iceberg Documentation

Release

October 23, 2015

1 Using iceberg **3**
1.1 Getting Started . . . 3
1.2 Iceberg's Queue System . . . 5
1.3 Filestore on Iceberg . . . 7

2 Iceberg Software **11**
2.1 Applications . . . 11
2.2 Libraries . . . 48
2.3 Compilers . . . 71
2.4 MPI . . . 79
2.5 Modules . . . 80

3 GPU Computing **83**
3.1 GPU Community and NVIDIA Research Centre . . . 83
3.2 GPU Hardware on iceberg . . . 83
3.3 Requesting access to GPU facilities . . . 83
3.4 Interactive use of the GPUs . . . 84
3.5 Compiling on the GPU using the NVIDIA Compiler . . . 84
3.6 Compiling on the GPU using the PGI Compiler . . . 84
3.7 GPU enabled Software . . . 85

4 Troubleshooting **87**
4.1 Frequently Asked Questions . . . 87

5 Cluster Specifications **93**
5.1 Summary of iceberg hardware specs. . . . 93
5.2 Worker Nodes CPU Specifications . . . 93
5.3 GPU Units Specifications . . . 93
5.4 Filestore . . . 94
5.5 Filestore Allocations . . . 94
5.6 Filestore recovery and backup policies . . . 94
5.7 Software and Operating System . . . 95

6 Glossary of Terms **97**

Welcome to the **Iceberg** user manual. Iceberg is the University of Sheffield's central High Perfomance Computing (HPC) resource. Its mission is to meet the intensive computing requirements of research projects at Sheffield University.

This manual aims to provide you all the information you need to use iceberg.

Using the Iceberg Service

- *Getting Started* - If you are new to HPC and Iceberg, we suggest you start here.
- *Iceberg Software* - Details about the software installed on Iceberg and how to use it.
- *Iceberg's Queue System* - Details about submitting interactive and batch jobs.
- *Troubleshooting* - Start here if you have a problem using the system.

About the Iceberg Service

- *Cluster Specifications* - Information about Iceberg's hardware.
- *Filestore on Iceberg* - Information about the amount of disk space you get on Iceberg and the different types of filesystem available.
- *The GPU Computing section* - GPU computing on Iceberg.

Research Computing Team

The research computing team are the team responsible for the iceberg service, as well as all other aspects of research computing. If you require support with iceberg, training or software for your workstations, the research computing team would be happy to help. Take a look at the Research Computing website or email research-it@sheffield.ac.uk.

CHAPTER

ONE

USING ICEBERG

The first step in using iceberg is getting an account and running an interactive session, if you don't know how to do this you should start with *Getting Started*.

Once you have connected to iceberg in interactive mode you can run jobs which take less than **8 hours** and which are graphical or require user interaction. For more information on what applications you can run read *Iceberg Software*.

If you require more memory or more processes when running interactive jobs you will need to tell the cluster management software (the scheduler) what you need. For more information on this read *Running an Interactive Shell*.

If you want to run jobs which are not interactive you should submit these to the 'queue'. Jobs in the queue will be run when there is free resources for them. To learn how to use the queue see *Submitting Jobs to the queue*.

1.1 Getting Started

If you have not used a High Performance Computing (HPC) cluster, Linux or even a command line before this is the place to start. This guide will get you set up using iceberg in the easiest way that fits your requirements.

1.1.1 Getting an Account

Before you can start using iceberg you need to register for an account. Accounts are availible for staff by emailing research-it@sheffield.ac.uk.

The following categories of students can also have an account on iceberg with the permission of their supervisors:

- Research Postgraduates
- Taught Postgraduates - project work
- Undergraduates 3rd & 4th year - project work

Student's supervisors can request an account for the student by emailing research-it@sheffield.ac.uk.

Note: Once you have obtained your iceberg username, you need to initialize your iceberg password to be the same as your normal password by using the CICS synchronize passwords system.

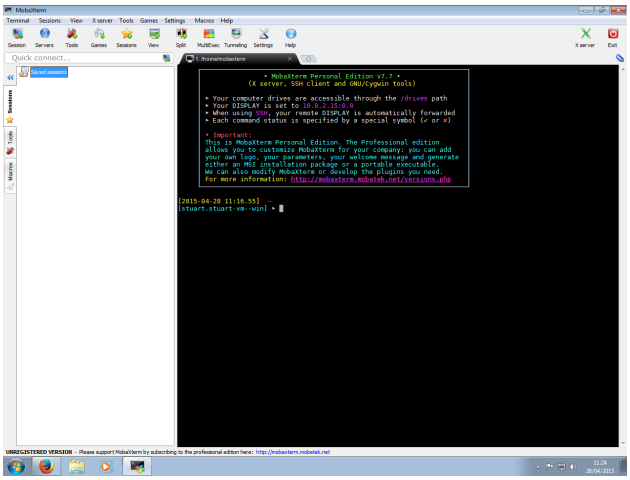
1.1.2 Connecting to iceberg (Terminal)

Accessing iceberg through a terminal is easy and the most flexible way of using iceberg, as it is the native way of interfacing with the linux cluster.

Windows

Download and install mobaxterm.

Running MobaXterm should display the following screen:



Once you have this screen you can jump to *Connect to iceberg*.

Mac OS/X and Linux

Linux and Mac OS/X both have a terminal emulator program pre-installed.

Open a terminal and then go to *Connect to iceberg*.

Connect to iceberg

Once you have a terminal open run the following command:

```
ssh -X <username>@iceberg.shef.ac.uk
```

where you replace *<username>* with your CICS username.

This should give you a prompt resembling the one below:

```
[te1st@iceberg-login2 ~]$
```

at this prompt type:

```
qsh
```

like this:

```
[te1st@iceberg-login2 ~]$ qsh
Your job 135355 ("INTERACTIVE") has been submitted
waiting for interactive job to be scheduled ....
Your interactive job 135355 has been successfully scheduled.
```

which will pop up another terminal window, which supports graphical applications.

Note: Iceberg is a compute cluster. When you login to the cluster you reach one of two login nodes. You **should not** run applications on the login nodes. Running qsh gives you an interactive terminal on one of the many worker nodes in the cluster.

If you only need terminal based (CLI) applications you can run the qrsh command. Which will give you a shell on a worker node, but without graphical application (X server) support.

What Next?

Now you have connected to iceberg, you can look at how to submit jobs with *Iceberg's Queue System* or look at *Iceberg Software*.

1.2 Iceberg's Queue System

To manage use of the iceberg cluster, there is a queue system (SoGE, a derivative of the Sun Grid Engine).

The queue system works by a user requeuesting some task, either a script or an interactive session, be run on the cluster and then the scheduler will take tasks from the queue based on a set of rules and priorities.

1.2.1 Running an Interactive Shell

If you wish to use the cluster for interactive use, such as running applications such as MATLAB or Ansys, or compiling software, you will need to requeuest that the scheduler gives you an interactive session. For an introduction to this see *Getting Started*.

There are two commands which give you an interactive shell:

```
[te1st@iceberg-login2 ~]$ qsh
```

and:

```
[te1st@iceberg-login2 ~]$ qrsh
```

`qsh` will open a separate xterm terminal and supports running graphical applications. qrsh gives a shell running on a worker node inside the currently open terminal, it does not support graphical applications because it has no X server forwarding configured.

You can configure the resources available to the interactive session by specifying them as command line options to the qsh or qrsh commands. For example to run a `qsh` session with access to 16 GB of virtual RAM:

```
[te1st@iceberg-login2 ~]$ qsh -l mem=16G
```

or a session with access to 8 cores:

```
[te1st@iceberg-login2 ~]$ qsh -pe openmp 8
```

A table of *Common Interactive Job Options* is given below, any of these can be combined together to requeuest more resources.

Note: Long running jobs *should* use the batch submission system rather than requeuesting an interactive session for a very long time. Doing this will lead to better cluster performance for all users.

Common Interactive Job Options

Command	Description
-l h_rt=hh:mm:ss	Specify the total maximum execution time for the job.
-l mem=xxG	Specify the maximum amount (xx) of memory to be used (per process or core).
-pe <env> <nn>	Specify a parallel environment and number of processors.
-pe openmp <nn>	The openmp parallel environment provides multiple threads on one node. It is the most commonly used in an interactive session. <nn> specifies the max number of threads.
-pe openmpi-ib <nn>	The openmpi parallel environment is for distributed memory computing, it enables multiple *processes* to run independently. It is more commonly used in batch mode.

1.2.2 Submitting Jobs to the queue

The power of iceberg really comes from the 'batch job' queue submission process. Using this system, you write a script which executes your job, tell the scheduler how many resources the task requires, then the scheduler will run it when the resources are available. As the task is running, the terminal output and any errors are captured and saved to a disk, so that you can see the output and verify the execution of the task.

Any task that can be executed without any user intervention while it is running can be submitted as a batch job to iceberg. This exculdes jobs that require a GUI, however, many common applications such as Ansys or MATLAB can also be used without their GUIs.

When you submit a batch job, you provide an executable file that will be run by the scheduler. This is normally a script file which provides commands and options to the program you are using. For instance, it might tell Ansys which files to use as input and where to save the output. Once you have a script file, or other executable file, you can submit it to the queue by running:

```
qsh myscript.sh
```

you can also specify extra arguments to this, or at the start of your script, to give you access to more cores or memory or change the maximum execution time, a full list of the availble options are given below.

1.2.3 All Scheduler Options

Com-mand	Description
-l h_rt=hh:mm:ss	Specify the total maximum execution time for the job.
-l mem=xxG	Specify the maximum amount (xx) of memory to be used.
-N	Job name, used to name output files and in the queue list.
-j	Join the error and normal output into one file rather than two.
-M	Email address to send notifications to.
-m bea	Type of notifications to send. Can be any combination of begin (b) end (e) or abort (a) i.e. *-m ea* for end and abortion messages.
-a	Specify the earliest time for a job to start, in the format MMDDhhmm. e.g. -a 01011130 will schedule the job to begin no sooner than 11:30 on 1st January.

1.2.4 Frequently Asked SGE Questions

How do you ensure that a job starts after a specified time?

Add the following line in your submission script

```
#$ -a time
```

but replace `time` with a time in the format MMDDhhmm

For example, for 22nd July at 14:10, you'd do

```
#$ -a 07221410
```

This won't guarantee that it will run precisely at this time since that depends on available resources. It will, however, ensure that the job runs *after* this time. If your resource requirements aren't too heavy, it will be pretty soon after. When I tried it, it started about 10 seconds afterwards but this will vary.

1.3 Filestore on Iceberg

Every user on the system has access to three different types of filestore. They differ in terms of the amount of space available, the speed of the underlying storage system, frequency of backup and the time that the data can be left there.

Here are the current details of filestore available to each user.

1.3.1 Home directory

All users have a home directory in the location `/home/username`. The filestore quota is **10 GB** per user.

Backup policy: `/home` has backup snapshots taken every 4 hours and we keep the 10 most recent. `/home` also has daily snapshots taken each night, and we keep 28 days worth, mirrored onto a separate storage system.

1.3.2 Data directory

Every user has access to a much larger data-storage area provided at the location `/data/username`.

The quota for this area is **100 GB** per user.

Backup policy: `/data` has snapshots taken every 4 hours and we keep the 10 most recent. `/data` also has daily snapshots taken each night, and we keep 7 days worth, but this is not mirrored.

1.3.3 Fastdata directory

All users also have access to a large fast-access data storage area under `/fastdata`.

In order to avoid interference from other users' files it is **vitally important** that you store your files in a directory created and named the same as your username. e.g.

```
mkdir /fastdata/yourusername
```

The fastdata area provides **260 Terabytes** of storage in total and takes advantage of the internal infiniband network for fast access to data.

Although `/fastdata` is available on all the worker nodes, only by accessing from the Intel-based nodes ensures that you can benefit from these speed improvements.

There are no quota controls on the `/fastdata` **area** but **files older than 3 months will be automatically deleted without warning.** We reserve the right to change this policy without warning in order to ensure efficient running of the service.

You can use the `lfs` command to find out which files under /fastdata are older than a certain number of days and hence approaching the time of deletion. For example, to find files 50 or more days old

```
lfs find -ctime +50 /fastdata/yourusername
```

1.3.4 Determining your current filestore allocation

To find out your current filestore quota allocation and usage type `quota`.

1.3.5 If you exceed your file storage allocation

As soon as the quota is exceeded your account becomes frozen. In order to avoid this situation it is strongly recommended that you

- Use the `quota` command to check your usage regularly.
- Copy files that do not need to be backed to the `/data/username` area, or remove them from iceberg completely.

1.3.6 Efficiency considerations - The /scratch areas

For jobs requiring a lot of Input and Output (I/O), it may sometimes be necessary to store copies of the data on the actual compute node on which your job is running. For this, you can create temporary areas of storage under the directory `/scratch`. **The** `/scratch` **area is local to each worker node** and is not visible to the other worker nodes or to the head-nodes. Therefore any data created by jobs should be transfered to either your `/data` or `/home` area before the job finishes if you wish to keep them.

The next best I/O performance that requires the minimum amount of work is achieved by keeping your data in the `/fastdata` area and running your jobs on the new intel nodes by specifying `-l arch=intel` in your job submission script.

These methods provide much faster access to data than the network attached storage on either `/home` or `/data` areas, but you must remember to copy important data back onto your `/home` area.

If you decide to use the `/scratch` area we recommend that under `/scratch` you create a directory with the same name as your username and work under that directory to avoid the possibility of clashing with other users.

Anything under the `/scratch` is deleted periodically when the worker-node is idle, whereas files on the `/fastdata` area will be deleted only when they are 3 months old.

1.3.7 Recovering snapshots

We take regular back-ups of your `/home` and `/data` directories and it is possible to directly access a limited subset of them.

There are 7 days worth of snapshots available in your `/home` and `/data` directories in a hidden directory called `.snapshot`. You need to explicitly `cd` into this directory to get at the files:

```
cd /home/YOURUSERNAME/.snapshot
```

The files are read-only. This allows you to attempt recover any files you might have accidentally deleted recently.

This does not apply for `/fastdata` for which we take no back-ups.

CHAPTER

TWO

ICEBERG SOFTWARE

These pages list the software availible on iceberg. If you notice an error or an omission, or wish to request new software please email the research computing team at research-it@sheffield.ac.uk.

2.1 Applications

2.1.1 CASTEP

CASTEP

Latest Version 8.0
URL http://www.castep.org/
location /usr/local/packages6/apps/intel/15/castep/8.0

Licensing

Only licensed users of CASTEP are entitled to use it and license details are available on CASTEP's website. Access to CASTEP on the system is controlled using a Unix group. That is, only members of the `castep` group can access and run the program. To be added to this group, you will need to contact the Iceberg email the team at research-it@sheffield.ac.uk and provide evidence of your eligibility to use CASTEP.

Interactive Usage

The serial version of CASTEP should be used for interactive usage. After connecting to iceberg (see *Connect to iceberg*), start an interactive session with the `qrsh` or `qsh` command. Make the serial version of CASTEP available using the command

```
module load apps/intel/15/castep/8.0-serial
```

The CASTEP executable is called `castep-serial` so if you execute

```
castep.serial
```

You should get the following

```
Usage:
castep <seedname>                : Run files <seedname>.cell [and <seedname>.param]
  "    [-d|--dryrun] <seedanme>  : Perform a dryrun calculation on files <seedname>.cell
  "    [-s|--search] <text>      : print list of keywords with <text> match in description
```

```
"    [-v|--version]            : print version information
"    [-h|--help] <keyword>     : describe specific keyword in <>.cell or <>.param
"         "      all           : print list of all keywords
"         "      basic         : print list of basic-level keywords
"         "      inter         : print list of intermediate-level keywords
"         "      expert        : print list of expert-level keywords
"         "      dummy         : print list of dummy keywords
```

If, instead, you get

```
-bash: castep.serial: command not found
```

It is probably because you are not a member of the `castep` group. See the section on Licensing above for details on how to be added to this group.

Interactive usage is fine for small CASTEP jobs such as the Silicon example given at http://www.castep.org/Tutorials/BasicsAndBonding

To run this example, you can do

```
# Get the files, decompress them and enter the directory containing them
wget http://www.castep.org/files/Si2.tgz
tar -xvzf ./Si2.tgz
cd Si2

#Run the CASTEP job in serial
castep.serial Si2

#Read the output using the more command
more Si2.castep
```

CASTEP has a built in help system. To get more information on using castep use

```
castep.serial -help
```

Alternatively you can search for help on a particular topic

```
castep.serial -help search keyword
```

or list all of the input parameters

```
castep.serial -help search all
```

Batch Submission - Parallel

The parallel version of CASTEP is called `castep.mpi`. To make the parallel environment available, use the module command

```
module load apps/intel/15/castep/8.0-parallel
```

As an example of a parallel submission, we will calculate the bandstructure of graphite following the tutorial at http://www.castep.org/Tutorials/BandStructureAndDOS

After connecting to iceberg (see *Connect to iceberg*), start an interactive session with the `qrsh` or `qsh` command. Download and decompress the example input files with the commands

```
wget http://www.castep.org/files/bandstructure.tgz
tar -xvzf ./bandstructure.tgz
```

Enter the directory containing the input files for graphite

```
cd bandstructure/graphite/
```

Create a file called `submit.sge` that contains the following

```
#!/bin/bash
#$ -pe openmpi-ib 4     # Run the calculation on 4 CPU cores
#$ -l rmem=4G           # Request 4 Gigabytes of real memory per core
#$ -l mem=4G            # Request 4 Gigabytes of virtual memory per core
module load apps/intel/15/castep/8.0-parallel

mpirun castep.mpi graphite
```

Submit it to the system with the command

```
qsub submit.sge
```

After the calculation has completed, get an overview of the calculation by looking at the file `graphite.castep`

```
more graphite.castep
```

Installation Notes

These are primarily for system administrators.

