An Introduction to High Performance Computing

Martin Vickers
Data Manager/HPC Systems Administrator
mjv08@aber.ac.uk

Vasileios Panagiotis Lenis
Training Officer
vpl@aber.ac.uk

Contents

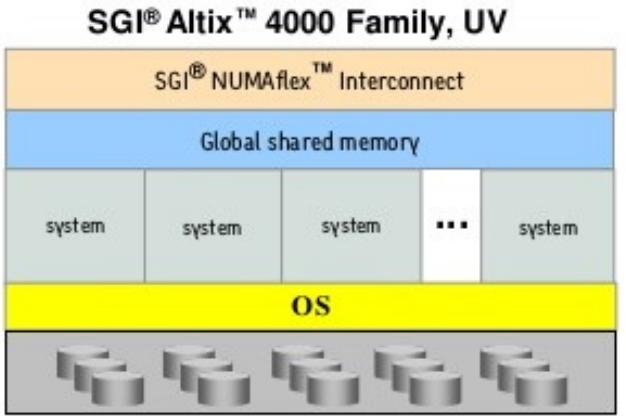
- What is HPC
- Our HPC(s)
- Submitting a job to the HPC
  - Monitoring your running job
  - When your job has finished
- Checking how the resources on the HPC are being used
- Submitting multiple jobs
- Best practice

High Performance Computing

- To tackle a problem that is computationally too 'big' for a desktop PC
  - Takes too long
  - Uses too much memory
- As a HPC user, you need to consider what you're using HPC for
  - “I have a program that needs more memory than my desktop PC has”
  - “I want to run a program on lots of different data”
  - “The program I'm running will take days/months on my desktop PC, and I want to check my email”

Types of HPC - A Supercomputer!!!

- A very large computer that links components to make it appear as one very large computer
- Custom made hardware
- Custom OS
- Very expensive

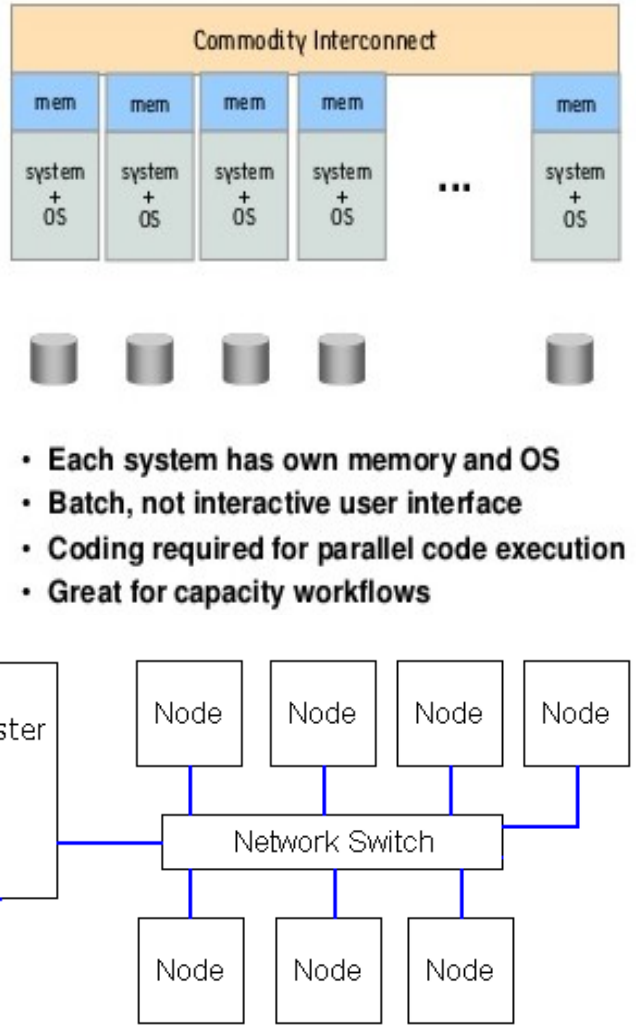


- All nodes operate on one large shared memory space
- Cache Coherency
- Eliminates data passing between nodes
- Big data sets fit entirely in memory
- Less memory per node required
- Simpler to program
- High Performance, Low Cost, Easy to Deploy

sgi

Types of HPC – Grid Computer

- AKA Beowulf Cluster
  - Named after a NASA machine
- Commodity hardware, normally identical machines, linked together using a scheduler
- Each server, “compute node”, has as many CPU cores and RAM as possible
- Each node has own OS



Bert and Ernie (IBERS)

- Login Node - Bert
- Master Node – Ernie
- Thirteen Compute Nodes
- 536 compute cores
- 3.4TB RAM
- 11TB Fast Disk (Scratch)
  - No Backup
- Two Storage Nodes (43TB combined)
  - Home and Group Dir's
  - Daily Backup
- 260TB Repository

Holly (IMAPS/IMPACS)

- Master/Login Node – Holly
- Sixteen Compute Nodes
- 144 compute cores
- 480GB RAM
- 33TB storage for everything
  - No Backup

Operating System

- Scientific Linux release 6
- An Enterprise GNU/Linux distribution
- Runs from the command line
- