Cranfield University
Astral HPC Cluster

User Guide

Issue 1.8

Andy Gittings
IT Department

February 2014

Introduction

This document is designed to give a new Astral user information on how to access and use the new system.

Acknowledgements

I’d like to thank all the users that beta tested the system and helped test and develop the examples in this document, in particular: Daniel Cozzolino, Domenico DiCugno, Rhodri Jones, Robert Sawko, Apostolos Spanelis & Ben Thornber.

System Details

Login Nodes

There are two “front-end” login nodes. These are known as hpclogin-1 and hpclogin-2. Internally to the cluster they are also known as “service1” and “service2” so please do not be alarmed if you see this hostname on your command prompt when logging in.

They each contain two Intel E5-2660 (Sandy Bridge) CPUs giving 16 CPU cores and have a total of 192GB of shared memory.

We do not encourage running of MPI code on these nodes except for debugging prior to submission to the batch system.

The cluster login nodes authenticate using the University Windows authentication credentials. The login nodes are running SuSE Enterprise 11 SP2 (Linux Kernel version 3.0.101).

Compute Nodes

The compute nodes are housed in standard rack mount 2U chassis, with four nodes per 2U chassis. There are 20 chassis in total spread over four racks. This gives a total of 80 compute nodes. Each node has two Intel E5-2660 (Sandy Bridge) CPUs giving 16 CPU cores and 72 nodes have 64GB shared memory the remaining 8 nodes have 128GB of shared memory. Taken together this gives a total of 1280 available cores. The nodes are named n001 through n080. The login and compute nodes are connected with Infiniband low-latency interconnect.

Storage

The login and compute nodes all have access to a Panasas enterprise storage system, which is providing a total of 34TB of usable storage. Each user has a home folder /panfs/storage/home/$USER – where $USER is your normal campus login ID. This home folder is designed only to hold minimal data (eg. Login profile information, ssh keys). It is has a quota set of 1GB, and can be referred to by the environment variable $H.

Additionally each user has a scratch folder – which is where data for job runs must be placed. The full path to this storage is /panfs/storage/$USER – however a shorter link of /scratch/$USER has been created for convenience, this can also be referred to by the environment variable $W.

Your scratch folder has an initial quota of 75GB this can be temporarily increased upon request.

IMPORTANT:

You must not use Astral storage for long-term storage of data, due to its high performance it is an expensive storage medium and it is NOT backed-up. It is your responsibility to move valuable files from both scratch and home folders to locations where they are secured by backup. This could be your University Z: drive or some local storage device.

Compilers & MPI

The system uses the Intel 2013 XE Cluster Suite for compilation (FORTRAN, C, C++) and MPI.

MPI wrappers are provided for compilation of code so you should have little difficulty recompiling your code for this system.

Performance

The system performance is measured using HPL (High Performance Linpack). The theoretical peak processor performance (R_{peak}) is 22.5 TFlops. The measured maximum performance achieved (R_{max}) is 18.8 TFlops.

This compares with Astral-1's R_{peak} of 10.2 TFlops and R_{max} of 7.2 TFlops.

Queuing System

The system incorporates a single Job Scheduling system, Altair PBS Pro and PBS Analytics to allow administrative analysis of system usage.

Using the Astral System

Logging in to front-end nodes

If you have never had an Astral account you will need to complete an account request form – available from the IT department's HPC pages on the University Intranet.

To connect, ssh –X (from a Linux machine) or PuTTY from a Windows machine to:

`hpclogin-1.central.cranfield.ac.uk`

or:

`hpclogin-2.central.cranfield.ac.uk`

You will use your standard University username and password to log in.

Please note: If you are logging in from off-campus, you will first need to ssh to the SSH gateway - **`hpcgate.cranfield.ac.uk`** then at the command line type the command "**`astral`**" to connect to the Astral service.

Processes run on the login nodes by users are limited to 100 minutes, so please do not use these nodes for running long solver processes.

To change directory to your scratch space, use:

cd $W

To return to your home folder:

cd $H

Do not use these directives from within PBS submission scripts.

Setting Your Environment

The Astral service uses the **module** environment to allow the easy setting of environment variables such as PATH and the Licensing environment for applications.

To begin with you should create a .bash_profile file in your home folder and add to this any modules that you intend to work with interactively from the login nodes.

A list of available modules can be obtained by typing: **`module avail`** at the command line.

For example a user that would use ANSYS CFX interactively on the login nodes would add the following line to their .bash_profile:

module load cfx/14.5

You can always check which modules are loaded by typing: **`module list`** at the command line.