CASTEP Version 8

Serial (1 CPU core) and Parallel versions of CASTEP were compiled. Both versions were compiled with version 15.0.3 of the Intel Compiler Suite and the Intel MKL versions of BLAS and FFT were used. The parallel version made use of OpenMPI 1.8.8

The Serial version was compiled and installed with

```
module load compilers/intel/15.0.3
install_dir=/usr/local/packages6/apps/intel/15/castep/8.0

tar -xzf ./CASTEP-8.0.tar.gz
cd CASTEP-8.0

#Compile Serial version
make INSTALL_DIR=$install_dir  FFT=mkl MATHLIBS=mkl10
make INSTALL_DIR=$install_dir  FFT=mkl MATHLIBS=mkl10 install install-tools
```

The directory `CASTEP-8.0` was then deleted and the parallel version was installed with

```
#!/bin/bash
module load libs/intel/15/openmpi/1.8.8
#The above command also loads Intel Compilers 15.0.3
#It also places the MKL in LD_LIBRARY_PATH

install_dir=/usr/local/packages6/apps/intel/15/castep/8.0
mkdir -p $install_dir

tar -xzf ./CASTEP-8.0.tar.gz
cd CASTEP-8.0

#Compile parallel version
make COMMS_ARCH=mpi  FFT=mkl MATHLIBS=mkl10
mv ./obj/linux_x86_64_ifort15/castep.mpi $install_dir
```

Testing

The following script was submitted via `qsub`

```
 #!/bin/bash
#$ -pe openmpi-ib 4
module load libs/intel/15/openmpi/1.8.8

cd CASTEP-8.0
make check COMMS_ARCH=mpi  MAX_PROCS=4 PARALLEL="--total-processors=4 --processors=4"
```

All tests passed.

2.1.2 GATK

GATK

Version 3.4-46
URL https://www.broadinstitute.org/gatk/

The Genome Analysis Toolkit or GATK is a software package for analysis of high-throughput sequencing data, developed by the Data Science and Data Engineering group at the Broad Institute. The toolkit offers a wide variety of tools, with a primary focus on variant discovery and genotyping as well as strong emphasis on data quality assurance. Its robust architecture, powerful processing engine and high-performance computing features make it capable of taking on projects of any size.

Interactive Usage

After connecting to Iceberg (see *Connect to iceberg*), start an interactive sesssion with the *qsh* or *qrsh* command.

The latest version of GATK (currently 3.4-46) is made available with the command

```
module load apps/binapps/GATK
```

Alternatively, you can load a specific version with

```
module load apps/binapps/GATK/3.4-46
```

This module command also changes the environment to use Java 1.7 since this is required by GATK 3.4-46. An environment variable called *GATKHOME* is created by the module command that contains the path to the requested version of GATK.

Thus, you can run the program with the command

```
java -jar $GATKHOME/GenomeAnalysisTK.jar -h
```

Which will give a large amount of help, beginning with the version information

```
---------------------------------------------------------------------------------
The Genome Analysis Toolkit (GATK) v3.4-46-gbc02625, Compiled 2015/07/09 17:38:12
Copyright (c) 2010 The Broad Institute
For support and documentation go to http://www.broadinstitute.org/gatk
---------------------------------------------------------------------------------
```

Documentation

The GATK manual is available online https://www.broadinstitute.org/gatk/guide/

Installation notes

The entire install is just a .jar file. Put it in the install directory and you're done.

Modulefile

- The module file is on the system at */usr/local/modulefiles/apps/binapps/GATK/3.4-46*

Its contents are

```
##%Module1.0#####################################################################
##
## GATK 3.4-46 modulefile
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

#This version of GATK needs Java 1.7
module load apps/java/1.7

proc ModulesHelp { } {
        puts stderr "Makes GATK 3.4-46 available"
}


set version 3.4-46
set GATK_DIR /usr/local/packages6/apps/binapps/GATK/$version

module-whatis   "Makes GATK 3.4-46 available"

prepend-path GATKHOME $GATK_DIR
```

2.1.3 Abaqus

Abaqus

Versions 6.13,6.12 and 6.11
Support Level FULL
Dependancies Intel Compiler
URL http://www.3ds.com/products-services/simulia/products/abaqus/
Local URL https://www.shef.ac.uk/wrgrid/software/abaqus

Abaqus is a software suite for Finite Element Analysis (FEA) developed by Dassault Systèmes.

Interactive usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` command. Alternatively, if you require more memory, for example 16 gigabytes, use the command `qsh -l mem=16G`

The lastest version of Abaqus (currently version 6.13) is made available with the command

```
module load apps/abaqus
```

Alternatively, you can make a specific version available with one of the following commands

```
module load apps/abaqus/613
module load apps/abaqus/612
module load apps/abaqus/611
```

After that, simply type `abaqus` to get the command-line interface to abaqus or type `abaqus cae` to get the GUI interface.

Abaqus example problems

Abaqus contains a large number of example problems which can be used to become familiar with Abaqus on the system. These example problems are described in the Abaqus documentation, and can be obtained using the Abaqus fetch command. For example, after loading the Abaqus module enter the following at the command line to extract the input file for test problem s4d

```
abaqus fetch job=s4d
```

This will extract the input file s4d.inp, to run the computation defined by this input file replace input=myabaqusjob with input=s4d in the commands and scripts below.

Batch submission of a single core job

In this example, we will run the s4d.inp file on a single core using 8 Gigabytes of memory. After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qrsh` command.

Load version 6.13-3 of Abaqus and fetch the s4d example by running the following commands

```
module load apps/abaqus/613
abaqus fetch job=s4d
```

Now, you need to write a batch submission file. We assume you'll call this `my_job.sge`

```
#!/bin/bash
#$ -S /bin/bash
#$ -cwd
#$ -l rmem=8G
#$ -l mem=8G

module load apps/abaqus

abq6133 job=my_job input=s4d.inp scratch=/scratch memory="8gb" interactive
```

Submit the job with the command `qsub my_job.sge`

Important notes:

- We have requested 8 gigabytes of memory in the above job. The `memory="8gb"` switch tells abaqus to use 8 gigabytes. The `#$ -l rmem=8G` and `#$ -l mem=8G` tells the system to reserve 8 gigabytes of real and virtual memory resptively. It is important that these numbers match.

- Note the word `interactive` at the end of the abaqus command. Your job will not run without it.

Batch submission of a single core job with user subroutine

In this example, we will fetch a simulation from Abaqus' built in set of problems that makes use of user subroutines (UMATs) and run it in batch on a single core. After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qrsh` command.

Load version 6.13-3 of Abaqus and fetch the umatmst3 example by running the following commands

```
module load apps/abaqus/613
abaqus fetch job=umatmst3*
```

This will produce 2 files: The input file `umatmst3.inp` and the Fortran user subroutine `umatmst3.f`.

Now, you need to write a batch submission file. We assume you'll call this `my_user_job.sge`

```
#!/bin/bash
#$ -S /bin/bash
#$ -cwd
#$ -l rmem=8G
#$ -l mem=8G

module load apps/abaqus/613
module load compilers/intel/12.1.15

abq6133 job=my_user_job input=umatmst3.inp user=umatmst3.f scratch=/scratch memory="8gb" interactive
```

Submit the job with the command `qsub my_user_job.sge`

Important notes:

- In order to use user subroutimes, it is necessary to load the module for the intel compiler.
- The user-subroutine itself is passed to Abaqus with the switch `user=umatmst3.f`

2.1.4 Anaconda Python

Anaconda Python

Support Level gold
Dependancies None
URL https://store.continuum.io/cshop/anaconda/
Version 2.3 (Providing Python 2.7.10)

Anaconda Python is a Python distribution that contains Python itself and a large number of popular modules.

Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` or *qrsh* command.

The latest version of Anaconda can be loaded with

```
module load apps/binapps/anacondapython
```

Alternatively, you can load a specific version of Anaconda Python using

```
module load apps/binapps/anacondapython/2.3
```

Python can then be started with the `python` command:

```
$ python

Python 2.7.10 |Anaconda 2.3.0 (64-bit)| (default, May 28 2015, 17:02:03)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
>>>
```

Available Python Modules

Anaconda Python provides a large number of Python modules including numpy, scipy, matplotlib, scikit-learn, scikit-image, ipython and many more. In addition to these standard modules, we have installed others following user requests. To see which modules are available, run the following command in an interactive Python session

```
help('modules')
```

Installation Notes

These are primarily for administrators of the system. Anaconda Python is a binary install of Python.

```
mkdir -p /usr/local/packages6/apps/binapps/anacondapython/2.3
chmod +x ./Anaconda-2.3.0-Linux-x86_64.sh

#Run the installer
./Anaconda-2.3.0-Linux-x86_64.sh
```

When the installer asks for location, use

```
/usr/local/packages6/apps/binapps/anacondapython/2.3
```

Some modules were requested by users that were not included in the standard Anaconda installer. I installed these by doing

```
module load apps/binapps/anacondapython/2.3
conda install wxPython
```

Module file

location `/usr/local/modulefiles/apps/binapps/anacondapython/2.3`

```
#%Module10.2#####################################################################
#

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        global ver
```

```
        puts stderr "   Adds Anaconda $ver to your environment variables."
}

# Anaconda version (not in the user's environment)
set     ver     2.3

module-whatis   "sets the necessary Anaconda $ver paths"

set ANACONDADIR /usr/local/packages6/apps/binapps/anacondapython/
set ANACONDAHOME $ANACONDADIR/$ver

setenv ANACONDAHOME $ANACONDAHOME
prepend-path PATH $ANACONDAHOME/bin
```

2.1.5 Ansys

Ansys

Version 14 , 14.5 , 15
Support Level FULL
Dependencies If using the User Defined Functions (UDF) will also need the following: For Ansys Mechanical, Workbench, CFX and AutoDYN : INTEL 14.0 or above Compiler For Fluent : GCC 4.6.1 or above
URL http://www.ansys.com/en_uk
Local URL http://www.shef.ac.uk/cics/research/software/fluent

ANSYS suite of programs can be used to numerically simulate a large variety of structural and fluid dynamics problems found in many engineering, physics, medical, aeronotics and automative industry applications

Interactive usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` command. Alternatively, if you require more memory, for example 16 gigabytes, use the command `qsh -l rmem=16G`

To make the latest version of Ansys available, run the following module command

```
module load apps/ansys
```

Alternatively, you can make a specific version available with one of the following commands

```
module load apps/ansys/14
module load apps/ansys/14.5
module load apps/ansys/15.0
```

After loading the modules, you can issue the following commands to run an ansys product. The teaching versions mentioned below are installed for use during ANSYS and FLUENT teaching labs and will only allow models of upto 500,000 elements.

```
ansyswb  : to run Ansys workbench
ansys    : to run Ansys Mechanical outside the workbench
ansystext: to run line-mode version of ansys
Ansys and Ansystext : to run the teaching license version of the above two commands.
fluent   : to run Fluent outside the workbench
fluentext: to run the line-mode version of Fluent
```

```
Fluent and Fluentext : to run the teaching license version of the above two commands.
icemcfx or icem: to run icemcfd outside the workbench.
```

Running Batch fluent and ansys jobs

The easiest way of running batch ansys and fluent jobs is as follows:

```
module load {version_you_require} for example: module load apps/ansys/15.0
followed by  runfluent  or runansys
```

runfluent and runansys command submits a fluent journal or ansys input file into the batch system and can take a number of different parameters, according to your requirements.

runfluent command

Just typing runfluent will display information on how to use it.

Usage: runfluent [2d,2ddp,3d or 3ddp] fluent_journal_file -time hh:mm:ss [-mem=nn] [-rmem=nn] [-mail your_email_address] [-nq] [-parallel nprocs][optional_extra_fluent_params].

Where all but the first two parameters are optional.

```
First parameter [2d , 2ddp , etc  ] is the dimensionality of the problem.
Second parameter, fluent_journal_file, is the file containing the fluent commands.
Other 'optional' parameters are:
-time hh:mm:ss is the cpu time needed in hours:minutes:seconds
-mem=nn is the virtual memory needed (Default=8G). Example: -mem 12G (for 12 GBytes)
-rmem=nn is the real memory needed.(Default=2G). Example: -rmem 4G (for 4 GBytes)
-mail email_address. You will receive emails about the progress of your job.
Example:-mail J.Bloggs@sheffield.ac.uk
-nq is an optional parameter to submit without confirming
-parallel nprocs : Only needed for parallel jobs to specify the no.of processors.
-project project_name : The job will use a project allocation.
fluent_params : any parameter not recognised will be passed to fluent itself.
```

Example: runfluent 3d nozzle.jou -time 00:30:00 -mem=10G

Fluent journal files are essentially a sequence of Fluent Commands you would have entered by starting fluent in non-gui mode.

Here is an example journal file:

```
 /file/read-case test.cas
 /file/read-data test.dat
 /solve iter 200
 /file/write-data testv5b.dat
 yes
/exit
yes
```

Note that there can be no graphics output related commands in the journal file as the job will be run in batch mode. Please see fluent documents for further details of journal files and how to create them.

By using the -g parameter, you can startup an interactive fluent session in non-gui mode to experiment. For example- fluent 3d -g

runansys command

RUNANSYS COMMAND SUBMITS ANSYS JOBS TO THE SUN GRID ENGINE

Usage: runansys ansys_inp_file [-time hh:mm:ss][-mem=nn] [-rmem=nn] [-parallel n] [-project proj_name] [-mail email_address] [other qsub parameters]

Where; ansys_inp_file is a file containing a series of Ansys commands.

```
-time hh:mm:ss  is the cpu time needed in hours:minutes:seconds, if not specified 1 hour will be assu
-mem=nn       is the virtual memory requirement.
-rmem=nn      is the real memory requirement.
-parallel n   request an n-way parallel ansys job
-gpu          use GPU.  Note for GPU users: -mem= must be greater than 18G.
-project project_name : The job will use a project's allocation.
-mail your_email_address  : Job progress report is emailed to you.
```

As well as time and memory, any other valid qsub parameter can be specified.Particularly users of UPF functions will need to specify -v ANS_USER_PATH=the_working_directory

All parameters except the ansys_inp file are optional.

Output files created by Ansys take their names from the jobname specified by the user. You will be prompted for a jobname as well as any other startup parameter you wish to pass to Ansys Example: runansys test1.dat -time 00:30:00 -mem 8G -rmem=3G -mail j.bloggs@shef.ac.uk

2.1.6 bcbio

bcbio

Latest version Unknown
Dependancies gcc 5.2, R 3.2.1, Anaconda Python 2.3
URL http://bcbio-nextgen.readthedocs.org/en/latest/

A python toolkit providing best-practice pipelines for fully automated high throughput sequencing analysis. You write a high level configuration file specifying your inputs and analysis parameters. This input drives a parallel pipeline that handles distributed execution, idempotent processing restarts and safe transactional steps. The goal is to provide a shared community resource that handles the data processing component of sequencing analysis, providing researchers with more time to focus on the downstream biology.

Usage

Load the development version of bcbio with the command. The development version may be upgraded or modified at any time without warning.

```
module load apps/gcc/5.2/bcbio/devel
```

This correctly populates the PATH, LD_LIBRARY_PATH and PERL5LIB environment variables for bcbio usage.

Example batch submission

TODO

Integration with SGE

TODO

Installation Notes

These are primarily for system administrators.

Development version

The development version was installed using gcc 5.2, R 3.2.1 and Anaconda Python 2.3.

- install_bcbio_devel.sge This is a SGE submit script. The long running time of the installer made it better-suite to being run as a batch job.
- bcbio-devel modulefile located on the system at `/usr/local/modulefiles/apps/gcc/5.2/bcbio/devel`

The first install attempt failed with the error

```
To debug, please try re-running the install command with verbose output:
export CC=${CC:-`which gcc`} && export CXX=${CXX:-`which g++`} && export SHELL=${SHELL:-/bin/bash} &&
Traceback (most recent call last):
  File "bcbio_nextgen_install.py", line 276, in <module>
    main(parser.parse_args(), sys.argv[1:])
  File "bcbio_nextgen_install.py", line 46, in main
    subprocess.check_call([bcbio["bcbio_nextgen.py"], "upgrade"] + _clean_args(sys_argv, args, bcbio)
  File "/usr/local/packages6/apps/binapps/anacondapython/2.3/lib/python2.7/subprocess.py", line 540,
    raise CalledProcessError(retcode, cmd)
subprocess.CalledProcessError: Command '['/usr/local/packages6/apps/gcc/5.2/bcbio/devel/anaconda/bin/
```

I manually ran the command

```
export CC=${CC:-`which gcc`} && export CXX=${CXX:-`which g++`} && export SHELL=${SHELL:-/bin/bash} &&
```

and it completed successfully. I then resubmitted the submit script which eventually completed successfully. It took several hours! At this point, I created the module file.

Bcbio was upgraded to the development version with the following interactive commands

```
module load apps/gcc/5.2/bcbio/devel
bcbio_nextgen.py upgrade -u development
```

The GATK .jar file was obtained from https://www.broadinstitute.org/gatk/download/ and installed to bcbio by running the following commands interactively

```
module load apps/gcc/5.2/bcbio/devel
bcbio_nextgen.py upgrade --tools --toolplus gatk=./cooper/GenomeAnalysisTK.jar
```

Testing

The following test script was submitted to the system. All tests passed. The output is at `/usr/local/packages6/apps/gcc/5.2/bcbio/devel/tests/tests_28_8_2015`

```
#!/bin/bash
#$ -pe openmp 12
#$ -l mem=4G  #Per Core!
#$ -l rmem=4G #Per Core!

bcbio_dir=/data/fe1mpc/bcbio_install/tools
```

```
module add apps/gcc/5.2/bcbio/devel

git clone https://github.com/chapmanb/bcbio-nextgen.git
cd bcbio-nextgen/tests
./run_tests.sh devel
./run_tests.sh rnaseq
```

2.1.7 bcl2fastq

bcl2fastq

Versions 1.8.4
Support Level Bronze
URL http://support.illumina.com/downloads/bcl2fastq_conversion_software_184.html

Illumina sequencing instruments generate per-cycle BCL basecall files as primary sequencing output, but many downstream analysis applications use per-read FASTQ files as input. bcl2fastq combines these per-cycle BCL files from a run and translates them into FASTQ files. bcl2fastq can begin bcl conversion as soon as the first read has been completely sequenced.

Usage

To make `bcl2fastq` available, use the following `module` command in your submission scripts

```
module load apps/bcl2fastq/1.8.4
```

Installation Notes

These notes are primarily for system administrators.

