at /local and is much smaller. For the exact specifications and capacities, see the System Specifications.

There is a critical difference between these two locations. The /scratch area is a common storage area. You can access it from all of the login, management and compute nodes on the HPC. This is not the same as /local, which is only available on each compute node. That is - if you job is running on Node001, the /local only exists on that particular node - you cannot access it anywhere else on the HPC.

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.

2.2.1 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
/home	All Nodes	~12TB
/local	Individual Compute Nodes	~400GB or ~1.5TB
/r_drive/<folder>	Head Nodes	N/A

Attention: /The r_drive/ location is NOT the University R:\ Drive - the ability to move data between R:\ and the HPC is currently undergoing testing.

The /r_drive/ locations are data mount points from the now defunct eRSA Project and are slowly being phased out. Any point under /r_drive/ will *auto mount on access*. Just attempt to touch or change to the correct directory under the /r_drive/ path and the HPC will handle this automatically for you. Until you do this, the directory **will be invisible**.

2.2.2 Usage Guidelines

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

/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 promp, 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
- Results from Jobs.
- 'Small' Data-Sets (<5GB)

/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

/Local

Local is the per-node, high speed flash storage that is specific to each node. When running a job, you want to run your data-sets on /local if at all possible - its the quickest storage location on the HPC. You MUST cleanup /local once you are done.

What to Store in /local

Only *transient files* should live on /local. Anything that your job is currently working on should 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.

2.3 HPC Research Data Flow

The following diagram illustrates the overall location of the HPC in the Research Data Management flow.

Fig. 1: DeepThought HPC Research Data Flow Diagram

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

2.4 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.

2.4.1 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.

2.4.2 Before we get started

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.

On DeepThought, are two main storage tiers, with a smaller pool for your documents and scripts. Firstly our bulk storage (approx 250TB) is the 'Scratch' area (located at /scratch/user/$FAN) - and is slower, spinning Hard-Disk Drives (HDD's). The smaller, hyper-fast NVMe Solid-State Drives (located at /local) are roughly 400GB on the 'standard' nodes (1-16) and 1.5TB on the 'high-capacity' nodes (19-21).

There is a critical difference between these two locations. The /scratch area is a common storage area. You can access it from all of the login, management and compute nodes on the HPC. This is not the same as /local, which is only available on each compute node. That is - if you job is running on Node001, the /local only exists on that particular node - you cannot access it anywhere else on the HPC.

- /home/$FAN
- /scratch/$FAN

There following location is treated specially as it holds the older eRSA data and is slowly being phased out.

- /r_drive/

This location will surface specific mount points on request. If you have access to these locations, they auto-mount **on usage**. That is, simply try and touch or access the correct location that you have access to, and it will be handled automatically for you. For example, if you have access to a mount point called 'molecular_data', then the following command with surface that mount point to you - `cd /r_drive/molecular_data/`.

/Home

Your 'home' directories. This is a small amount of storage (~11TB total) to store your small bits and pieces. This is the analogous to the Windows 'Documents' folder.

At a command promp, your home directory usually gets 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
- Results from Jobs.

- 'Small' Data-Sets (<5GB)

/Scratch

Scratch is your working space. Depending upon your dataset, you may need to run your job here - this is not optimal and will be much slower than running it from /local. Scratch is still not an area to store your data permanently - there are no backups in place for the HPC, so ensure you follow the HPC Research Data Flow and the HPC Job Data Flow.