We would strongly recommend request that you do not place **`module load`** directives in your .bash_profile for batch runs.
Instead these should be placed inside the submission script itself – see the example templates listed in appendix A for examples of this.

Running MPI Code

Running MPI code directly on the login nodes, is as stated previously, discouraged. However short runs to test code can be performed by loading the environment as follows:

```
module load impi
. mpivars.sh
```

(N.B. the second line is: dot space mpivars.sh)

You can then use mpirun in the normal way to execute your code:

```
mpirun –n 16 <myexec>
```

Where <myexec> is the name of your MPI executable.

Submitting batch jobs to compute nodes

In order to submit a job to the batch system, you need to use a submission script. We have supplied templates for common applications this are listed Appendix A.

You will need to modify these scripts to use your data files, include your own email address (for notifications) and to change the number of CPU resources required.

Once you have a script, ensure that this is placed in a folder within your /scratch/$USER area with all the data required by the application.

Assuming the submission script is called myjob.sub (it can be called anything you wish) to submit you would use the **`qsub`** command:

```
qsub myjob.sub
```

The system will respond with:

```
NNNN.service1
```

Where NNNN is the job number.

Examining a running batch job's output file

Specified in the PBS submission script is the line:

```
#PBS –k oe
```

This ensures that both the standard output and standard error from the application are placed in your home folder (i.e. /panfs/storage/home/$USER). It is important to keep your home folder tidy, as if you reach your quota these output files will not be created/updated and the job will fail.

They will take the form:

```
Jobname.oNNNN
Jobname.eNNNN
```

Where Jobname is the name you gave to the job in the submission script, and NNNN is the job number as given by the scheduler.

So, for example to check the standard output of a job called "myjob" which was submitted as job number 402 you can use:

```
cat ~/myjob.o402
```

Listing the scheduler queue

To check the job queue you should use **`qstat:`**

Command	**Description**
`qstat`	Shows the status of all PBS jobs. The time displayed is the ***CPU time*** used by the job.
`qstat -s`	Shows the status of all PBS jobs. The time displayed is the ***wall-time*** used by the job.
`qstat -u userid`	Shows the status all PBS jobs submitted by the user ***userid***. The time displayed is the *wall-time* used by the job.
`qstat -n`	Shows the status all PBS jobs along with a list of compute nodes that the job is running on.
`qstat -f job number`	Shows detailed information about the job specified by ***job number***

A job can be in one of the following states (as given by the output from qstat):

State	**Meaning**
Q	The job is ***queued*** and is waiting to start.
R	The job is currently ***running***.
E	The job is currently ***ending***.
H	The job has a user or system ***hold*** on it and will not be eligible to run until the ***hold*** is removed

Removing a job from the queue

You can remove a job from the PBS queue by using the **`qdel`** command:

```
qdel NNNN
```

Where NNNN is the job number - you can only remove your own jobs – only the super-user can remove other users' jobs.

Checking disk space

To display your used disk space (against quota) type:

```
getquota
```

This will output information of the form:

```
---------------------------------------------------------------------------

Scratch space used: 65.62 GB out of 75.00 GB
Home space used: 652.58 MB out of 1.00 GB.

---------------------------------------------------------------------------
```

The quota values from this command are only updated every 30 minutes, however the system updates the internal quota allocations instantly.

Displaying Node Utilisation

For compatibility with Astral-1 we have engineered a new version of the sinfo command to show the node status in an easy to read form.

```
sinfo
```

Will output:

```
ASTRAL - PBS Node status
========================

Down: ## [2]
Busy: ###################################### [38]
Free: ####################################### [39]

Note: 2 nodes will always show as down.
```

About the scheduler

Our queues are based upon wall-time – i.e. the real time we expect the job to take to run. The scheduler takes this into account when scheduling jobs.

We are using a fair-share policy to ensure that users gain a fair access to the system (ie. A user that has used more CPU time will have a lower priority than a user that has used less CPU time).

However, the system is also configured to use "backfill", so you may notice shorter low priority jobs being run before your higher priority job – please do not be concerned, PBS is essentially filling in the resource gaps with short jobs whilst it is waiting for resources to become available for your larger job. PBS's ability to look into the future is purely based on the specified wall-clock time (in our case: queue specification) – so obviously, this only works efficiently if users specify the correct queue for their jobs.

We have the ability to analyse historical data on requested wall-time (i.e. queue) and actual time taken, so we may contact users that repeatedly over estimate their usage.