Compilation was done using gcc 4.4.7. I tried it with gcc 4.8 but ended up with a lot of errors. The package is also dependent on Perl. Perl 5.10.1 was used which was the system Perl installed at the time. The RPM Perl-XML-Simple also needed installing.

```
export TMP=/tmp
export SOURCE=${TMP}/bcl2fastq
export BUILD=${TMP}/bcl2fastq-1.8.4-build
mkdir -p /usr/local/packages6/apps/gcc/4.4.7/bcl2fastq/1.8.4
export INSTALL=/usr/local/packages6/apps/gcc/4.4.7/bcl2fastq/1.8.4

mv bcl2fastq-1.8.4.tar.bz2 ${TMP}
cd ${TMP}
tar xjf bcl2fastq-1.8.4.tar.bz2

mkdir ${BUILD}
cd ${BUILD}
${SOURCE}/src/configure --prefix=${INSTALL}

make
make install
```

2.1.8 Bowtie2

bowtie2

Versions 2.2.26
URL http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

Interactive Usage

After connecting to Iceberg (see *Connect to iceberg*), start an interactive sesssion with the *qsh* or *qrsh* command.

The latest version of bowtie2 (currently 2.2.26) is made available with the command

```
module load apps/gcc/5.2/bowtie2
```

Alternatively, you can load a specific version with

```
module load apps/gcc/5.2/bowtie2/2.2.6
```

This command makes the bowtie2 executables available to your session by adding the install directory to your PATH variable. This allows you to simply do something like the following

```
bowtie2 --version
```

which gives results that looks something like

```
/usr/local/packages6/apps/gcc/5.2/bowtie2/2.2.6/bowtie2-align-s version 2.2.6
64-bit
Built on node063
Fri Oct 23 08:40:38 BST 2015
Compiler: gcc version 5.2.0 (GCC)
Options: -O3 -m64 -msse2  -funroll-loops -g3 -DPOPCNT_CAPABILITY
Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8, 8, 8, 8, 8}
```

Installation notes

bowtie2 2.2.6 was installed using gcc 5.2

```
#Build
module load compilers/gcc/5.2
unzip bowtie2-2.2.6-source.zip
cd bowtie2-2.2.6
make

#Install
cd ..
mkdir -p /usr/local/packages6/apps/gcc/5.2/bowtie2
mv ./bowtie2-2.2.6 /usr/local/packages6/apps/gcc/5.2/bowtie2/2.2.6
```

Testing

No test suite was found.

Modulefile

- The module file is on the system at */usr/local/modulefiles/apps/gcc/5.2/bowtie2/2.2.6*

The contents of the module file is

```
#%Module1.0#####################################################################
##
## bowtie2 2.2.6 modulefile
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

module load compilers/gcc/5.2


proc ModulesHelp { } {
        puts stderr "Makes bowtie 2.2.6 available"
}

set version 2.2.6
set BOWTIE2_DIR /usr/local/packages6/apps/gcc/5.2/bowtie2/$version

module-whatis   "Makes bowtie2 v2.2.6 available"

prepend-path PATH $BOWTIE2_DIR
```

2.1.9 bwa

bwa

Versions 0.7.12
URL http://bio-bwa.sourceforge.net/

BWA (Burrows-Wheeler Aligner) is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate. BWA-MEM also has better performance than BWA-backtrack for 70-100bp Illumina reads.

Interactive Usage

After connecting to Iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` command.

The lastest version of bwa (currently 0.7.12) is made available with the command

```
module load apps/gcc/5.2/bwa
```

Alternatively, you can load a specific version with

```
module load apps/gcc/5.2/bwa/0.7.12
```

This command makes the bwa binary available to your session.

Documentation

Once you have made bwa available to the system using the *module* command above, you can read the man pages by typing

```
man bwa
```

Installation notes

bwa was installed using gcc 5.2

```
module load compilers/gcc/5.2

#build
module load compilers/gcc/5.2
tar -xvjf ./bwa-0.7.12.tar.bz2
cd bwa-0.7.12
make

#Sort out manfile
mkdir -p share/man/man1
mv bwa.1 ./share/man/man1/

#Install
mkdir -p /usr/local/packages6/apps/gcc/5.2/bwa/
cd ..
mv bwa-0.7.12 /usr/local/packages6/apps/gcc/5.2/bwa/0.7.12/
```

Testing

No test suite was found.

Modulefile

- The module file is on the system at */usr/local/modulefiles/apps/gcc/5.2/bwa/0.7.12*
- The module file is on github.

2.1.10 Code Saturne 4.0

Code Saturne

Version 4.0
Support Level Bronze
URL http://code-saturne.org/cms/
Documentation http://code-saturne.org/cms/documentation
Location /usr/local/packages6/apps/gcc/4.4.7/code_saturne/4.0

Code_Saturne solves the Navier-Stokes equations for 2D, 2D-axisymmetric and 3D flows, steady or unsteady, laminar or turbulent, incompressible or weakly dilatable, isothermal or not, with scalars transport if required.

Usage

To make code saturne available, run the following module command after starting a `qsh` session.

```
module load apps/code_saturne/4.0.0
```

Troubleshooting

If you run Code Saturne jobs from `/fastdata` they will fail since the underlying filesystem used by `/fastdata` does not support posix locks. If you need more space than is available in your home directory, use `/data` instead. If you use `/fastdata` with Code Saturne you may get an error message similar to the one below

```
File locking failed in ADIOI_Set_lock(fd 14,cmd F_SETLKW/7,type F_WRLCK/1,whence 0) with return value
- If the file system is NFS, you need to use NFS version 3, ensure that the lockd daemon is running
- If the file system is LUSTRE, ensure that the directory is mounted with the 'flock' option.
ADIOI_Set_lock:: Function not implemented
ADIOI_Set_lock:offset 0, length 8
--------------------------------------------------------------------------
MPI_ABORT was invoked on rank 0 in communicator MPI_COMM_WORLD
with errorcode 1.
NOTE: invoking MPI_ABORT causes Open MPI to kill all MPI processes.
You may or may not see output from other processes, depending on
exactly when Open MPI kills them.
--------------------------------------------------------------------------
 solver script exited with status 1.
Error running the calculation.
Check Code_Saturne log (listing) and error* files for details.
```

Installation Notes

These notes are primarily for system administrators.

Installation notes for the version referenced by module `module load apps/code_saturne/4.0.0`:

Pre-requisites: This version of Code Saturne was built with the following:-

- gcc 4.4.7
- *cgns* 3.2.1
- *MED* 3.0.8
- OpenMPI 1.8.3
- *HDF5 (gcc build)* 1.8.14

```
module load libs/gcc/4.4.7/cgns/3.2.1

tar -xvzf code_saturne-4.0.0.tar.gz
mkdir code_saturn_build
cd code_saturn_build/
./../code_saturne-4.0.0/configure --prefix=/usr/local/packages6/apps/gcc/4.4.7/code_saturne/4.0 --wit
```

This gave the following configuration

```
Configuration options:
 use debugging code: no
 MPI (Message Passing Interface) support: yes
 OpenMP support: no

The package has been configured. Type:
 make
 make install

To generate and install the PLE package


Configuration options:
 use debugging code: no
 use malloc hooks: no
 use graphical user interface: yes
 use long integers: yes
 Zlib (gzipped file) support: yes
 MPI (Message Passing Interface) support: yes
   MPI I/O support: yes
   MPI2 one-sided communication support: yes
 OpenMP support: no
 BLAS (Basic Linear Algebra Subprograms) support: no
 Libxml2 (XML Reader) support: yes
 ParMETIS (Parallel Graph Partitioning) support: no
 METIS (Graph Partitioning) support: no
 PT-SCOTCH (Parallel Graph Partitioning) support: no
 SCOTCH (Graph Partitioning) support: no
 CCM support: no
 HDF (Hierarchical Data Format) support: yes
 CGNS (CFD General Notation System) support: yes
 MED (Model for Exchange of Data) support: yes
   MED MPI I/O support: yes
 MEDCoupling support: no
 Catalyst (ParaView co-processing) support: no
 EOS support: no
 freesteam support: no
 SALOME GUI support: yes
 SALOME Kernel support: yes
 Dynamic loader support (for YACS): dlopen
```

I then did

```
make
make install
```

Post Install Steps

To make Code Saturne aware of the SGE system:

- Created /usr/local/packages6/apps/gcc/4.4.7/code_saturne/4.0/etc/code_saturne.cfg: See code_saturne.cfg 4.0
- Modified /usr/local/packages6/apps/gcc/4.4.7/code_saturne/4.0/share/code_saturne/batch/ba See: batch.SGE 4.0

Testing

This module has not been yet been properly tested and so should be considered experimental.

Several user's jobs up to 8 cores have been submitted and ran to completion.

Module File

Module File Location: /usr/local/modulefiles/apps/code_saturne/4.0.0

```
##%Module1.0#####################################################################
##
## code_saturne 4.0 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
      global code-saturneversion

      puts stderr "   Adds 'code_saturn-$codesaturneversion' to your PATH environment variable and ne
}

set     codesaturneversion 4.0.
module load mpi/gcc/openmpi/1.8.3

module-whatis   "loads the necessary 'code_saturne-$codesaturneversion' library paths"

set cspath /usr/local/packages6/apps/gcc/4.4.7/code_saturne/4.0
prepend-path MANPATH $cspath/share/man
prepend-path PATH $cspath/bin
```

2.1.11 Cufflinks

Cufflinks

Version 2.2.1
URL http://cole-trapnell-lab.github.io/cufflinks

Cufflinks assembles transcripts, estimates their abundances, and tests for differential expression and regulation in RNA-Seq samples. It accepts aligned RNA-Seq reads and assembles the alignments into a parsimonious set of transcripts. Cufflinks then estimates the relative abundances of these transcripts based on how many reads support each one, taking into account biases in library preparation protocols.

Interactive Usage

After connecting to Iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` or *qrsh* command.

The latest version of Cufflinks (currently 2.2.1) is made available with the command

```
module load apps/binapps/cufflinks
```

Alternatively, you can load a specific version with

```
module load apps/binapps/cufflinks/2.2.1
```

This command makes the cufflinks binary directory available to your session by adding it to the PATH environment variable.

Documentation

The Cufflinks manual is available online at http://cole-trapnell-lab.github.io/cufflinks/manual/

Installation notes

A binary install was used

```
tar -xvzf cufflinks-2.2.1.Linux_x86_64.tar.gz
mkdir -p /usr/local/packages6/apps/binapps/cufflinks
mv cufflinks-2.2.1.Linux_x86_64 /usr/local/packages6/apps/binapps/cufflinks/2.2.1
```

Modulefile

- The module file is on the system at */usr/local/modulefiles/apps/binapps/cufflinks/2.2.1*

Its contents is

```
## cufflinks 2.2.1 modulefile
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        puts stderr "Makes cufflinks 2.2.1 available"
}

set version 2.2.1
set CUFF_DIR /usr/local/packages6/apps/binapps/cufflinks/$version

module-whatis   "Makes cufflinks 2.2.1 available"

prepend-path PATH $CUFF_DIR
```

2.1.12 GROMACS

GROMACS

Latest version 5.1
Dependancies mpi/intel/openmpi/1.10.0
URL http://www.gromacs.org/

Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` or `qrsh` command. To make GROMACS available in this session, run one of the following command:

```
source /usr/local/packages6/apps/intel/15/gromacs/5.1/bin/GMXRC
```

Installation notes

Version 5.1

Compilation Choices:

- Use latest intel compilers
- -DGMX_MPI=on
- -DCMAKE_PREFIX_PATH=/usr/local/packages6/apps/intel/15/gromcas
- -DGMX_FFT_LIBRARY="fftw3"
- -DGMX_BUILD_OWN_FFTW=ON

The script used to build gromacs can be found here.

2.1.13 IDL

IDL

Version 8.4
Support Level extras
Dependancies java
URL http://www.exelisvis.co.uk/ProductsServices/IDL.aspx
Documentation http://www.exelisvis.com/docs/using_idl_home.html

IDL is a data analysis language that first appeared in 1977.

Usage

If you wish to use the IDLDE then you may need to request more memory for the interactive session using something like *qsh -l mem=8G*.

IDL can be activated using the module file:

```
module load apps/idl/8.4
```

then run using *idl* or *idlde* for the interactive development environment.

Installation notes

Extract the supplied linux x86-64 tar file, run the install shell script and point it at the install directory.

2.1.14 JAGS

JAGS

Latest version 4.8.2
Dependancies compilers/gcc/4.8.2
URL http://mcmc-jags.sourceforge.net/

JAGS is Just Another Gibbs Sampler. It is a program for analysis of Bayesian hierarchical models using Markov Chain Monte Carlo (MCMC) simulation not wholly unlike BUGS. JAGS was written with three aims in mind:

- To have a cross-platform engine for the BUGS language
- To be extensible, allowing users to write their own functions, distributions and samplers.
- To be a plaftorm for experimentation with ideas in Bayesian modelling

Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` or `qrsh` command. To make JAGS available in this session, run one of the following module command

```
module load apps/gcc/4.8.3/JAGS/3.1
module load apps/gcc/4.8.2/JAGS/3.4
```

You can now run the `jags` command

```
jags
```

```
Welcome to JAGS 3.4.0 on Fri Jun 12 13:13:31 2015
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
.
```

The rjags and runjags interfaces in R

rjags and runjags are CRAN packages that provide an R interface to jags. They are not installed in R by default.

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qrsh` command. Run the following module commands

```
module load compilers/gcc/4.8.2
module load apps/gcc/4.8.2/JAGS/3.4
module load apps/R/3.2
```

Launch R by typing `R` and pressing return. Within R, execute the following commands

```
install.packages('rjags')
install.packages('runjags')
```

and follow the on-screen inctructions. Answer `y` to any questions about the creation of a personal library should they be asked.

The packages will be stored in a directory called *R* within your home directory.

You should only need to run the `install.packages` commands once. When you log into the system in future, you will only need to run the `module` commands above to make JAGS available to the system.

You load the rjags packages the same as you would any other R package

```
library('rjags')
library('runjags')
```

If you received an error message such as

```
Error : .onLoad failed in loadNamespace() for 'rjags', details:
  call: dyn.load(file, DLLpath = DLLpath, ...)
  error: unable to load shared object '/home/fe1mpc/R/x86_64-unknown-linux-gnu-library/3.2/rjags/libs
  libjags.so.3: cannot open shared object file: No such file or directory
Error: package or namespace load failed for 'rjags'
```

the most likely cause is that you forget to load the necessary modules before starting R.

Installation notes

- Version 3.4

JAGS 3.4 was built with gcc 4.8.2

```
module load compilers/gcc/4.8.2
tar -xvzf ./JAGS-3.4.0.tar.gz
cd JAGS-3.4.0
mkdir -p /usr/local/packages6/apps/gcc/4.8.2/JAGS/3.4
./configure --prefix=/usr/local/packages6/apps/gcc/4.8.2/JAGS/3.4
make
make install
```

- Version 3.1

JAGS 3.1 was built with gcc 4.8.2

```
module load compilers/gcc/4.8.2
tar -xvzf ./JAGS-3.1.0.tar.gz
cd JAGS-3.1.0
mkdir -p /usr/local/packages6/apps/gcc/4.8.2/JAGS/3.1
./configure --prefix=/usr/local/packages6/apps/gcc/4.8.2/JAGS/3.1
make
make install
```

2.1.15 Maple

Maple

Versions 2015
Support Level FULL
Dependancies None
URL http://www.maplesoft.com/products/maple/

Scientific Computing and Visualisation

Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` command.

The lastest version of Maple (currently 2015) is made available with the command

```
module load apps/binapps/maple
```

Alternatively, you can load a specific version with

```
module load apps/binapps/maple/2015
```

You can then run the graphical version of Maple by entering `xmaple` or the command line version by entering `maple`.

Installation notes

These are primarily for administrators of the system.

The module file is at `/usr/local/modulefiles/apps/binapps/maple/2015`

```
#%Module10.2#####################################################################
#

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        global ver

        puts stderr " Makes Maple $ver available to the system."
}

# Maple version (not in the user's environment)
set     ver     2015

module-whatis   "sets the necessary Maple $ver paths"

prepend-path PATH /usr/local/packages6/maple/bin/
```

2.1.16 Mathematica

Wolfram Mathematica

Dependancies None
URL http://www.wolfram.com/mathematica/
Version 10.2

Mathematica is a technical computing environment and programming language with strong symbolic and numerical abilities.

Single Core Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive session with `qsh`.

The latest version of Mathematica can be loaded with

```
module load apps/binapps/mathematica
```

Alternatively, you can load a specific version of Mathematica using

```
module load apps/binapps/mathematica/10.2
```

Mathematica can then be started with the `mathematica` command

```
mathematica
```

Multicore Interactive Usage

Mathematica has extensive parallel functionality. To use it, you should request a parallel interactive session. For example, to request 4 cores

```
qsh -pe openmp 4
```

Load and launch Mathematica

```
module load apps/binapps/mathematica
mathematica
```

In Mathematica, let's time how long it takes to calculate the first 20 million primes on 1 CPU core

```
AbsoluteTiming[primelist = Table[Prime[k], {k, 1, 20000000}];]
```

When I tried this, I got 78 seconds. Your results may vary greatly. Now, let's launch 4 ParallelKernels and redo the calculation in parallel

```
 LaunchKernels[4]
 AbsoluteTiming[primelist =
ParallelTable[Prime[k], {k, 1, 20000000},
 Method -> "CoarsestGrained"];]
```

When I tried this, I got 29 seconds – around 2.7 times faster. This illustrates a couple of points:-

- You should always request the same number of kernels as you requested in your `qsh` command (in this case, 4). If you request more, you will damage performance for yourself and other users of the system.
- N kernels doesn't always translate to N times faster.