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
- Intermediate files

2.4.3 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:

```
bash-3.2$ scp FILENAME FAN@deepthought@flinders.edu.au:/home/FAN
```

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 Deepthough '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
```

2.4.4 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

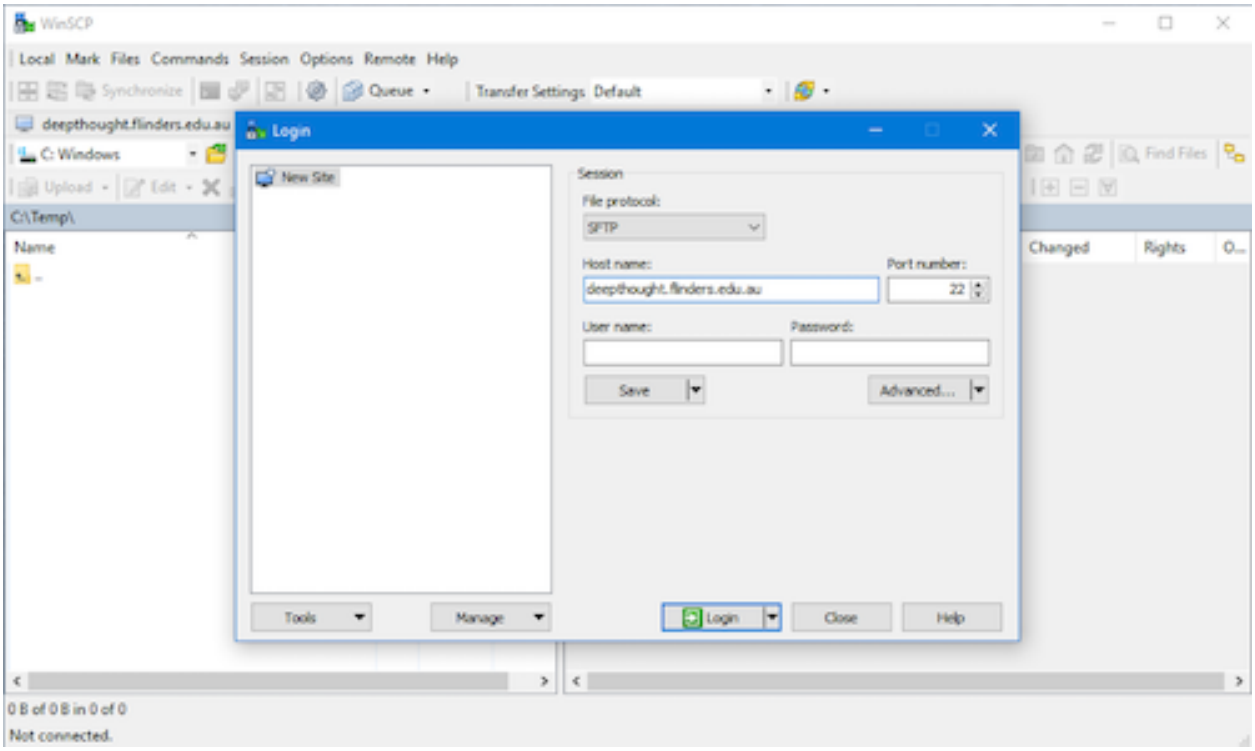
This is not an exhaustive list - feel free to use whatever you wish that supports the SFTP protocol.

- WinSCP
- FileZilla

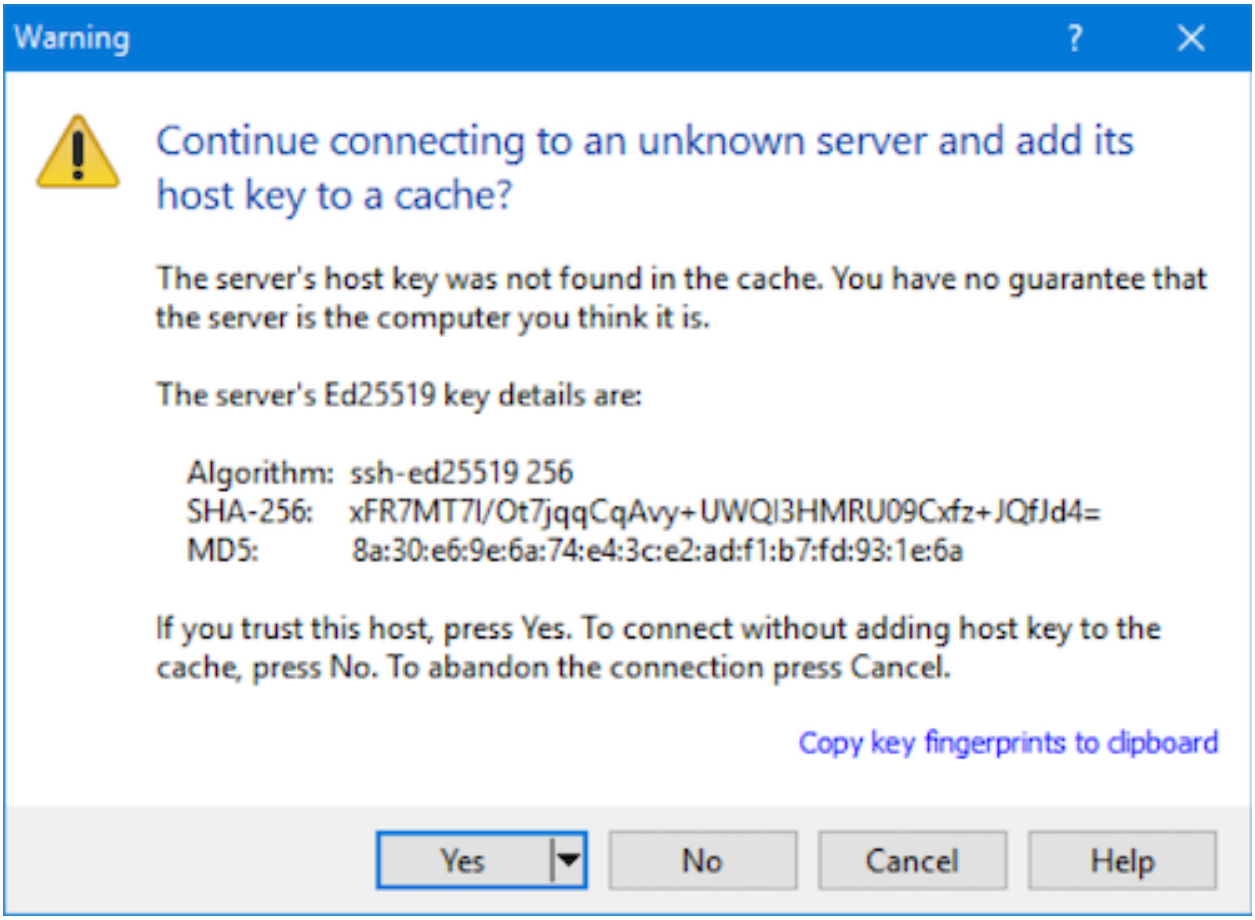
This guide will focus on WinSCP.

Getting Connected with WinSCP

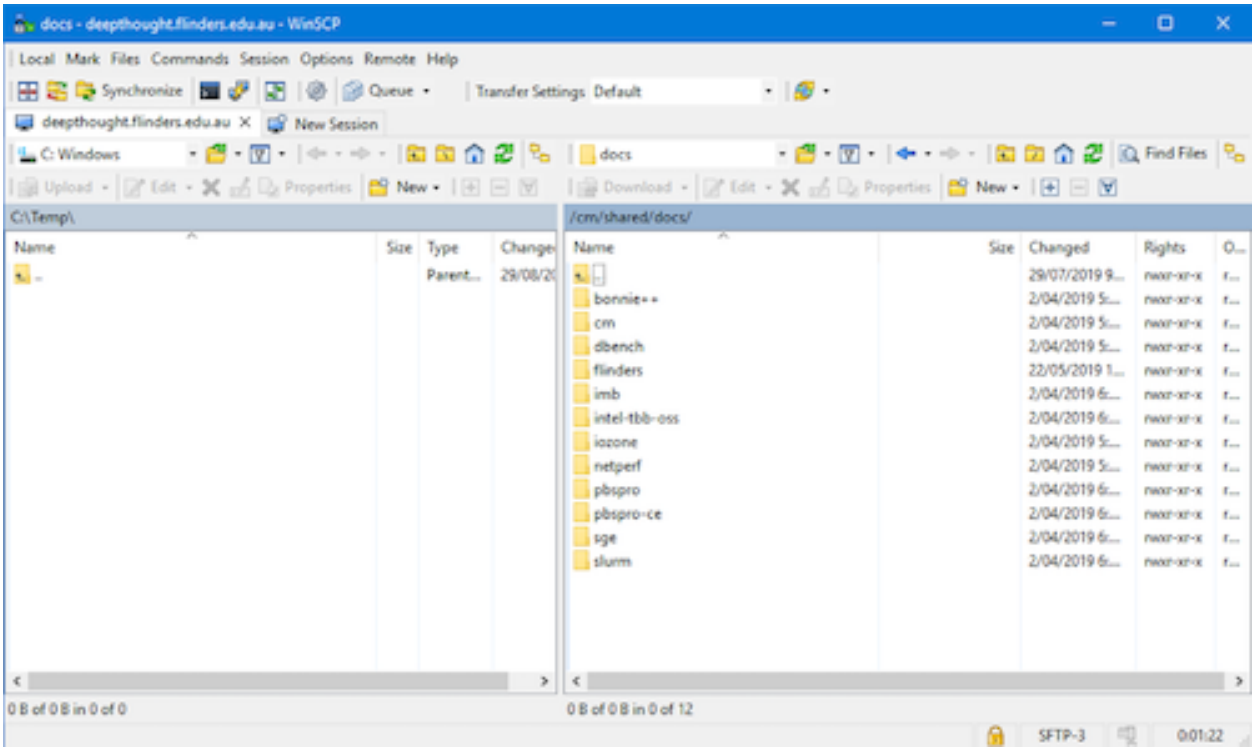
Open WinSCP, enter deepthought.flinders.edu.au as the host to connect to, and click Login. You should have a screen that looks like this.



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



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



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

2.5 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!

2.5.1 Tips and Tricks

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

2.5.2 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.

2.6 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.

Fig. 2: DeepThought HPC Job Data Flow Diagram

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

2.7 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."

2.7.1 System Specifications

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

2.7.2 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 people share and could use really.

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 = 65,536,000` (Measured Per MB)

Total: `65,600,000`

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.

2.7.3 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>
```