The express queue (Astral-1/GRID) has now changed to **`day_express`** – where a two hour job will only be run during office hours – such that the results are more likely to be used. If you need a queue for out-of-hours use – please use the **`short`** queue instead.

During busy periods users may be limited to having 4 jobs *running* simultaneously. Users can obviously have more than 4 jobs in the pending queue.

Using the system efficiently

Where possible use multiples of 16 CPUs (as each node has 16 cores). If you need less than 16 cores consider using the Grid which currently has 8 cores per node.

Size your jobs to the system's physical memory. Please remember that there is a finite amount of shared memory per node (typically 64GB – or 4GB per core). If your process exceeds this it will start to use virtual memory and your process will slow down considerably, this can also cause a node to fail.

If you need more memory per core, consider reducing the ncpus (processors per node – default and maximum 16) value in your submission script, such that the 64GB of memory will be shared between less CPUs. See appendix D for more information.

Appendix A – Template PBS Scripts

These scripts are also available in electronic form in:
`/usr/local/commercial/pbs`

You will need to edit these scripts to refer to your data files and email address.

Please ensure that if you edit using a Windows editor, you run the `dos2unix` command on the file before submission using `qsub`. If you do not do this your job will **not** run.

Application	**PBS submission Template filename (in /usr/local/commercial/pbs)**
FLUENT	fluent.sub
Abaqus	abaqus.sub
STARCCM+	starccm.sub
LS-DYNA	lsdyna.sub
CFX	cfx.sub
MATLAB	matlab.sub
OpenFOAM	openfoam.sub
MPI	mpi.sub

For example to copy a FLUENT submission template to the current directory:

```
cp /usr/local/commercial/pbs/fluent.sub .
```

[N.B. there must be a space between the filename “fluent.sub” and the “.” at the end of the line.]

Appendix B – Building MPI Code

Basic FORTRAN MPI build environment

At the command line (or add to your ~/.bash_profile)

```
module load ifort
module load impi
. mpivars.sh
. ifortvars.sh intel64

export I_MPI_F77=ifort
export I_MPI_F90=ifort
```

You can now compile an MPI file using `mpif90` or `mpif77`:

```
mpif90 –o hellompi hellompi.f90
```

And test using `mpirun`:

```
mpirun –n 16 ./hellompi
```

Basic C/C++ MPI build environment

At the command line (or add to your ~/.bash_profile)

```
module load icc
module load impi
. mpivars.sh
. iccvars.sh intel64

export I_MPI_CC=icc
export I_MPI_CXX=icpc
```

You can now compile an MPI file using `mpicc` or for C++ `mpicxx`:

```
mpicc –o hellompi hellompi.c
```

And test using `mpirun`:

```
mpirun –n 16 ./hellompi
```

N.B. Adding the –xavx option to the compiler will optimise the core to use the Intel AVX instruction set which may increase performance.

Appendix C – PBS Queues

Queue Name	Description (wall-clock time)
quick	30 minute queue for debugging
express	2 hours at any time
day_express	Default queue: 2 hours during working day
short	8 hours
medium	24 hours
interactive	24 hours
long	72 hours
very_long	120 hours
extra_long	240 hours
special	Indefinite – by special arrangement only
low_priority	Only run when the cluster is underutilised specific users only

Appendix D – Specifying Resource Requirements

“Bigmem” nodes

There are eight nodes (n001 through n008) which have twice the amount of memory per node as the rest of the cluster nodes. These nodes have 128GB (or 8GB per cpu core).

These nodes will ordinarily be the last nodes used for normal jobs, however can be specified as a resource requirement in the submission script. You can achieve this by adding the directive **`:bigmem=True`** to the **`select`** statement in your submission script:

```
#PBS -l select=2:ncpus=16:mpiprocs=16:bigmem=True
```

Specifying CPU requirements

The template examples in appendix A all have the following PBS directive:

```
#PBS -l select=2:ncpus=16:mpiprocs=16
```

What this means is “I would like two chunks of 16 cores”. PBS has the concept of “chunks” an atomic resource limit. In our case on Astral2 a CPU chunk has a maximum of 16 CPUs, so **1:ncpus=32** would **not** work as no node has 32 CPUs.

We could also ask for **4:ncpus=8**, **8:ncpus=4** etc… each would give the same 32 CPU result and work with Astral’s limits.

Going back to the example, we know that each node has 16 cores and 64GB of RAM, so in this example our process would use 32 CPUs, and each process would potentially have available the full 4GB of memory on the system (16 x 4GB = 64GB).