Installation notes

These are primarily for administrators of the system

```
mkdir -p /usr/local/packages6/apps/binapps/mathematica/10.2
chmod +x ./Mathematica_10.2.0_LINUX.sh
./Mathematica_10.2.0_LINUX.sh
```

The installer is interactive. Here's the session output

```
-------------------------------------------------------------------------------------------------------------
                                                                                                Wolfram Mathemati
-------------------------------------------------------------------------------------------------------------

Copyright (c) 1988-2015 Wolfram Research, Inc. All rights reserved.

WARNING: Wolfram Mathematica is protected by copyright law and international treaties. Unauthorized r
prosecuted to the maximum extent possible under law.

Enter the installation directory, or press ENTER to select /usr/local/Wolfram/Mathematica/10.2:
>

Error: Cannot create directory /usr/local/Wolfram/Mathematica/10.2.

You may need to be logged in as root to continue with this installation.

Enter the installation directory, or press ENTER to select /usr/local/Wolfram/Mathematica/10.2:
> /usr/local/packages6/apps/binapps/mathematica/10.2

Now installing...

[*********************************************************************************************

Type the directory path in which the Wolfram Mathematica script(s) will be created, or press ENTER to
> /usr/local/packages6/apps/binapps/mathematica/10.2/scripts

Create directory (y/n)?
> y


WARNING: No Avahi Daemon was detected so some Kernel Discovery features will not be available. You ca

For Red Hat based distributions, try running (as root):

yum install avahi

Installation complete.
```

Remove the `playerpass` file

```
rm /usr/local/packages6/apps/binapps/mathematica/10.2/Configuration/Licensing/playerpass
```

Install the University network `mathpass` file at `/usr/local/packages6/apps/binapps/mathematica/10.2/Configu`

2.1.17 MATLAB

MATLAB

Versions 2013a , 2013b , 2014a, 2015a
Support Level FULL
Dependancies None
URL http://uk.mathworks.com/products/matlab
Local URL http://www.shef.ac.uk/wrgrid/software/matlab
Documentation http://uk.mathworks.com/help/matlab

Scientific Computing and Visualisation

Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` command.

The lastest version of MATLAB (currently 2015a) is made available with the command

```
module load apps/matlab
```

Alternatively, you can load a specific version with one of of the following commands

```
module load apps/matlab/2013a
module load apps/matlab/2013b
module load apps/matlab/2014a
module load apps/matlab/2015a
```

You can then run MATLAB by entering `matlab`

Serial (one CPU) Batch usage

Here, we assume that you wish to run the program `hello.m` on the system.

First, you need to write a batch submission file. We assume you'll call this `my_job.sge`

```
#!/bin/bash
#$ -l rmem=4G                      # Request  4 Gigabytes of real memory
#$ -l mem=16G                      # Request 16 Gigabytes of virtual memory
$ -cwd                             # Run job from current directory
module load apps/matlab            # Make latest version of MATLAB available

matlab -nodesktop -r 'hello'
```

Ensuring that `hello.m` and `myjob.sge` are both in your current working directory, submit your job to the batch system

```
qsub my_job.sge
```

Some notes about this example:

- We are running the script `hello.m` but we drop the *.m* in the call to MATLAB. That is, we do `-r 'hello'` rather than `-r hello.m`.
- All of the `module` commands introducted in the Interactive usage section will also work in batch mode. This allows you to select a specific version of MATLAB if you wish.

Parallel MATLAB on iceberg

Currently we recommend the 2015a version of MATLAB for parallel work.

The default cluster configuration named 'local' provides parallel working environment by using the CPUs of the worker-node that is running the current MATLAB session. Each iceberg worker-node can run multiple users' jobs simultaneously. Therefore depending on who else is using that node at the time, parallel MATLAB jobs can create contentions between jobs and slow them considerably. It is therefore advisable to start parallel MATLAB jobs that will use the 'local' profile from a parallel SGE job. For example, to use the local profile with 5 workers, do the following;

Start a parallel OPENMP job with 6 workers.

    Qsh -pe openmp 6

Run MATLAB in that session and select 5 workers.

```
MATLAB
parpool ('local' , 5 )
```

The above example will use 5 MATLAB workers on a single iceberg-node to run a parallel task.

To take advantage of the multiple iceberg-nodes, you will need to make use of a parallel cluster profile named 'sge'. This can be done by issuing a locally provided MATLAB command named `iceberg` that imports the parallel cluster profile named `sge` that can take advantage of the SGE scheduler to run larger parallel jobs.

When using the 'sge' profile, MATLAB will be able to submit multiple MATLAB jobs the the SGE scheduler from within MATLAB itself. However, each job will have the default resource requirements unless the following trick is deployed. For example, during your MATLAB session type:

```
global sge_params
sge_params='-l mem=16G -l h_rt=36:00:00'
```

to make sure that all the MATLAB batch jobs will use upto 16GBytes of memory and will not be killed unless they exceed 36 hours of run time.

Installation notes

These notes are primarily for system administrators.

Requires the floating license server licserv4.shef.ac.uk to serve the licenses for the version of MATLAB to be installed (or higher versions) . An install script and associated files are downloadable from Mathworks site along with all the required toolbox specific installation files.

2.1.18 OpenSim

OpenSim

Support Level bronze
Dependancies None
URL https://simtk.org/home/opensim
Version 3.3

OpenSim is a freely available, user extensible software system that lets users develop models of musculoskeletal structures and create dynamic simulations of movement.

Usage

The latest version of OpenSim can be loaded with

```
module load apps/gcc/4.8.2/opensim
```

Installation Notes

These are primarily for administrators of the system.

Built using:

```
cmake /home/cs1sjm/Downloads/OpenSim33-source/
-DCMAKE_INSTALL_PREFIX=/usr/local/packages6/apps/gcc/4.8.2/opensim/3.3/
-DSIMBODY_HOME=/usr/local/packages6/libs/gcc/4.8.2/simbody/3.5.3/
-DOPENSIM_STANDARD_11=ON

make -j 8

make install
```

2.1.19 Paraview

Paraview

Version 4.3
Support Level extras
Dependancies openmpi (1.8)
URL http://paraview.org/
Documentation http://www.paraview.org/documentation/

Paraview is a parallel visualisation tool.

Using Paraview on iceberg

This guide describes how to use Paraview from iceberg. Paraview is a parallel visualisation tool designed for use on clusters like iceberg. Because iceberg is designed primarily as a headless computational cluster paraview has not been installed on iceberg in such a way that you load the GUI remotely on iceberg [1]. Paraview therefore runs on iceberg in client + server mode, the server running on iceberg, and the client on your local machine.

Configuring the Client

To use Paraview on iceberg, first download and install Paraview 4.3 [2]. Once you have installed Paraview locally (the client) you need to configure the connection to iceberg. In Paraview go to File > Connect, then click Add Server, name the connection iceberg, and select 'Client / Server (reverse connection)' for the Server Type, the port should retain the default value of 11111. Then click configure, on the next screen leave the connection as manual and click save. Once you are back at the original connect menu, click connect to start listening for the connection from iceberg.

Starting the Server

Once you have configured the local paraview client, login to iceberg from the client machine via ssh [3] and run *qsub-paraview*. This will submit a job to the scheduler que for 16 processes with 4GB or RAM each. This is designed to be used for large visualisation tasks, smaller jobs can be requested by specifying standard qsub commands to *qsub-paraview* i.e. *qsub-paraview -pe openmpi-ib 1* will only request one process.

Assuming you still have the client listening for connections, once the paraview job starts in the que it should connect to your client and you should be able to start accessing data stored on iceberg and rendering images.

[1] It is not possible to install the latest version of the paraview GUI on iceberg due to the Qt version shipped with Scientific Liunx 5.
[2] The client and server versions have to match.
[3] Connecting to Paraview via the automatic method descibed here is not supported on the MyApps portal.

A Note on Performance

When you run Paraview locally it will use the graphics hardware of your local machine for rendering. This is using hardware in your computer to create and display these images. On iceberg there is no such hardware, it is simulated via software. Therefore for small datasets you will probably find you are paying a performance penalty for using Paraview on iceberg.

The advantage however, is that the renderer is on the same cluster as your data, so no data transfer is needed, and you have access to very large amounts of memory for visualising very large datasets.

Manually Starting the Server

The *qsub-paraview* command is a wrapper that automatically detects the client IP address from the SSH connection and submits the job. It is possible to customise this behavior by copying and modifying this script. This for instance would allow you to start the paraview server via MyApps or from a different computer to the one with the client installed. The script used by *qsub-paraview* also serves as a good example script and can be copied into your home directory by running *cp /usr/local/bin/pvserver_submit.sh ~/*. This script can then be qsubmitted as normal by *qsub*. The client IP address can be added manually by replacing *echo $SSH_CLIENT | awk '{ print $1}'* with the IP address. More information on Paraview client/server can be found Here.

Installation

Custom build scripts are availible in */usr/local/extras/paraview/build_scripts* which can be used to recompile.

2.1.20 Plink

Plink

Versions 1.9 beta 3
Support Level Bronze
Dependancies None
URL https://www.cog-genomics.org/plink2

PLINK is a free, open-source whole genome association analysis toolset, designed to perform a range of basic, large-scale analyses in a computationally efficient manner.

Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh or `qrsh command.

The latest version of Plink is made available with the command

```
module load apps/binapps/plink
```

Alternatively, you can load a specific version with

```
module load apps/binapps/plink/1.9
```

You can now execute the `plink` command on the command line.

Installation notes

These are primarily for administrators of the system.

The binary version of Plink was installed

```
mkdir plink_build
cd plink_build
unzip plink_linux_x86_64.zip
rm plink_linux_x86_64.zip
mkdir -p /usr/local/packages6/apps/binapps/plink/1.9
mv * /usr/local/packages6/apps/binapps/plink/1.9
```

The module file is at `/usr/local/modulefiles/apps/binapps/plink/1.9`

```
#%Module10.2#####################################################################
#

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        global ver

        puts stderr " Makes Plink $ver available to the system."
}

# Plink version (not in the user's environment)
set     ver     1.9

module-whatis   "sets the necessary Plink $ver paths"

prepend-path PATH /usr/local/packages6/apps/binapps/plink/$ver
```

2.1.21 Python

Python

Support Level gold
Dependancies None
URL https://python.org
Version multiple

There are multiple ways of accessing Python on iceberg, this section gives an overview, and details on the recommended method.

Avalible Python Installations

Anaconda Python

Unless you need to install custom modules beyond those included in Anaconda on Python 2 this is the recommended method. See *Anaconda Python* for more details.

conda Python

This is the primary method of using Python on iceberg if you need to install custom packages, the rest of this document will be talking about this system.

System Python

The system Python 2.6 is available by default as it is installed with the cluster OS Scientific Python 5, it is not recommended for general use.

Using conda Python

After connecting to iceberg (see *Connect to iceberg*), start an interactive session with the `qsh` or `qrsh` command.

Conda Python can be loaded with:

```
module load apps/python-conda
```

The `root` conda environment (the default) provides Python 3 and no extra modules, it is automatically updated, and not recommended for general use, just as a base for your own environments. There is also a `python2` environment, which is the same by with a Python 2 installation.

Using conda Environments

Once the system conda module is loaded you have to load or create the desired conda environments. For the documentation on conda environments see here.

You can load a conda environment with:

```
source activate python2
```

and unload one with:

```
source deactivate
```

which will return you to the `root` environment.

It is possible to list all the available environments with:

```
conda env list
```

Provided system-wide are a set of anaconda environments, these will be installed with the anaconda version number in the environment name, and never modified. They will therefore provide a static base for derivative environments or for using directly.

Creating an Environment

Every user can create their own environments, and packages shared with the system-wide environments will not be reinstalled or copied to your file store, they will be `symlinked`, this reduces the space you need in your `/home` directory to install many different Python environments.

To create a clean environment with just Python 2 and numpy you can run:

```
conda create -n mynumpy python=2.7 numpy
```

This will download the latest release of Python 2.7 and numpy, and create an environment named `mynumpy`.

If you wish to modify an existing environment, such as one of the anaconda installations, you can `clone` that environment:

```
conda create --clone anaconda3-2.3.0 -n myexperiment
```

This will create an environment called `myexperiment` which has all the anaconda 2.3.0 packages installed with Python 3.

Installing Packages Inside an Environment

Once you have created your own environment you can install additional packages or different versions of packages into it. There are two methods for doing this, `conda` and `pip`, if a package is available through conda it is strongly recommended that you use conda to install packages. You can search for packages using conda:

```
conda search pandas
```

then install the package using:

```
conda install pandas
```

if you are not in your environment you will get a permission denied error when trying to install packages, if this happens, create or activate an environment you own.

If a package is not available through conda you can search for and install it using pip:

```
pip search colormath
```

```
pip install colormath
```

Installation Notes

These are primarily for administrators of the system.

The conda package manager is installed in `/usr/share/packages6/conda`, it was installed using the miniconda installer.

The two "root" environments `root` and `python2` can be updated using the update script located in `/usr/local/packages6/conda/_envronments/conda-autoupdate.sh`. This should be run regularly to keep this base environments upto date with Python, and more importantly with the conda package manager itself.

Installing a New Version of Anaconda

Perform the following:

```
$ cd /usr/local/packages6/conda/_envronments/
$ cp anaconda2-2.3.0.yml anaconda2-x.y.z.yml
```

then edit that file modifying the environment name and the anaconda version under requirements then run:

```
$ conda env create -f anaconda2-x.y.z.yml
```

then repeat for the Python 3 installation.

2.1.22 R

R

Dependencies BLAS
URL http://www.r-project.org/
Documentation http://www.r-project.org/

R is a statistical computing language.

Interactive Usage

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` command.

The latest version of R can be loaded with

```
module load apps/R
```

Alternatively, you can load a specific version of R using one of the following

```
module load apps/R/3.2.2
module load apps/R/3.2.1
module load apps/R/3.2.0
module load apps/R/3.1.2
```

R can then be run with

```
$ R
```

Serial (one CPU) Batch usage

Here, we assume that you wish to run the program `my_code.R` on the system. With batch usage it is recommended to load a specific version of R, for example `module load apps/R/3.2.2`, to ensure the expected output is achieved.

First, you need to write a batch submission file. We assume you'll call this `my_job.sge`

```
#!/bin/bash
#$ -S /bin/bash
#$ -cwd               # Run job from current directory

module load apps/R/3.2.2     # Recommended to load a specific version of R

R CMD BATCH my_code.R my_code.R.o$JOB_ID
```

Note that R must be called with both the `CMD` and `BATCH` options which tell it to run an R program, in this case `my_code.R`. If you do not do this, R will attempt to open an interactive prompt.

The final argument, `my_code.R.o$JOBID`, tells R to send output to a file with this name. Since `$JOBID` will always be unique, this ensures that all of your output files are unique. Without this argument R sends all output to a file called `my_code.Rout`.

Ensuring that `my_code.R` and `my_job.sge` are both in your current working directory, submit your job to the batch system

```
qsub my_job.sge
```

Replace `my_job.sge` with the name of your submission script.

Graphical output

By default, graphical output from batch jobs is sent to a file called `Rplots.pdf`

Installing additional packages

As you will not have permissions to install packages to the default folder, additional R packages can be installed to your home folder `~/`. To create the appropriate folder, install your first package in R in interactive mode. Load an interactive R session as described above, and install a package with

```
install.packages()
```

You will be prompted to create a personal package library. Choose yes. The package will download and install from a CRAN mirror (you may be asked to select a nearby mirror, which you can do simply by entering the number of your preferred mirror).

Once the chosen package has been installed, additional packages can be installed either in the same way, or by creating a .R script. An example script might look like

```
install.packages("dplyr")
install.packages("devtools")
```

Call this using `source()`. For example if your script is called `packages.R` and is stored in your home folder, source this from an interactive R session with

```
source("~/packages.R")
```

These additional packages will be installed without prompting to your personal package library.

To check your packages are up to date, and update them if necessary, run the following line from an R interactive session

```
update.packages(lib.loc = "~/R/x86_64-unknown-linux-gnu-library/3.2/")
```

The folder name after `~/R/` will likely change, but this can be completed with tab autocompletion from the R session. Ensure `lib.loc` folder is specified, or R will attempt to update the wrong library.

R Packages that require external libraries

Some R packages require external libraries to be installed before you can install and use them. Since there are so many, we only install those libraries that have been explicitly requested by users of the system.

The associated R packages are not included in the system install of R, so you will need to install them yourself to your home directory following the instructions linked to below.

- *geos* This is the library required for the `rgeos` package.
- *JAGS* This is the library required for the `rjags` and `runjags` packages

Using the Rmath library in C Programs

The Rmath library allows you to access some of R's functionality from a C program. For example, consider the C-program below

```
#include <stdio.h>
#define MATHLIB_STANDALONE
#include "Rmath.h"

main(){
   double shape1,shape2,prob;

   shape1 = 1.0;
   shape2 = 2.0;
   prob = 0.5;

   printf("Critical value is %lf\n",qbeta(prob,shape1,shape2,1,0));
}
```

This makes use of R's `qbeta` function. You can compile and run this on a worker node. Start a session on a worker node with `qrsh` or `qsh` and load the R module

```
module load apps/R/3.2.2
```

Assuming the program is called `test_rmath.c`, compile with

```
gcc test_rmath.c -lRmath -lm -o test_rmath
```

For full details about the functions made available by the Rmath library, see section 6.7 of the document Writing R extensions

Installation Notes

These notes are primarily for administrators of the system.

Version 3.2.2

- What's new in R version 3.2.2

This was a scripted install. It was compiled from source with gcc 4.4.7 and with `--enable-R-shlib` enabled. You will need a large memory `qrsh` session in order to successfully run the build script. I used `qrsh -l rmem=8G -l mem=16G`

- install_R_3.2.2.sh Downloads, compiles and installs R 3.2.2 and the `Rmath` library.
- R 3.2.2 Modulefile located on the system at `/usr/local/modulefiles/apps/R/3.2.2`
- Install log-files were manually copied to `/usr/local/packages6/R/3.2.2/install_logs` on the system. This step should be included in the next version of the install script.

Version 3.2.1

This was a manual install. It was compiled from source with gcc 4.4.7 and with `--enable-R-shlib` enabled.

- Install notes
- R 3.2.1 Modulefile located on the system at `/usr/local/modulefiles/apps/R/3.2.1`

Older versions

Install notes for older versions of R are not available.

2.1.23 Samtools

Samtools

Versions 1.2
URL http://samtools.sourceforge.net/

SAM (Sequence Alignment/Map) format is a generic format for storing large nucleotide sequence alignments

Interactive Usage

After connecting to Iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` command.

The lastest version of Samtools (currently 1.2) is made available with the command

```
module load apps/gcc/5.2/samtools
```

Alternatively, you can load a specific version with

```
module load apps/gcc/5.2/samtools/1.2
```

This command makes the samtools binary directory available to your session.