2.7.4 SLURM: Advanced

Job Arrays

Job arrays is a popular strategy to process large numbers of a same workflow repetitively in one go, often reduce 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

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

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.

2.7.5 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

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.

2.7.6 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
#################################################################
# 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!
#
#SBATCH --job-name=FAN_JOBNAME
#
#################################################################
# If you want email updates form SLURM for your job.
# Change MYEMAIL to your email address
```

(continues on next page)

(continued from previous page)

```
#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
#################################################################
# You can leave this commented out, or submit to hpc_general
# Valid partitions are hpc_general and hpc_melfeu
##SBATCH --partition=PARTITIONNAME
#
#################################################################
# Tell SLURM how long your job should run for, at most.
# SLURM will kill/stop the job if it goes over this amount of time.
# Currently, this is unlimited - however, but the longer your job
# runs, the worse your Fairshare score becomes!
#
# In the future this will have a limit, so best to get used to
# setting it now.
#
# The command format is as follows: #SBATCH --time=DAYS-HOURS
#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 cant talk to
# another bucket!
#
#SBATCH --ntasks=1
# If each task will need more that a single CPU, then alter this
# value. Remeber, 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
#################################################################
# 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 it. 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)
#SBATCH --mem-per-cpu=4000
# Or, you can ask for a 'total amount of RAM'
##SBATCH --mem=12G
#################################################################
```

(continues on next page)

(continued from previous page)

```
# Change the number of GPU's required and 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.
# This line requests 0 GPU's by default.
#
#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 add miniconda/3.0 cuda10.0/toolkit/10.0.130
#module add miniconda/3.0
#module add 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.