Important: Please ensure that the mpiprocs value is equal to the ncpus value for efficient use of the system.

If our job was running out of memory, we could set the same number of processes over more nodes and gain access to more RAM e.g.

```
#PBS -l select=4:ncpus=8:mpiprocs=8
#PBS -l place=exclhost
```

So now we’re asking for node exclusivity on 4 chunks of 8 cores (ie. 64 cores) but still a total of 32 processes. But, those processes will have access to twice the memory as the first example, 8GB per core (8 x 8GB = 64GB). In this case it is important to specify the “place=exclhost” otherwise PBS could place your job with other processes.

Specifying licence requirements

We have extended the PBS scheduler to allow users to specify the number of application licences that their job requires. This enables the job to be scheduled when both CPU, memory and licence requirements are met.

For ANSYS the following lines need to be added to the submission script:

```
#PBS -l cfd_base=1
#PBS -l cfd_hpc=12
```

The above example is for a 16 core job. Where cfd_base is always 1 (and includes the first 4 cores). cfd_hpc is then equal to the number of CPU cores requested minus 4.

Appendix E – File-system Information

.panfs files

If you encounter files with names of the form:
`.panfs.cc7410a.1113355902480856000` please be aware that you cannot delete them from the login nodes.

These files are created when two processes have tried to access a file in conflicting ways normally when one process has requested to delete a file which is in use by another process. The file system has renamed the open file in order to satisfy both requests.

If these files persist, please contact the IT Service Desk and we will remove them for you.

Accessing from Windows

You can use an SCP utility such as WinSCP to connect to the login nodes and transfer data.

In addition, the login nodes also export your home and scratch directories as Windows shares. To connect the path is:

\\hpclogin-2\home

\\hpclogin-2\scratch

If you are already logged in to a University machine, you will not be prompted to login again.

Useful commands

The ansyslicence command will display ANSYS licence usage. Eg.

```
# ansyslicence
Users of aa_mcad:  (Total of 505 licenses issued;  Total of 0 licenses in use)
Users of aa_r:  (Total of 300 licenses issued;  Total of 139 licenses in use)
Users of aa_ds:  (Total of 200 licenses issued;  Total of 0 licenses in use)
Users of aa_t_a:  (Total of 200 licenses issued;  Total of 0 licenses in use)
Users of aa_turbo:  (Total of 100 licenses issued;  Total of 2 licenses in use)
Users of aa_mesh:  (Total of 5 licenses issued;  Total of 1 license in use)
Users of aa_r_hpc:  (Total of 1000 licenses issued;  Total of 930 licenses in use)
```

You can also specify a username to get information about that users' licence use:

```
# ansyslicence c120613
    c120613 n076.default.domain n076 (v2011.1010) (cclic-
2.central.cranfield.ac.uk/1055 27415), start Tue 7/17 18:54
    c120613 n065.default.domain n065 (v2011.1010) (cclic-
2.central.cranfield.ac.uk/1055 9972), start Thu 7/19 12:42
    c120613 n037.default.domain n037 (v2011.1010) (cclic-
2.central.cranfield.ac.uk/1055 2976), start Thu 7/19 17:27
    c120613 n076.default.domain n076 (v2011.1010) (cclic-
2.central.cranfield.ac.uk/1055 511), start Tue 7/17 18:54, 76 licenses
    c120613 n065.default.domain n065 (v2011.1010) (cclic-
2.central.cranfield.ac.uk/1055 2118), start Thu 7/19 12:42, 60 licenses
    c120613 n037.default.domain n037 (v2011.1010) (cclic-
2.central.cranfield.ac.uk/1055 8742), start Thu 7/19 17:27, 60 licenses
```

Appendix F – Fair-share & Queuing

Fair-share Explained

In order to maintain fair access to resources, the PBS scheduler is configured to prioritise users' jobs according to previous usage levels. i.e. high-usage users' jobs have lower priority in the queue.

The way this works is as follows:

- The scheduler re-computes the entire run queue every 5 minutes (or 300 seconds). This means jobs can move up/down in the queue dynamically.

- At each computation period, the amount of CPU seconds consumed is added to each user's fair-share count. E.g. if you have a 32 CPU job running, your count will increase by (300 * 32 = 9600) every 5 minutes during the run of your job.

- Every 24 hours all users' total count is halved, such that it will decay over time.

- The fair-share count is taken into account when prioritising jobs, lower count gives a higher priority.