Documentation

Once you have made samtools available to the system using the *module* command above, you can read the man pages by typing

```
man samtools
```

Installation notes

Samtools was installed using gcc 5.2

```
module load compilers/gcc/5.2
```

```
tar -xvjf ./samtools-1.2.tar.bz2
cd samtools-1.2
mkdir -p /usr/local/packages6/apps/gcc/5.2/samtools/1.2
make prefix=/usr/local/packages6/apps/gcc/5.2/samtools/1.2
make prefix=/usr/local/packages6/apps/gcc/5.2/samtools/1.2 install
```

Testing

The test suite was run with

```
make test 2>&1 | tee make_tests.log
```

The summary of the test output was

```
Test output:
Number of tests:
    total            .. 368
    passed           .. 336
    failed           .. 0
    expected failure .. 32
    unexpected pass  .. 0

test/merge/test_bam_translate test/merge/test_bam_translate.tmp
test/merge/test_pretty_header
test/merge/test_rtrans_build
test/merge/test_trans_tbl_init
cd test/mpileup && ./regression.sh
Samtools mpileup tests:

EXPECTED FAIL: Task failed, but expected to fail;
when running $samtools mpileup -x -d 8500 -B -f mpileup.ref.fa deep.sam|awk '{print $4}'

Expected   passes:   123
Unexpected passes:   0
Expected   failures: 1
Unexpected failures: 0
```

The full log is on the system at */usr/local/packages6/apps/gcc/5.2/samtools/1.2/make_tests.log*

Modulefile

- The module file is on the system at */usr/local/modulefiles/apps/gcc/5.2/samtools/1.2*
- The module file is on github.

2.2 Libraries

2.2.1 Boost C++ Library v1.41

Boost version 1.41

Version 1.41
Support Level Bronze
Dependancies libs/gcc/4.4.7/icu/42
URL www.boost.org
Documentation http://www.boost.org/doc/libs/1_41_0/
Location /usr/local/packages6/libs/gcc/4.4.7/boost/1.41

Boost provides free peer-reviewed portable C++ source libraries.

Usage

This build of the Boost library was built with gcc 4.4.7 and so should only be used with that version of gcc. To make the library available, run the following module command.

```
module load libs/gcc/4.4.7/boost/1.41
```

Build a simple program using Boost

Many boost libraries are header-only which makes them particularly simple to compile. The following program reads a sequence of integers from standard input, uses Boost.Lambda to multiply each number by three, and writes them to standard output (taken from http://www.boost.org/doc/libs/1_41_0/more/getting_started/unix-variants.html):

```
#include <boost/lambda/lambda.hpp>
#include <iostream>
#include <iterator>
#include <algorithm>

int main()
{
    using namespace boost::lambda;
    typedef std::istream_iterator<int> in;

    std::for_each(
        in(std::cin), in(), std::cout << (_1 * 3) << " " );
}
```

Copy this into a file called example1.cpp and compile with

```
g++ example1.cpp -o example
```

Provided you loaded the module given above, and you are using gcc version 4.4.7, the program should compile without error.

Linking to a Boost library

The following program is taken from the official Boost dcoumentation http://www.boost.org/doc/libs/1_41_0/more/getting_started/unix-variants.html

```
#include <boost/regex.hpp>
#include <iostream>
#include <string>

int main()
{
    std::string line;
    boost::regex pat( "^Subject: (Re: |Aw: )*(.*)" );

    while (std::cin)
    {
        std::getline(std::cin, line);
        boost::smatch matches;
        if (boost::regex_match(line, matches, pat))
            std::cout << matches[2] << std::endl;
    }
}
```

This program makes use of the Boost.Regex library, which has a separately-compiled binary component we need to link to. Assuming that the above program is called example2.cpp, compile with the following command

```
g++ example2.cpp -o example2 -lboost_regex
```

If you get an error message that looks like this:

```
example2.cpp:1:27:  error:  boost/regex.hpp:  No such file or directory
```

the most likely cause is that you forgot to load the module as detailed above.

Installation Notes

This section is primarily for administrators of the system.

This build of boost was built with gcc 4.4.7 and ICU version 42.

```
module load libs/gcc/4.4.7/icu/42
tar -xvzf ./boost_1_41_0.tar.gz
cd boost_1_41_0
./bootstrap.sh --prefix=/usr/local/packages6/libs/gcc/4.4.7/boost/1.41
./bjam -sICU_PATH=/usr/local/packages6/libs/gcc/4.4.7/icu/42 install
```

Testing

The two examples above were compiled and ran.

Module File

Module File Location: `/usr/local/modulefiles/libs/gcc/4.4.7/boost/1.41`

```
#%Module1.0#####################################################################
##
## Boost 1.41 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

module load libs/gcc/4.4.7/icu/42

proc ModulesHelp { } {
        puts stderr "Makes the Boost 1.41 library available"
}

set BOOST_DIR /usr/local/packages6/libs/gcc/4.4.7/boost/1.41

module-whatis   "Makes the Boost 1.41 library available"

prepend-path LD_LIBRARY_PATH $BOOST_DIR/lib
prepend-path CPLUS_INCLUDE_PATH $BOOST_DIR/include
prepend-path LIBRARY_PATH $BOOST_DIR/lib
```

2.2.2 Boost C++ Library v1.58

Boost version 1.58

Version 1.58
Support Level Bronze
Dependancies libs/gcc/4.8.2/libunistring/0.9.5, libs/gcc/4.8.2/icu/55, compilers/gcc/4.8.2
URL www.boost.org
Documentation http://www.boost.org/doc/libs/1_58_0/
Location /usr/local/packages6/libs/gcc/4.8.2/boost/1.58.0/

Boost provides free peer-reviewed portable C++ source libraries.

Usage

This build of the Boost library requires gcc version 4.8.2. To make the compiler and library available, run the following module commands

```
module load compilers/gcc/4.8.2
module load libs/gcc/4.8.2/boost/1.58
```

Build a simple program using Boost

Many boost libraries are header-only which makes them particularly simple to compile. The following program reads a sequence of integers from standard input, uses Boost.Lambda to multiply each number by three, and writes them to standard output (taken from http://www.boost.org/doc/libs/1_58_0/more/getting_started/unix-variants.html):

```
#include <boost/lambda/lambda.hpp>
#include <iostream>
#include <iterator>
#include <algorithm>

int main()
{
    using namespace boost::lambda;
    typedef std::istream_iterator<int> in;

    std::for_each(
        in(std::cin), in(), std::cout << (_1 * 3) << " " );
}
```

Copy this into a file called example1.cpp and compile with

```
g++ example1.cpp -o example
```

Provided you loaded the modules given above, and you are using gcc version 4.8.2, the program should compile without error.

Linking to a Boost library

The following program is taken from the official Boost documentation http://www.boost.org/doc/libs/1_58_0/more/getting_started/unix-variants.html

```
#include <boost/regex.hpp>
#include <iostream>
#include <string>

int main()
{
    std::string line;
    boost::regex pat( "^Subject: (Re: |Aw: )*(.*)" );

    while (std::cin)
    {
        std::getline(std::cin, line);
        boost::smatch matches;
        if (boost::regex_match(line, matches, pat))
```

```
            std::cout << matches[2] << std::endl;
    }
}
```

This program makes use of the Boost.Regex library, which has a separately-compiled binary component we need to link to. Assuming that the above program is called example2.cpp, compile with the following command

```
g++ example2.cpp -o example2 -lboost_regex
```

If you get an error message that looks like this:

```
example2.cpp:1:27:  error:  boost/regex.hpp:  No such file or directory
```

the most likely cause is that you forgot to load the module as detailed above.

Installation Notes

This section is primarily for administrators of the system.

```
module load compilers/gcc/4.8.2
module load libs/gcc/4.8.2/libunistring/0.9.5
module load libs/gcc/4.8.2/icu/55
tar -xvzf ./boost_1_58_0.tar.gz
cd boost_1_58_0
./bootstrap.sh --prefix=/usr/local/packages6/libs/gcc/4.8.2/boost/1.58.0/
```

It complained that it could not find the icu library but when I ran

```
./b2 install --prefix=/usr/local/packages6/libs/gcc/4.8.2/boost/1.58.0
```

It said that it had detected the icu library and was compiling it in

Testing

Compiled and ran the two example files given above.

Module File

Module File Location: `/usr/local/modulefiles/libs/gcc/4.8.2/boost/1.58`

```
#%Module1.0#####################################################################
##
## boost 1.58 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

module load libs/gcc/4.8.2/libunistring/0.9.5
module load libs/gcc/4.8.2/icu/55

proc ModulesHelp { } {
        puts stderr "Makes the Boost 1.58 library available"
}

set BOOST_DIR /usr/local/packages6/libs/gcc/4.8.2/boost/1.58.0
```

```
module-whatis   "Makes the Boost 1.58 library available"

prepend-path LD_LIBRARY_PATH $BOOST_DIR/lib
prepend-path CPLUS_INCLUDE_PATH $BOOST_DIR/include
prepend-path LIBRARY_PATH $BOOST_DIR/lib
```

2.2.3 cgns

cgns

Version 3.2.1
Support Level Bronze
Dependancies libs/hdf5/gcc/openmpi/1.8.14
URL http://cgns.github.io/WhatIsCGNS.html
Location /usr/local/packages6/libs/gcc/4.4.7/cgnslib

The CFD General Notation System (CGNS) provides a general, portable, and extensible standard for the storage and retrieval of computational fluid dynamics (CFD) analysis data.

Usage

To make this library available, run the following module command

```
module load libs/gcc/4.4.7/cgns/3.2.1
```

This will also load the module files for the prerequisite libraries, Open MPI 1.8.3 and HDF5 1.8.14 with parallel support.

Installing

This section is primarily for administrators of the system.

- This is a prerequisite for Code Saturne version 4.0.
- It was built with gcc 4.4.7, openmpi 1.8.3 and hdf 1.8.14

```
module load libs/hdf5/gcc/openmpi/1.8.14
tar -xvzf cgnslib_3.2.1.tar.gz
mkdir /usr/local/packages6/libs/gcc/4.4.7/cgnslib
cd /usr/local/packages6/libs/gcc/4.4.7/cgnslib
mkdir 3.2.1
cd 3.2.1
cmake ~/cgnslib_3.2.1/
ccmake .
```

Configured the following using ccmake

```
CGNS_ENABLE_PARALLEL            ON
 MPIEXEC                         /usr/local/mpi/gcc/openmpi/1.8.3/bin/mpiexec
 MPI_COMPILER                    /usr/local/mpi/gcc/openmpi/1.8.3/bin/mpic++
 MPI_EXTRA_LIBRARY               /usr/local/mpi/gcc/openmpi/1.8.3/lib/libmpi.s
 MPI_INCLUDE_PATH                /usr/local/mpi/gcc/openmpi/1.8.3/include
 MPI_LIBRARY                     /usr/local/mpi/gcc/openmpi/1.8.3/lib/libmpi_c
 ZLIB_LIBRARY                    /usr/lib64/libz.so
```

```
FORTRAN_NAMING                   LOWERCASE_
 HDF5_INCLUDE_PATH               /usr/local/packages6/hdf5/gcc-4.4.7/openmpi-1.8.3/hdf5-1.8.14/inclu
 HDF5_LIBRARY                    /usr/local/packages6/hdf5/gcc-4.4.7/openmpi-1.8.3/hdf5-1.8.14/lib/li
 HDF5_NEED_MPI                    ON
 HDF5_NEED_SZIP                   OFF
 HDF5_NEED_ZLIB                   ON
 CGNS_BUILD_CGNSTOOLS             OFF
 CGNS_BUILD_SHARED                ON
 CGNS_ENABLE_64BIT                ON
 CGNS_ENABLE_FORTRAN              ON
 CGNS_ENABLE_HDF5                 ON
 CGNS_ENABLE_SCOPING              OFF
 CGNS_ENABLE_TESTS                ON
 CGNS_USE_SHARED                  ON
 CMAKE_BUILD_TYPE                 Release
 CMAKE_INSTALL_PREFIX             /usr/local/packages6/libs/gcc/4.4.7/cgnslib/3.2.1
```

Once the configuration was complete, I did

```
make
make install
```

Module File

Module File Location: `/usr/local/modulefiles/libs/gcc/4.4.7/cgns/3.2.1`

```
##%Module1.0#####################################################################
##
## cgns 3.2.1 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        puts stderr "Makes the cgns 3.2.1 library available"
}

module-whatis   "Makes the cgns 3.2.1 library available"
module load libs/hdf5/gcc/openmpi/1.8.14

set CGNS_DIR /usr/local/packages6/libs/gcc/4.4.7/cgnslib/3.2.1

prepend-path LD_LIBRARY_PATH $CGNS_DIR/lib
prepend-path CPATH $CGNS_DIR/include
```

2.2.4 fftw

fftw

Latest version 3.3.4
URL http://www.fftw.org/
Location /usr/local/packages6/libs/gcc/5.2/fftw/3.3.4

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST).

Usage

To make this library available, run the following module command

```
module load libs/gcc/5.2/fftw/3.3.4
```

Installation notes

This section is primarily for administrators of the system. FFTW 3.3.4 was compiled with gcc 5.2

```
module load compilers/gcc/5.2
mkdir -p /usr/local/packages6/libs/gcc/5.2/fftw/3.3.4
tar -xvzf fftw-3.3.4.tar.gz
cd fftw-3.3.4
./configure --prefix=/usr/local/packages6/libs/gcc/5.2/fftw/3.3.4 --enable-threads --enable-openmp --
make
make check
```

Result was lots of numerical output and

```
-------------------------------------------------------------
         FFTW transforms passed basic tests!
-------------------------------------------------------------

-------------------------------------------------------------
         FFTW threaded transforms passed basic tests!
-------------------------------------------------------------
```

Installed with

```
make install
```

Module file

Modulefile is on the system at `/usr/local/modulefiles/libs/gcc/5.2/fftw/3.3.4`

```
#%Module1.0#####################################################################
##
## fftw 3.3.4 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

module load compilers/gcc/5.2

proc ModulesHelp { } {
        puts stderr "Makes the FFTW 3.3.4 library available"
}

set FFTW_DIR /usr/local/packages6/libs/gcc/5.2/fftw/3.3.4
```

```
module-whatis   "Makes the FFTW 3.3.4 library available"

prepend-path LD_LIBRARY_PATH $FFTW_DIR/lib
prepend-path CPLUS_INCLUDE_PATH $FFTW_DIR/include
prepend-path LIBRARY_PATH $FFTW_DIR/lib
```

2.2.5 FLTK

FLTK

Version 1.3.3
URL http://www.fltk.org/index.php
Location /usr/local/packages6/libs/gcc/5.2/fltk/1.3.3

FLTK (pronounced "fulltick") is a cross-platform C++ GUI toolkit for UNIX®/Linux® (X11), Microsoft® Windows®, and MacOS® X. FLTK provides modern GUI functionality without the bloat and supports 3D graphics via OpenGL® and its built-in GLUT emulation.

Usage

```
module load libs/gcc/5.2/fltk/1.3.3
```

Installation notes

This section is primarily for administrators of the system.

- This is a pre-requisite for GNU Octave version 4.0
- It was built with gcc 5.2

```
module load compilers/gcc/5.2
mkdir -p /usr/local/packages6/libs/gcc/5.2/fltk/1.3.3
tar -xvzf ./fltk-1.3.3-source.tar.gz
cd fltk-1.3.3-source
./configure --prefix=/usr/local/packages6/libs/gcc/5.2/fltk/1.3.3 --enabled-shared
make
make install
```

Module File

Modulefile at `/usr/local/modulefiles/libs/gcc/5.2/fltk/1.3.3`

```
#%Module1.0#####################################################################
##
## fltk 1.3.3 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##
```

```
module load compilers/gcc/5.2

proc ModulesHelp { } {
        puts stderr "Makes the FLTK 1.3.3 library available"
}

set FLTK_DIR /usr/local/packages6/libs/gcc/5.2/fltk/1.3.3

module-whatis   "Makes the FLTK 1.3.3 library available"

prepend-path LD_LIBRARY_PATH $FLTK_DIR/lib
prepend-path CPLUS_INCLUDE_PATH $FLTK_DIR/include
prepend-path LIBRARY_PATH $FLTK_DIR/lib
prepend-path PATH $FLTK_DIR/bin
```

2.2.6 geos

geos

Version 3.4.2
Support Level Bronze
Dependancies compilers/gcc/4.8.2
URL http://trac.osgeo.org/geos/
Location /usr/local/packages6/libs/gcc/4.8.2/geos/3.4.2

GEOS - Geometry Engine, Open Source

Usage

To make this library available, run the following module commands

```
module load compilers/gcc/4.8.2
module load libs/gcc/4.8.2/geos/3.4.2
```

We load version 4.8.2 of gcc since gcc 4.8.2 was used to build this version of geos.

The rgeos interface in R

rgeos is a CRAN package that provides an R interface to geos. It is not installed in R by default so you need to install a version in your home directory.

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qrsh` or `qsh` command. Run the following module commands

```
module load apps/R/3.2.0
module load compilers/gcc/4.8.2
module load libs/gcc/4.8.2/geos/3.4.2
```

Launch R and run the command

```
install.packages('rgeos')
```

If you've never installed an R package before on the system, it will ask you if you want to install to a personal library. Answer `y` to any questions you are asked.

The library will be installed to a sub-directory called `R` in your home directory and you should only need to perform the above procedure once.

Once you have performed the installation, you will only need to run the `module` commands above to make the geos library available to the system. Then, you use `rgeos` as you would any other library in R

```
library('rgeos')
```

Installation notes

This section is primarily for administrators of the system.

```
qrsh
tar -xvjf ./geos-3.4.2.tar.bz2
cd geos-3.4.2
mkdir -p /usr/local/packages6/libs/gcc/4.8.2/geos/3.4.2
module load compilers/gcc/4.8.2
./configure prefix=/usr/local/packages6/libs/gcc/4.8.2/geos/3.4.2
```

Potentially useful output at the end of the configure run

```
Swig: false
Python bindings: false
Ruby bindings: false
PHP bindings: false
```

Once the configuration was complete, I did

```
make
make install
```

Testing

Compile and run the test-suite with

```
make check
```

All tests passed.