# Move your dataset to /local
cd /local
mkdir -p /local/$SLURM_JOBID/$SLURM_ARRAY_TASK_ID/
cd /local/$SLURM_JOBID/$SLURM_ARRAY_TASK_ID/
cp /scratch/user/<FAN>/dataset ./

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


################################################################
# Once you job has finished its processing, copy back your results
# and ONLY the results to /scratch, then cleanup the temporary
# working directory

cp /local/$SLURM_JOBID/$SLURM_ARRAY_TASK_ID/results/ /scratch/user/<FAN>/???

rm -rf /local/$SLURM_JOBID/

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

2.8 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.

2.8.1 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 ServiceOne ticket, or contact the HPC Support team at their email.

2.8.2 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

2.8.3 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.

2.8.4 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-gxcq1oq        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/2.33                                    (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.

2.8.5 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.

2.9 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.

2.9.1 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.

2.9.2 Storage Layout

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

Per node /local: ~400GB to 1.5TB, depending on node layout

2.9.3 Node Breakdown

- 19 Compute Nodes, with ~1800 Cores and ~10TB of RAM total
- 2 Login Nodes, with High-Availability Failover
- 4 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
  - ~400GB of NVMe SSD's

GPU Nodes

There are 2 dedicated GPU nodes, each with:

- 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
  - ~400GB of NVMe SSD's

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
  - 1.5TB of NVMe SSD's

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

2.10 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!

2.10.1 What are the SLURM Partitions?

There are three at this point:

- hpc_general
- hpc_gpu
- hpc_melfeu

You can omit the

- #SBATCH partition=<name> directive

as the sane-default for you is the hpc_general partition. If you need access to the GPU's you **must** user the hpc_gpu queue.

2.10.2 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 immediacy 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 with the rest of your program, as you suddenly have a process missing.

2.10.3 IsoSeq3: Installation

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=3.7
- 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

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

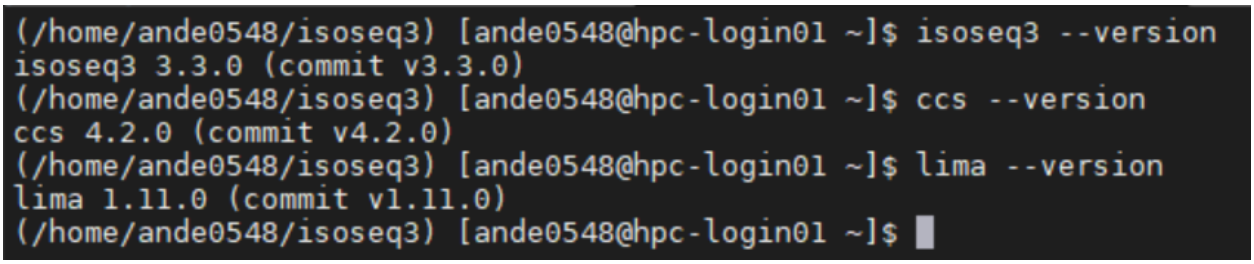
Which will get you a working version.

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.



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.

2.10.4 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

2.11 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!

2.11.1 PDF Version

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

2.11.2 EBPUB Version

- EPUB Version is missing some pages of content

2.11.3 Web / ReadTheDocs / Online Version

None at the moment.

2.12 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

2.13 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 backup systems are in place
- Cleanup you /home and /scratch regularly
- Cleanup and /local storage you used at the end of each job

2.14 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 head node(s).

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.

2.14.1 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 3.8
- RGDAL
- CUDA 10.1 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.

2.15 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

2.16 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 behavior.

2.16.1 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

2.16.2 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.

2.16.3 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

2.16.4 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 behaviors
4. Revocation of HPC Access

2.17 Citing DeepThoughtHPC

To cite DeepThoughtHPC service, please use following or download BibTex/EndNote file. Your citation will demonstrate the research impact that DeepThoughtHPC service brings and support ongoing funding efforts for this service.

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

2.18 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.

CHAPTER 3

License