Module File

Module File Location: `/usr/local/modulefiles/libs/gcc/4.8.2/geos/3.4.2`

```
more /usr/local/modulefiles/libs/gcc/4.8.2/geos/3.4.2
#%Module1.0#####################################################################
##
## geos 3.4.2 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        puts stderr "Makes the geos 3.4.2 library available"
}
```

```
set GEOS_DIR /usr/local/packages6/libs/gcc/4.8.2/geos/3.4.2

module-whatis   "Makes the geos 3.4.2 library available"

prepend-path LD_LIBRARY_PATH $GEOS_DIR/lib
prepend-path PATH $GEOS_DIR/bin
```

2.2.7 HDF5 (gcc build)

HDF5 (gcc build)

Version 1.8.14 and 1.8.13
Support Level bronze
Dependancies openmpi (1.8.3)
URL http://www.hdfgroup.org/HDF5/
Documentation http://www.hdfgroup.org/HDF5/doc/

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.

Two primary versions of this library are provided, MPI parallel enabled versions and serial versions.

Usage - Serial

The serial versions were built with gcc version 4.8.2. As such, if you are going to build anything against these versions of HDF5, we recommend that you use gcc 4.8.2 which can be enabled with the following module command

```
module load compilers/gcc/4.8.2
```

To enable the serial version of HDF5, use one of the following module commands depending on which version of the library you require:

```
module load libs/hdf5/gcc/1.8.14
module load libs/hdf5/gcc/1.8.13
```

Usage – Parallel

The MPI Parallel version was built using gcc version 4.4.7 and OpenMPI version 1.8.3. Version 4.4.7 of gcc is the default compiler on the system so no module command is required for this.

To make the MPI version of HDF5 available, use one of the following module commands

```
module load libs/hdf5/gcc/openmpi/1.8.14
module load libs/hdf5/gcc/openmpi/1.8.13
```

It is not necessary to load the OpenMPI module in either case since this is done automatically on execution of one of the above commands.

Optional HDF5 features

Our HDF5 builds have zlib support built in but do not have support for the SZIP Encoder.

Installation notes

This section is primarily for administrators of the system.

This package is built from the source code distribution from the HDF Group website.

Two primary versions of this library are provided, a MPI parallel enabled version and a serial version. The serial version has the following configuration flags enabled:

```
--enable-fortran --enable-fortran2003 --enable-cxx --enable-shared
```

The parallel version has the following flags:

```
--enable-fortran --enable-fortran2003 --enable-shared --enable-parallel
```

The parallel library does not support C++, hence it being disabled for the parallel build.

2.2.8 HDF5 (PGI build)

HDF5 (PGI build)

Latest version 1.8.15-patch1
Dependancies PGI Compiler 15.7, PGI Openmpi 1.8.8
URL http://www.hdfgroup.org/HDF5/
Documentation http://www.hdfgroup.org/HDF5/doc/

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.

Usage

This version of HDF5 was compiled using version 15.7 of the PGI Compiler and OpenMPI 1.8.8. To make it available, run the following module command after starting a `qsh` or `qrsh` session

```
module load libs/hdf5/pgi/1.8.15-patch1
```

This module also loads the relevant modules for OpenMPI and PGI Compiler. To see which modules have been loaded, use the command `module list`

Installation notes

This section is primarily for administrators of the system.

Version 1.8.15-patch1

Compiled using PGI 15.7 and OpenMPI 1.8.8

- install_pgi_hdf5_1.8.15-patch1.sh Install script
- Modulefile located on the system at `/usr/local/modulefiles/libs/hdf5/pgi/1.8.15-patch1`

2.2.9 icu - International Components for Unicode

icu version 42

Version 42
Support Level Bronze
Dependancies compilers/gcc/4.4.7
URL http://site.icu-project.org/
Location /usr/local/packages6/libs/gcc/4.4.7/icu/42

ICU is the premier library for software internationalization, used by a wide array of companies and organizations

Usage

To make the library available, run the following module command

```
module load libs/gcc/4.4.7/icu/42
```

Installation notes

This section is primarily for administrators of the system.

icu version 42 is the version of icu linked to by the system version of boost – version 1.41. This build is to allow compilation of that version of boost.

```
tar -xvzf icu4c-4_2_1-src.tgz
cd icu/source/
./runConfigureICU Linux/gcc --prefix=/usr/local/packages6/libs/gcc/4.4.7/icu/42
make
make install
```

Testing

```
make check
```

Last few lines of output were

```
All tests passed successfully...]
Elapsed Time: 00:00:12.000
make[2]: Leaving directory `/home/fe1mpc/icu/source/test/cintltst'
---------------
ALL TESTS SUMMARY:
ok:  testdata iotest cintltst
===== ERRS:  intltest
make[1]: *** [check-recursive] Error 1
make[1]: Leaving directory `/home/fe1mpc/icu/source/test'
make: *** [check-recursive] Error 2
```

The error can be ignored since it is a bug in the test itself:

- http://sourceforge.net/p/icu/mailman/message/32443311/

Module File

Module File Location: `/usr/local/modulefiles/libs/gcc/4.4.7/icu/42`

```
##%Module1.0#####################################################################
##
## icu 42 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##


proc ModulesHelp { } {
        puts stderr "Makes the icu library available"
}

set ICU_DIR /usr/local/packages6/libs/gcc/4.4.7/icu/42

module-whatis   "Makes the icu library available"

prepend-path LD_LIBRARY_PATH $ICU_DIR/lib
prepend-path LIBRARY_PATH $ICU_DIR/lib
prepend-path CPLUS_INCLUDE_PATH $ICU_DIR/include
```

2.2.10 icu - International Components for Unicode

icu version 55

Version 55
Support Level Bronze
Dependancies compilers/gcc/4.8.2
URL http://site.icu-project.org/
Location /usr/local/packages6/libs/gcc/4.8.2/icu/55

ICU is the premier library for software internationalization, used by a wide array of companies and organizations

Usage

This build of the icu library requires gcc version 4.8.2. To make the compiler and library available, run the following module commands

```
module load compilers/gcc/4.8.2
module load libs/gcc/4.8.2/icu/55
```

Installation Notes

This section is primarily for administrators of the system.

Icu 55 is a pre-requisite for the version of boost required for an experimental R module used by one of our users.

```
module load compilers/gcc/4.8.2
tar -xvzf icu4c-55_1-src.tgz
cd icu/source
./runConfigureICU Linux/gcc --prefix=/usr/local/packages6/libs/gcc/4.8.2/icu/55/
make
```

Testing

make check

Last few lines of output were

```
[All tests passed successfully...]
 Elapsed Time: 00:00:00.086
 make[2]: Leaving directory `/home/fe1mpc/icu/icu/source/test/letest'
 ---------------
 ALL TESTS SUMMARY:
 All tests OK:  testdata intltest iotest cintltst letest
 make[1]: Leaving directory `/home/fe1mpc/icu/icu/source/test'
 make[1]: Entering directory `/home/fe1mpc/icu/icu/source'
 verifying that icu-config --selfcheck can operate
 verifying that make -f Makefile.inc selfcheck can operate
 PASS: config selfcheck OK
 rm -rf test-local.xml
```

Module File

Module File Location: */usr/local/modulefiles/libs/gcc/4.8.2/icu/55*

```
##%Module1.0#####################################################################
##
## icu 55 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##


proc ModulesHelp { } {
        puts stderr "Makes the icu library available"
}

set ICU_DIR /usr/local/packages6/libs/gcc/4.8.2/icu/55

module-whatis   "Makes the icu library available"

prepend-path LD_LIBRARY_PATH $ICU_DIR/lib
prepend-path LIBRARY_PATH $ICU_DIR/lib
prepend-path CPLUS_INCLUDE_PATH $ICU_DIR/include
```

2.2.11 MED

MED

Version 3.0.8
Support Level Bronze
URL http://www.salome-platform.org/downloads/current-version
Location /usr/local/packages6/libs/gcc/4.4.7/med/3.0.8

The purpose of the MED module is to provide a standard for storing and recovering computer data associated to numerical meshes and fields, and to facilitate the exchange between codes and solvers.

Usage

To make this library available, run the following module command

```
module load libs/gcc/4.4.7/med/3.0.8
```

Installation notes

This section is primarily for administrators of the system.

- This is a pre-requisite for Code Saturne version 4.0.
- It was built with gcc 4.4.7, openmpi 1.8.3 and hdf5 1.8.14

```
module load mpi/gcc/openmpi/1.8.3
tar -xvzf med-3.0.8.tar.gz
cd med-3.0.8
mkdir -p /usr/local/packages6/libs/gcc/4.4.7/med/3.0.8
./configure --prefix=/usr/local/packages6/libs/gcc/4.4.7/med/3.0.8 --disable-fortran --with-hdf5=/usr
make
make install
```

Fortran was disabled because otherwise the build failed with compilation errors. It's not needed for Code Saturne 4.0.

Python was disabled because it didn't have MPI support.

testing

The following was submiited as an SGE job from the med-3.0.8 build directory

```
#!/bin/bash

#$ -pe openmpi-ib 8
#$ -l mem=6G

module load mpi/gcc/openmpi/1.8.3
make check
```

All tests passed

Module File

```
#%Module1.0#####################################################################
##
## MED 3.0.8 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        puts stderr "Makes the MED 3.0.8 library available"
}

module-whatis   "Makes the MED 3.0.8 library available"

set MED_DIR /usr/local/packages6/libs/gcc/4.4.7/med/3.0.8

prepend-path LD_LIBRARY_PATH $MED_DIR/lib64
prepend-path CPATH $MED_DIR/include
```

2.2.12 NAG Fortran Library (Serial)

Produced by experts for use in a variety of applications, the NAG Fortran Library has a global reputation for its excellence and, with hundreds of fully documented and tested routines, is the largest collection of mathematical and statistical algorithms available.

This is the serial (1 CPU core) version of the NAG Fortran Library. For many routines, you may find it beneficial to use the parallel version of the library.

Usage

There are several versions of the NAG Fortran Library available. The one you choose depends on which compiler you are using. As with many libraries installed on the system, NAG libraries are made available via `module` commands which are only available once you have started a `qrsh` or `qsh` session.

In addition to loading a module for the library, you will usually need to load a module for the compiler you are using.

NAG for Intel Fortran

Use the following command to make Mark 25 of the serial (1 CPU core) version of the NAG Fortran Library for Intel compilers available

```
module load libs/intel/15/NAG/fll6i25dcl
```

Once you have ensured that you have loaded the module for the *Intel Compilers* you can compile your NAG program using

```
ifort your_code.f90 -lnag_mkl -o your_code.exe
```

which links to a version of the NAG library that's linked against the high performance Intel MKL (which provides high-performance versions of the BLAS and LAPACK libraries). Alternatively, you can compile using

```
ifort your_code.f90 -lnag_nag -o your_code.exe
```

Which is linked against a reference version of BLAS and LAPACK. If you are in any doubt as to which to choose, we suggest that you use `-lnag_mkl`

Running NAG's example programs

Most of NAG's routines come with example programs that show how to use them. When you use the `module` command to load a version of the NAG library, the script `nag_example` becomes available. Providing this script wth the name of the NAG routine you are interested in will copy, compile and run the example program for that routine into your current working directory.

For example, here is an example output for the NAG routine `a00aaf` which identifies the version of the NAG library you are using. If you try this yourself, the output you get will vary according to which version of the NAG library you are using

```
nag_example a00aaf
```

If you have loaded the `module` for fll6i25dcl this will give the following output

```
Copying a00aafe.f90 to current directory
cp /usr/local/packages6/libs/intel/15/NAG/fll6i25dcl/examples/source/a00aafe.f90 .

Compiling and linking a00aafe.f90 to produce executable a00aafe.exe
ifort -I/usr/local/packages6/libs/intel/15/NAG/fll6i25dcl/nag_interface_blocks a00aafe.f90 /usr/local

Running a00aafe.exe
./a00aafe.exe > a00aafe.r
 A00AAF Example Program Results

 *** Start of NAG Library implementation details ***

 Implementation title: Linux 64 (Intel 64 / AMD64), Intel Fortran, Double Precision (32-bit integers)
            Precision: FORTRAN double precision
         Product Code: FLL6I25DCL
                 Mark: 25.1.20150610 (self-contained)

 *** End of NAG Library implementation details ***
```

Functionality

The key numerical and statistical capabilities of the Fortran Library are shown below.

- Click here for a complete list of the contents of the Library.
- Cick here to see what's new in Mark 25 of the library.

Numerical Facilities

- Optimization, both Local and Global
- Linear, quadratic, integer and nonlinear programming and least squares problems
- Ordinary and partial differential equations, and mesh generation
- Solution of dense, banded and sparse linear equations and eigenvalue problems
- Solution of linear and nonlinear least squares problems
- Curve and surface fitting and interpolation
- Special functions
- Numerical integration and integral equations
- Roots of nonlinear equations (including polynomials)

- Option Pricing Formulae
- Wavelet Transforms

Statistical Facilities

- Random number generation
- Simple calculations on statistical data
- Correlation and regression analysis
- Multivariate methods
- Analysis of variance and contingency table analysis
- Time series analysis
- Nonparametric statistics

Documentation

- The NAG Fortran Library Manual (Link to NAG's webbsite)

Installation notes

fll6i25dcl

These are primarily for system administrators

```
tar -xvzf ./fll6i25dcl.tgz
./install.sh
```

The installer is interactive. Answer the installer questions as follows

```
Do you wish to install NAG Mark 25 Library? (yes/no):
yes
```

License file gets shown

```
[accept/decline]? :
accept

Where do you want to install the NAG Fortran Library Mark 25?
Press return for default location (/opt/NAG)
or enter an alternative path.
The directory will be created if it does not already exist.
>
/usr/local/packages6/libs/intel/15/NAG/
```

Module Files

fll6i25dcl

- The module file is on the system at `/usr/local/modulefiles/libs/intel/15/NAG/fll6i25dcl`
- The module file is on github.

2.2.13 NetCDF (PGI build)

NetCDF

Latest version
Dependancies PGI Compiler 15.7, PGI OpenMPI 1.8.8, PGI HDF5 1.8.15-patch1
URL http://www.unidata.ucar.edu/software/netcdf/
Documentation http://www.unidata.ucar.edu/software/netcdf/docs/

NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. NetCDF was developed and is maintained at Unidata. Unidata provides data and software tools for use in geoscience education and research. Unidata is part of the University Corporation for Atmospheric Research (UCAR) Community Programs (UCP). Unidata is funded primarily by the National Science Foundation.

Usage

This version of NetCDF was compiled using version 15.7 of the PGI Compiler, OpenMPI 1.8.8 and HDF5 1.8.15-patch1. To make it available, run the following module command after starting a `qsh` or `qrsh` session

```
module load libs/pgi/netcdf/4.3.3.1
```

This module also loads the relevant modules for OpenMPI, PGI Compiler and HDF5. To see which modules have been loaded, use the command `module list`

Installation notes

This section is primarily for administrators of the system.

Version 4.3.3.1

Compiled using PGI 15.7, OpenMPI 1.8.8 and HDF5 1.8.15-patch1

- install_pgi_netcdf_4.3.3.1.sh Install script
- Install logs are on the system at `/usr/local/packages6/libs/pgi/netcdf/4.3.3.1/install_logs`
- Modulefile located on the system at `ls /usr/local/modulefiles/libs/pgi/netcdf/4.3.3.1`

2.2.14 pcre

pcre

Version 8.37
Support Level Bronze
Dependancies None
URL http://www.pcre.org/
Location /usr/local/packages6/libs/gcc/4.4.7/pcre/8.37

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5. PCRE has its own native API, as well as a set of wrapper functions that correspond to the POSIX regular expression API. The PCRE library is free, even for building proprietary software.

Usage

To make this library available, run the following module command after starting a `qsh` or `qrsh` session.

```
module load libs/gcc/4.4.7/pcre/8.37
```

This also makes the updated `pcregrep` command available and will replace the system version. Check the version you are using with `pcregrep -V`

```
pcregrep -V

pcregrep version 8.37 2015-04-28
```

Installation notes

This section is primarily for administrators of the system.

```
qrsh
tar -xvzf pcre-8.37.tar.gz
cd pcre-8.37
mkdir -p /usr/local/packages6/libs/gcc/4.4.7/pcre/8.37
./configure --prefix=/usr/local/packages6/libs/gcc/4.4.7/pcre/8.37
```

The configuration details were

```
pcre-8.37 configuration summary:

Install prefix .................. : /usr/local/packages6/libs/gcc/4.4.7/pcre/8.37
C preprocessor .................. : gcc -E
C compiler ...................... : gcc
C++ preprocessor ................ : g++ -E
C++ compiler .................... : g++
Linker .......................... : /usr/bin/ld -m elf_x86_64
C preprocessor flags ............ :
C compiler flags ................ : -g -O2 -fvisibility=hidden
C++ compiler flags .............. : -O2 -fvisibility=hidden -fvisibility-inlines-hidden
Linker flags .................... :
Extra libraries ................. :

Build 8 bit pcre library ........ : yes
Build 16 bit pcre library ....... : no
Build 32 bit pcre library ....... : no
Build C++ library ............... : yes
Enable JIT compiling support .... : no
Enable UTF-8/16/32 support ...... : no
Unicode properties .............. : no
Newline char/sequence ........... : lf
\R matches only ANYCRLF ......... : no
EBCDIC coding ................... : no
EBCDIC code for NL .............. : n/a
Rebuild char tables ............. : no
Use stack recursion ............. : yes
POSIX mem threshold ............. : 10
Internal link size .............. : 2
Nested parentheses limit ........ : 250
Match limit ..................... : 10000000
Match limit recursion ........... : MATCH_LIMIT
Build shared libs ............... : yes
```

```
Build static libs ..... : yes
Use JIT in pcregrep ..... : no
Buffer size for pcregrep ..... : 20480
Link pcregrep with libz ..... : no
Link pcregrep with libbz2 ..... : no
Link pcretest with libedit ..... : no
Link pcretest with libreadline .. : no
Valgrind support ..... : no
Code coverage ..... : no
```

Once the configuration was complete, I did

```
make
make install
```

Testing was performed and all tests passed

```
make check

============================================================================
Testsuite summary for PCRE 8.37
============================================================================
# TOTAL: 5
# PASS:  5
# SKIP:  0
# XFAIL: 0
# FAIL:  0
# XPASS: 0
# ERROR: 0
============================================================================
```

Module File

Module File Location: `/usr/local/modulefiles/libs/gcc/4.4.7/pcre/8.37`

```
#%Module1.0#####################################################################
##
## pcre 8.37 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        puts stderr "Makes the pcre 8.37 library available"
}

module-whatis   "Makes the pcre 8.37 library available"

set PCRE_DIR /usr/local/packages6/libs/gcc/4.4.7/pcre/8.37

prepend-path LD_LIBRARY_PATH $PCRE_DIR/lib
prepend-path CPATH $PCRE_DIR/include
prepend-path PATH $PCRE_DIR/bin
```

2.2.15 SimBody

SimBody

Version 3.5.3
Support Level Bronze
Dependancies compilers/gcc/4.8.2
URL https://simtk.org/home/simbody
Location /usr/local/packages6/libs/gcc/4.8.2/simbody/3.5.3

Usage

To make this library available, run the following module command

```
module load libs/gcc/4.8.2/simbody
```

Installing

This section is primarily for administrators of the system.

Installed using the following procedure:

```
module load compilers/gcc/4.8.2
```

```
cmake ../simbody-Simbody-3.5.3/ -DCMAKE_INSTALL_PREFIX=/usr/local/packages6/libs/gcc/4.8.2/simbody/3
```

```
make -j 8
```

2.3 Compilers

2.3.1 CMake

CMake is a build tool commonly used when compiling other libraries.

CMake is installed in */usr/local/packages6/cmake*.

Usage

CMake can be loaded with:

```
module load compilers/cmake
```

Installation

Run the following commands:

```
module load apps/python/2.7

./bootstrap --prefix=/usr/local/packages6/cmake/3.3.0/
--mandir=/usr/local/packages6/cmake/3.3.0/man --sphinx-man

gmake -j8

gmake install
```

2.3.2 GNU Compiler Collection (gcc)

The GNU Compiler Collection (gcc) is a widely used, free collection of compilers including C (gcc), C++ (g++) and Fortran (gfortran). The defaut version of gcc on the system is 4.4.7

```
gcc -v

Using built-in specs.
Target: x86_64-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-
Thread model: posix
gcc version 4.4.7 20120313 (Red Hat 4.4.7-11) (GCC)
```

It is possible to switch to other versions of the gcc compiler suite using modules. After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qrsh` or *qsh* command. Choose the version of the compiler you wish to use using one of the following commands

```
module load compilers/gcc/5.2
module load compilers/gcc/4.8.2
module load compilers/gcc/4.5.3
```

Confirm that you've loaded the version of gcc you wanted using `gcc -v`.

- **NOTE (7th August 2015)**: The version 5.2 build has only just been built and has only been minimally tested. It should be treated as experimental for now.

Documentation

- What's new in the gcc version 5 series?

Installation Notes

These notes are primarily for system administrators

- gcc version 5.2 was installed using :
  - install_gcc_5.2.sh
  - gcc 5.2 modulefile located on the system at `/usr/local/modulefiles/compilers/gcc/5.2`
- Installation notes for versions 4.8.2 and below are not available.

2.3.3 git

git

Latest version 2.5
Dependancies gcc 5.2
URL https://git-scm.com/

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Usage

An old version of git is installed as part of the system's opertaing system. As such, it is available everywhere, including on the log-in nodes

```
$ git --version
git version 1.7.1
```

This was released in April 2010. We recommend that you load the most up to date version using modules - something that can only be done after starting an interactive `qrsh` or `qsh` session

```
module load apps/gcc/5.2/git/2.5
```

Installation notes

Version 2.5 of git was installed using gcc 5.2 using the following install script and module file:

- install_git_2.5.sh
- git 2.5 modulefile located on the system at `/usr/local/modulefiles/compilers/git/2.5`

2.3.4 Intel Compilers

Intel Compilers help create C, C++ and Fortran applications that can take full advantage of the advanced hardware capabilities available in Intel processors and co-processors. They also simplify that development by providing high level parallel models and built-in features like explicit vectorization and optimization reports.

Making the Intel Compilers available

After connecting to iceberg (see *Connect to iceberg*), start an interactive session with the `qsh` or `qrsh` command. To make one of the versions of the Intel Compiler available, run one of the following module commands

```
module load compilers/intel/15.0.3
module load compilers/intel/14.0
module load compilers/intel/12.1.15
module load compilers/intel/11.0
```

Compilation examples

C

To compile the C hello world example into an executable called `hello` using the Intel C compiler

```
icc hello.c -o hello
```

C++

To compile the C++ hello world example into an executable called `hello` using the Intel C++ compiler

```
icpc hello.cpp -o hello
```

Fortran

To compile the Fortran hello world example into an executable called `hello` using the Intel Fortran compiler

```
ifort hello.f90 -o hello
```

Detailed Documentation

Once you have loaded the module on Iceberg, `man` pages are available for Intel compiler products

```
man ifort
man icc
```

The following links are to Intel's website

- User and Reference Guide for the Intel® C++ Compiler 15.0
- User and Reference Guide for the Intel® Fortran Compiler 15.0
- Step by Step optimizing with Intel C++ Compiler

Related Software on the system

Users of the Intel Compilers may also find the following useful:

- *NAG Fortran Library (Serial)* - A library of over 1800 numerical and statistical functions.

Installation Notes

The following notes are primarily for system administrators.

- Version 15.0.3
  - The install is located on the system at `/usr/local/packages6/compilers/intel/2015/`
  - The license file is at `/usr/local/packages6/compilers/intel/license.lic`
  - The environment variable `INTEL_LICENSE_FILE` is set by the environment module and points to the license file location
  - Download the files `l_ccompxe_2015.3.187.tgz` (C/C++) and `l_fcompxe_2015.3.187.tgz` (Fortran) from Intel Portal.
  - Put the above .tgz files in the same directory as install_intel15.sh and silent_master.cfg
  - Run `install_intel15.sh`
  - To find what was required in the module file, I did

```
env > base.env
source /usr/local/packages6/compilers/intel/2015/composer_xe_2015.3.187/bin/compilervars.sh
env > after_intel.env
diff base.env after_intel.env
```

- The module file is on iceberg at `/usr/local/modulefiles/compilers/intel/15.0.3`

version 14 and below

Installation notes are not available for these older versions of the Intel Compiler.

2.3.5 NAG Fortran Compiler

The NAG Fortran Compiler is robust, highly tested, and valued by developers all over the globe for its checking capabilities and detailed error reporting. The Compiler is available on Unix platforms as well as for Microsoft Windows and Apple Mac platforms. Release 6.0 has extensive support for both legacy and modern Fortran features, and also supports parallel programming with OpenMP.

Making the NAG Compiler available

After connecting to iceberg (see *Connect to iceberg*), start an interactive sesssion with the `qsh` or `qrsh` command. To make the NAG Fortran Compiler available, run the following module command

```
module load compilers/NAG/6.0
```

Compilation examples

To compile the Fortran hello world example into an executable called `hello` using the NAG compiler

```
nagfor hello.f90 -o hello
```

Detailed Documentation

Once you've run the NAG Compiler module command, `man` documentation is available

```
man nagfor
```

Online documentation:

- PDF version of the NAG Fortran Compiler Manual
- NAG Fortran Compiler Documentation Index (NAG's Website)

Installation Notes

The following notes are primarily for system sysadmins

```
mkdir -p /usr/local/packages6/compilers/NAG/6.0
tar -xvzf ./npl6a60na_amd64.tgz
cd NAG_Fortran-amd64/
```

Run the interactive install script

```
./INSTALL.sh
```

Accept the license and answer the questions as follows

```
Install compiler binaries to where [/usr/local/bin]?
/usr/local/packages6/compilers/NAG/6.0

Install compiler library files to where [/usr/local/lib/NAG_Fortran]?
/usr/local/packages6/compilers/NAG/6.0/lib/NAG_Fortran

Install compiler man page to which directory [/usr/local/man/man1]?
/usr/local/packages6/compilers/NAG/6.0/man/man1

Suffix for compiler man page [1] (i.e. nagfor.1)?
Press enter

Install module man pages to which directory [/usr/local/man/man3]?
/usr/local/packages6/compilers/NAG/6.0/man/man3

Suffix for module man pages [3] (i.e. f90_gc.3)?
Press Enter
```

Licensing

Add the license key to /usr/local/packages5/nag/license.lic

The license key needs to be updated annually before 31st July.

Point to this license file using the environment variable $NAG_KUSARI_FILE

This is done in the environment module

Module File

Module file location is `/usr/local/modulefiles/compilers/NAG/6.0`

```
##%Module1.0#####################################################################
##
## NAG Fortran Compiler 6.0 module file
##

## Module file logging
source /usr/local/etc/module_logging.tcl
##

proc ModulesHelp { } {
        puts stderr "Makes the NAG Fortran Compiler v6.0 available"
}

module-whatis   "Makes the NAG Fortran Compiler v6.0 available"

set NAGFOR_DIR /usr/local/packages6/compilers/NAG/6.0

prepend-path PATH $NAGFOR_DIR
prepend-path MANPATH $NAGFOR_DIR/man
setenv NAG_KUSARI_FILE /usr/local/packages5/nag/license.lic
```

2.3.6 PGI Compilers

The PGI Compiler suite offers C,C++ and Fortran Compilers. For full details of the features of this compiler suite, see PGI's website at http://www.pgroup.com/products/pgiworkstation.htm

Making the PGI Compilers available

After connecting to iceberg (see *Connect to iceberg*), start an interactive session with the `qsh` or `qrsh` command. To make one of the versions of the PGI Compiler Suite available, run one of the following module commands

```
module load compilers/pgi/15.7
module load compilers/pgi/14.4
module load compilers/pgi/13.1
```

Compilation examples

C

To compile the C hello world example into an executable called `hello` using the PGI C compiler

```
pgcc hello.c -o hello
```

C++

To compile the C++ hello world example into an executable called `hello` using the PGI C++ compiler

```
pgc++ hello.cpp -o hello
```

Fortran

To compile the Fortran hello world example into an executable called `hello` using the PGI Fortran compiler

```
pgf90 hello.f90 -o hello
```

Additional resources

- *Using the PGI Compiler with GPUs on Iceberg*

Installation Notes

Version 15.7

The installer is interactive. Most of the questions are obvious. Here is how I answered the rest

Installation type

```
A network installation will save disk space by having only one copy of the
compilers and most of the libraries for all compilers on the network, and
the main installation needs to be done once for all systems on the network.

1  Single system install
2  Network install

Please choose install option: 1
```

Path

```
Please specify the directory path under which the software will be installed.
The default directory is /opt/pgi, but you may install anywhere you wish,
assuming you have permission to do so.

Installation directory? [/opt/pgi] /usr/local/packages6/compilers/pgi
```

CUDA and AMD components

```
Install CUDA Toolkit Components? (y/n) y
Install AMD software components? (y/n) y
```

AMCL version

```
This PGI version links with ACML 5.3.0 by default.  Also available:
  (1) ACML 5.3.0
  (2) ACML 5.3.0 using FMA4
Enter another value to override the default (1)
1
```

Other questions

```
Install JAVA JRE [yes] yes
Install OpenACC Unified Memory Evaluation package? (y/n) n
Do you wish to update/create links in the 2015 directory? (y/n) y
Do you wish to install MPICH? (y/n) y
Do you wish to generate license keys? (y/n) n
Do you want the files in the install directory to be read-only? (y/n) n
```

The license file is on the system at `/usr/local/packages6/compilers/pgi/license.dat` and is a 5 seat network license. Licenses are only used at compile time.

Extra install steps

Unlike gcc, the PGI Compilers do not recognise the environment variable LIBRARY_PATH which is used by a lot of installers to specify the locations of libraries at compile time. This is fixed by creating a `siterc` file at `/usr/local/packages6/compilers/pgi/linux86-64/15.7/bin/siterc` with the following contents

```
# get the value of the environment variable LIBRARY_PATH
variable LIBRARY_PATH is environment(LD_LIBRARY_PATH);

# split this value at colons, separate by -L, prepend 1st one by -L
variable library_path is
default($if($LIBRARY_PATH,-L$replace($LIBRARY_PATH,":", -L)));

# add the -L arguments to the link line
append LDLIBARGS=$library_path;
```

At the time of writing (August 2015), this is documented on PGI's website at https://www.pgroup.com/support/link.htm#lib_path_ldflags

Modulefile

The PGI compiler installer creates a suitable modulefile that's configured to our system. It puts it at `/usr/local/packages6/compilers/pgi/modulefiles/pgi64/15.7` so all that is required is to copy this to where we keep modules at `/usr/local/modulefiles/compilers/pgi/15.7`

2.4 MPI

2.4.1 OpenMPI (gcc version)

OpenMPI (gcc version)

Latest Version 1.10.0
Dependancies gcc
URL http://www.open-mpi.org/

The Open MPI Project is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers.

Module files

The latest version is made available using

```
module load mpi/gcc/openmpi
```

Alternatively, you can load a specific version using one of

```
module load mpi/gcc/openmpi/1.10.0
module load mpi/gcc/openmpi/1.8.8
module load mpi/gcc/openmpi/1.8.3
module load mpi/gcc/openmpi/1.6.4
module load mpi/gcc/openmpi/1.4.4
module load mpi/gcc/openmpi/1.4.3
```

Installation notes

These are primarily for administrators of the system.

Version 1.10.0

Compiled using gcc 4.4.7.

- install_openMPI_1.10.0.sh Downloads, compiles and installs OpenMPI 1.10.0 using the system gcc.
- Modulefile 1.10 located on the system at `/usr/local/modulefiles/mpi/gcc/openmpi/1.10.0`

Version 1.8.8

Compiled using gcc 4.4.7.

- install_openMPI.sh Downloads, compiles and installs OpenMPI 1.8.8 using the system gcc.
- Modulefile located on the system at `/usr/local/modulefiles/mpi/gcc/openmpi/1.8.8`

2.4.2 OpenMPI (PGI version)

OpenMPI (PGI version)

Latest Version 1.8.8
Support Level FULL
Dependancies PGI
URL http://www.open-mpi.org/

The Open MPI Project is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers.

These versions of OpenMPI make use of the PGI compiler suite.

Module files

The latest version is made available using

```
module load mpi/pgi/openmpi
```

Alternatively, you can load a specific version using one of

```
module load mpi/pgi/openmpi/1.8.8
module load mpi/pgi/openmpi/1.6.4
```

Installation notes

These are primarily for administrators of the system.

Version 1.8.8

Compiled using PGI 15.7.

- install_openMPI.sh Downloads, compiles and installs OpenMPI 1.8.8 using v15.7 of the PGI Compiler.
- Modulefile located on the system at `/usr/local/modulefiles/mpi/pgi/openmpi/1.8.8`

Installation notes for older versions are not available.

2.5 Modules

In general the software available on iceberg is loaded and unloaded via the use of the modules system [4].

Modules make it easy for us to install many versions of different applications, compilers and libraries side by side and allow users to setup the computing environment to include exactly what they need.

Important Note: Modules are not available on the login node. You must move to a worker node using either `qrsh` or `qsh` before any of the following commands will work.

Available modules can be listed using the following command:

```
module avail
```

[4] http://modules.sourceforge.net/

Modules have names that look like `apps/python/2.7`. To load this module, you'd do:

```
module load apps/python/2.7
```

You can unload this module with:

```
module unload apps/python/2.7
```

It is possible to load multiple modules at once, to create your own environment with just the software you need. For example, perhaps you want to use version 4.8.2 of the gcc compiler along with MATLAB 2014a

```
module load compilers/gcc/4.8.2
module load apps/matlab/2014a
```

Confirm that you have loaded these modules wih

```
module list
```

Remove the MATLAB module with

```
module unload apps/matlab/2014a
```

Remove all modules to return to the base environment

```
module purge
```

2.5.1 Module Command Reference

Here is a list of the most useful module commands. For full details, type `man module` at an iceberg command prompt.

- `module list` – lists currently loaded modules
- `module avail` – lists all available modules
- `module load modulename` – loads module modulename
- `module unload modulename` – unloads module modulename
- `module switch oldmodulename newmodulename` – switches between two modules
- `module initadd modulename` – run this command once to automatically ensure that a module is loaded when you log in. (It creates a .modules file in your home dir which acts as your personal configuration.)
- `module show modulename` - Shows how loading modulename will affect your environment
- `module purge` – unload all modules
- `module help modulename` – may show longer description of the module if present in the modulefile
- `man module` – detailed explanation of the above commands and others

CHAPTER

THREE

GPU COMPUTING

Graphics Processing Units (GPUs) were, as the name suggests, originally designed for the efficient processing of graphics. Over time, they were developed into systems that were capable of performing general purpose computing which is why some people refer to modern GPUs as GP-GPUs (General Purpose Graphical Processing Units).

Graphics processing tpically involves relatively simple computations that need to applied to millions of on-screen pixels in parallel. As such, GPUs tend to be very quick and efficient at computing certain types of parallel workloads.

The GPUComputing@Sheffield website aims to facilitate the use of GPU computing within University of Sheffield research by providing resources for training and purchasing of equipment as well as providing a network of GPU users and research projects within the University.

3.1 GPU Community and NVIDIA Research Centre

The University of Sheffield has been officially affiliated with NVIDIA since 2011 as an NVIDIA CUDA Research Centre. As such NVIDIA offer us some benefits as a research institution including discounts on hardware, technical liaisons, online training and priority seed hardware for new GPU architectures. For first access to hardware, training and details of upcoming events, discussions and help please join the GPUComputing google group.

3.2 GPU Hardware on iceberg

Iceberg currently contains 16 GPU units:

- 8 Nvidia Tesla Kepler K40M GPU units. Each unit contains 2880 CUDA cores, 12GB of memory and is capable of up to 1.43 Teraflops of double precision compute power.
- 8 Nvidia Tesla Fermi M2070 GPU units. Each unit contains 448 CUDA cores, 6GB of memory and is capable of up to 515 Gigaflops of double precision compute power.

3.3 Requesting access to GPU facilities

In order to ensure that the nodes hosting the GPUs run only GPU related tasks, we have defined a special project-group for accessing these nodes. If you wish to take advantage of the GPU processors, please contact research-it@sheffield.ac.uk asking to join the GPU project group.

Any iceberg user can apply to join this group. However, because our GPU resources are limited we will need to discuss the needs of the user and obtain consent from the project leader before allowing usage.

3.4 Interactive use of the GPUs

Once you are included in the GPU project group you may start using the GPU enabled nodes interactively by typing

```
qsh -l gpu=1 -P gpu
```

the `gpu=` parameter determines how many GPUs you are requesting. Currently, the maximum number of GPUs allowed per job is set to 4, i.e. you cannot exceed `gpu=4`. Most jobs will only make use of one GPU.

If your job requires selecting the type of GPU hardware, one of the following two optional parameters can be used to make that choice

```
-l gpu_arch=nvidia-m2070
-l gpu_arch=nvidia-k40m
```

Interactive sessions provide you with 2 Gigabytes of CPU RAM by default which is significantly less than the amount of GPU RAM available. This can lead to issues where your session has insufficient CPU RAM to transfer data to and from the GPU. As such, it is recommended that you request enough CPU memory to communicate properly with the GPU

```
-l gpu_arch=nvidia-m2070 -l mem=7G
-l gpu_arch=nvidia-k40m -l mem=13G
```

The above will give you 1Gb more CPU RAM than GPU RAM for each of the respective GPU architectures.

3.5 Compiling on the GPU using the NVIDIA Compiler

To compile GPU code using the NVIDA compiler, nvcc, first start an *interactive GPU session*. Next, you need to set up the compiler environment via one of the following module statements

```
module load libs/cuda/3.2.16
module load libs/cuda/4.0.17
module load libs/cuda/6.5.14
```

depending on the version of CUDA you intend to use. This makes the `nvcc` CUDA compiler available. For example

```
nvcc filename.cu -arch sm_20
```

will compile the CUDA program contained in the file `filename.cu`. The `-arch` flag above signifies the compute capability of the intended GPU hardware, which is 20 for our current GPU modules. If you are intending to generate code for older architectures you may have to specify `sm_10` or `sm_13` for example.

3.6 Compiling on the GPU using the PGI Compiler

The PGI Compilers are a set of commercial Fortran,C and C++ compilers from the Portland Group. To make use of them, first start an *interactive GPU session* and run one of the following module commands, depending on which version of the compilers you wish to use

```
module load compilers/pgi/13.1
module load compilers/pgi/14.4
```

The PGI compilers have several features that make them interesting to users of GPU hardware:-

3.6.1 OpenACC Directives

OpenACC is a relatively new way of programming GPUs that can be significantly simpler to use than low-level language extensions such as CUDA or OpenCL. From the OpenACC website :

> The OpenACC Application Program Interface describes a collection of compiler directives to specify loops and regions of code in standard C, C++ and Fortran to be offloaded from a host CPU to an attached accelerator. OpenACC is designed for portability across operating systems, host CPUs, and a wide range of accelerators, including APUs, GPUs, and many-core coprocessors.
>
> The directives and programming model defined in the OpenACC API document allow programmers to create high-level host+accelerator programs without the need to explicitly initialize the accelerator, manage data or program transfers between the host and accelerator, or initiate accelerator startup and shutdown.

For more details concerning OpenACC using the PGI compilers, see The PGI OpenACC website.

3.6.2 CUDA Fortran

In mid 2009, PGI and NVIDIA cooperated to develop CUDA Fortran. CUDA Fortran includes a Fortran 2003 compiler and tool chain for programming NVIDIA GPUs using Fortran.

- CUDA Fortran Programming Guide.

3.6.3 CUDA-x86

NVIDIA CUDA was developed to enable offloading computationally intensive kernels to massively parallel GPUs. Through API function calls and language extensions, CUDA gives developers explicit control over the mapping of general-purpose computational kernels to GPUs, as well as the placement and movement of data between an x86 processor and the GPU.

The PGI CUDA C/C++ compiler for x86 platforms allows developers using CUDA to compile and optimize their CUDA applications to run on x86-based workstations, servers and clusters with or without an NVIDIA GPU accelerator. When run on x86-based systems without a GPU, PGI CUDA C applications use multiple cores and the streaming SIMD (Single Instruction Multiple Data) capabilities of Intel and AMD CPUs for parallel execution.

- PGI CUDA-x86 guide.

3.7 GPU enabled Software

3.7.1 Ansys

See Issue #25 on github

3.7.2 Maple

TODO

3.7.3 MATLAB

TODO

3.7.4 NVIDIA Compiler

TODO - Link to relevant section

3.7.5 PGI Compiler

TODO - Link to releveant section

CHAPTER

FOUR

TROUBLESHOOTING

In this section, we'll discuss some tips for solving problems with iceberg. It is suggested that you work through some of the ideas here before contacting the service desk for assistance.

4.1 Frequently Asked Questions

4.1.1 I'm a new user and my password is not recognised

When you get a username on the system, the first thing you need to do is to syncronise your passwords which will set your password to be the same as your network password.

4.1.2 I can no longer log onto iceberg

If you are confident that you have no password or remote-access-client related issues but you still can not log onto iceberg you may be having problems due to exceeding your iceberg filestore quota. If you exceed your filestore quota in your /home area it is sometimes possible that crucial system files in your home directory gets truncated that effect the subsequent login process.

If this happens, you should immediately email research-it@sheffield.ac.uk and ask to be unfrozen.

4.1.3 My program, which use to work, stopped working

If a program that use to work suddenly stops working one day, it may be due to reasons such as;

- Hardware Faults
- Software Environment Changes
- Filestore Related Issues

Hardware faults:

It is very rare but not impossible to hit a hardware fault while using iceberg. Considering we have well over 100 machines operating %100 of the time, chances of one of these failing while running your job is rare but still possible. Possibilities of hardware faults effecting your jobs will increase if you are running parallel jobs. This is because the failures of the communications equipment (for MPI jobs) will also be a factor. If your job fails for some inexpicable 'or incomprehensible!' reason with hints of hardware problems it may be worth resubmitting it to see if it will run OK the next time. This is because, there is a very strong chance that your next job will be running on a different worker-node and hence using different hardware.

Software Environment Changes:

Programs rely on the underlying operating system and libraries to provide some of their functionality as well as prepare the environment for task execution. If the underlying operating system is subjected to any change 'such as caused by operating system, software, library or package updates, there is a risk of it effecting the users' programs.

If your code is written with 'portability in mind' in a standard, supported language such as Fortran or C, you will be less effected by such changes as recompiling your program will usually fix the problem. However, when working with a source-code, you will not be immune from problems arising from updates to the compilers.

From time to time we update our suite of compilers and inform the users via iceberg news It is then advisable to recompile your programs to generate new executables.

Filestore Related Issues:

Surprisingly this is one of the common causes of failure. By default all users have a filestore limit of 10 GBytes in their /home areas and 100 GBytes in their /data areas. Further to this there is also /fastdata area which has no quota imposed on it but the files that are left untouched for two months gets deleted and can not be recovered.

If you have a program or job that is creating large files, you must ensure that you have sufficient free filestore available before you start running it. While your job is running, if your filestore gets filled and hits the quota limit, the job will terminate abruptly and unpleasantly, leaving some of the output files corrupt, truncated and unusable.

As /data area is much bigger than /home area, we strongly recommend working in your /data area while running jobs that produce large output files.

4.1.4 I can not log into iceberg via the applications portal

Most of the time such problems arise due to due to JAVA version issues. As JAVA updates are released regularly, these problems are usually caused by the changes to the JAVA plug-in for the browser. Follow the trouble-shooting link from the iceberg browser-access page to resolve these problems. There is also a link on that page to test the functionality of your java plug-in. It can also help to try a different browser to see if it makes any difference. All failing, you may have to fall back to one of the non-browser access methods.

4.1.5 My batch job terminates without any messages or warnings

When a batch job that is initiated by using the qsub command or runfluent, runansys or runabaqus commands, it gets allocated specific amount of virtual memory and real-time. If a job exceeds either of these memory or time limits it gets terminated immediately and usually without any warning messages.

It is therefore important to estimate the amount of memory and time that is needed to run your job to completion and specify it at the time of submitting the job to the batch queue.

Please refer to the section on hitting-limits and estimating-resources for information on how to avoid these problems.

4.1.6 Exceeding your disk space quota

Each user of the system has a fixed amount of disk space available in their home directory. If you exceed this quota, various problems can emerge such as an inability to launch applications and run jobs. To see if you have exceeded your disk space quota, run the quota command:

```
quota

Size  Used Avail Use% Mounted on
10.1G  5.1G     0 100% /home/foo11b
100G     0   100G   0% /data/foo11b
```

In the above, you can see that the quota was set to 10.1 gigabytes and all of this is in use. Any jobs submitted by this user will likely result in an Eqw status. The recommended action is for the user to delete enough files to allow normal work to continue.

Sometimes, it is not possible to log-in to the system because of a full quota, in which case you need to contact research-it@sheffield.ac.uk and ask to the unfrozen.

4.1.7 I am getting warning messages and warning emails from my batch jobs about insufficient memory!

There are two types of memory resources that can be requested when submitting batch jobs using the qsub command. These are, virtual memory (-l mem=nnn) and real memory (-l rmem=nnn). Virtual memory limit specified should always be greater than equal to the real memory limit specification.

If a job exceeds its virtual memory resource it gets terminated. However if a job exceeds its real memory resource it does not get terminated but an email message is sent to the user asking him to specify a larger rmem= parameter the next time, so that the job can run more efficiently.

4.1.8 What is rmem (real_memory) and mem (virtual_memory)

Running a program always involves loading the program instructions and also its data i.e. all variables and arrays that it uses into the computers "RAM" memory. A program's entire instructions and its entire data, along with any dynamic link libraries it may use, defines the VIRTUAL STORAGE requirements of that program. If we did not have clever operating systems we would need as much physical memory (RAM) as the virtual-storage requirements of that program. However, operating systems are clever enough to deal with situations where we have insufficient REAL MEMORY to load all the program instructions and data into the available Real Memory (i.e. RAM) . This technique works because hardly any program needs to access all its instructions and its data simultaneously. Therefore the operating system loads into RAM only those bits of the instructions and data that are needed by the program at a given instance. This is called PAGING and it involves copying bits of the programs instructions and data to/from hard-disk to RAM as they are needed.

If the REAL MEMORY (i.e. RAM) allocated to a job is much smaller than the entire memory requirements of a job (i.e. VIRTUAL MEMORY) then there will be excessive need for 'paging' that will slow the execution of the program considerably due to the relatively slow speeds of transferring information to/from the disk into RAM.

On the other hand if the Real Memory (RAM) allocated to a job is larger than the Virtual Memory requirement of that job then it will result in waste of RAM resources which will be idle duration of that job.

It is therefore crucial to strike a fine balance between the VIRTUAL MEMORY (i.e. mem) and the PHYSICAL MEMORY (i.e. rmem) allocated to a job. Virtual memory limit defined by the -l mem parameter defines the maximum amount of virtual-memory your job will be allowed to use. If your job's virtual memory requirements exceed this limit during its execution your job will be killed immediately. Real memory limit defined by the -l rmem parameter defines the amount of RAM that will be allocated to your job.

The way we have configured SGE, if your job starts paging excessively your job is not killed but you receive warning messages to increase the RAM allocated to your job next time by means of the rmem parameter.

It is important to make sure that your -l mem value is always greater than your -l rmem value so as not to waste the valuable RAM resources as mentioned earlier.

4.1.9 Insufficent memory in an interactive session

By default, an interactive session provides you with 2 Gigabytes of RAM (sometimes called real memory) and 6 Gigabytes of Virtual Memory. You can request more than this when running your `qsh` or `qrsh` command

```
qsh -l mem=64G   -l rmem=8G
```

This asks for 64 Gigabytes of Virtual Memory and 8 Gigabytes of RAM (real memory). Note that you should

- not specify more than 768 Gigabytes of virtual memory (mem)
- not specify more than 256 GB of RAM (real memory) (rmem)

4.1.10 Windows-style line endings

If you prepare text files such as your job submission script on a Windows machine, you may find that they do not work as intended on the system. A very common example is when a job immediately goes into `Eqw` status after you have submitted it.

The reason for this behaviour is that Windows and Unix machines have different conventions for specifying 'end of line' in text files. Windows uses the control characters for 'carriage return' followed by 'linefeed', `\r\n`, whereas Unix uses just 'linefeed' `\n`.

The practical upshot of this is that a script prepared in Windows using Notepad looking like this

```
#!/bin/bash
echo 'hello world'
```

will look like the following to programs on a Unix system

```
#!/bin/bash\r
echo 'hello world'\r
```

If you suspect that this is affecting your jobs, run the following command on the system

```
dos2unix your_files_filename
```

4.1.11 error: no DISPLAY variable found with interactive job

If you receive the error message

```
error: no DISPLAY variable found with interactive job
```

the most likely cause is that you forgot the -X switch when you logged into iceberg. That is, you might have typed

```
ssh username@iceberg.sheffield.ac.uk
```

instead of

```
ssh -X username@iceberg.sheffield.ac.uk
```

4.1.12 Problems connecting with WinSCP

Some users have reported issues while connetcing to the system using WinSCP, usually when working from home with a poor connection and when accessing folders with large numbers of files.

In these instances, turning off `Optimize Connection Buffer Size` in WinSCP can help:

- In WinSCP, goto the settings for the site (ie. from the menu `Session->Sites->SiteManager`)
- From the `Site Manager` dialog click on the selected session and click edit button
- Click the advanced button
- The Advanced Site Settings dialog opens.
- Click on connection
- Untick the box which says `Optimize Connection Buffer Size`

4.1.13 Login Nodes RSA Fingerprint

The RSA key fingerprint for Iceberg's login nodes is "de:72:72:e5:5b:fa:0f:96:03:d8:72:9f:02:d6:1d:fd".

CHAPTER

FIVE

CLUSTER SPECIFICATIONS

5.1 Summary of iceberg hardware specs.

- Total CPUs: 3440 cores
- Total GPUs: 16 units
- Total Memory: 31.8 TBytes
- Permanent Filestore: 45 TBytes
- Temporary Filestore: 260 TBytes
- Physical size: 8 Racks
- Maximum Power Consumption: 83.7 KWatts
- All nodes are connected via fast infiniband.

For reliability, there are two iceberg head-nodes 'for loging in' configured to take over from each other in case of hardware failure.

5.2 Worker Nodes CPU Specifications

Intel Ivybridge based nodes

- 92 nodes, each with 16 cores and 64 GB of total memory (i.e. 4 GB per core).
- 4 nodes, each with 16 cores and 256 GB of total memory (i.e. 16GB per core).
- Each node uses2 of Intel E5 2650V2 8-core processors (hence 2*8=16 cores).
- Scratch space on local disk of on each node is 400 GB

Intel Westmere based nodes

- 103 nodes, each with 12 cores and 24 GB of total memory (i.e. 2 GB per core)
- 4 nodes with 12 cores and 48 GB of total memory (i.e. 4GB per core)
- Each node uses 2 of Intel X5650 6-core processors (hence 2*6=12 cores)

5.3 GPU Units Specifications

8 Nvidia Tesla Kepler K40Ms GPU units

- Each GPU unit contains 2880 thread processor cores
- Each GPU unit has 12GB of GDR memory. Hence total GPU memory is 8*12=96 GB
- Each GPU unit is capable of about 4.3TFlop of single precision floating point performance, or 1.4TFlops at double precision.
- Hencemaximum GPU processing power of is 11.2 TFlops in total.

8 Nvidia Tesla Fermi M2070s GPU units

- Each GPU unit contains 448 thread processor cores
- Each GPU unit contains 6GB of GDR memory. Hence total GPU memory is 8*6=48 GB
- Each GPU unit is capable of about 1TFlop of single precision floating point performance, or 0.5TFlops at double precision.
- Hence maximum GPU processing power is 8 TFlops in total.

5.4 Filestore

- 45 TBytes NFS mounted filestoreproviding users with storage on /home and /data areas
- 260 TBytes Infiniband connected parallel filestore providing storage on /fastdata area

5.5 Filestore Allocations

By default users get;

- 5 GBytes of storage on their /home area
- 50 GBytes of storage on their /data area.
- Currently we set no limits on the /fastdata area but the files that have not been modified for 3 months will get deleted.

From time to time we shall review our data storage and allocation policies depending on usage and inform our users.

It is strongly recommended that anyone wishing to use the /fastdata area creates a subdirectory with the same name as their username for storing their data.

The /fastdata area is faster to access from the new (INTEL-based) nodes where the new infiniband connections are in use.

5.6 Filestore recovery and backup policies

If you do loose files by accidental deletion, over-writing etc. and you wish us to recover them, do let us know as soon as possible to increase the chances of successful recovery.

- Users' /home areas are fully backed up to allow recovery of lost data.
- /data and /fastdata areas are not backed up, however ...
- Due to mirroring it will usually be possible to recover lost or deleted files from the /data areas, providing we are informed quickly after such an incident.
- It is not possible to recover lost and/or deleted files from the /fastdata areas.

5.7 Software and Operating System

Users normally log into a head node and then use one (or more) of the worker nodes to run their jobs on. Scheduling of users' jobs on the worker nodes are managed by the 'Sun Grid Engine' software. Jobs can be run interactively (qsh) or submitted as batch jobs (qsub).

- The operating system is 64-bit Scientific Linux "which is Redhat based" on all nodes
- The Sun Grid Engine for batch and interactive job scheduling
- Many Applications, Compilers, Libraries and Parallel Processing Tools. See Section on Software.

CHAPTER

SIX

GLOSSARY OF TERMS

The worlds of scientific and high performance computing are full of technical jargon that can seem daunting to newcomers. This glossary attempts to explain some of the most common terms as quickly as possible.

- **HPC** - 'High Performance Computing'. The exact definition of this term is sometimes a subject of great debate but for the purposes of our services you can think of it as 'Anything that requires more resources than a typical high-spec desktop or laptop PC'.
- **GPU** - Acronymn for Graphics Processing Unit.
- **SGE** - Acronymn for Sun Grid Engine.
- **Sun Grid Engine** - The Sun Grid Engine is the software that controls and allocates resources on the system. There are many variants of 'Sun Grid Engine' in use worldwide and the variant in use at Sheffield is the Son of Grid Engine
- **Wallclock time** - The actual time taken to complete a job as measured by a clock on the wall or your wristwatch.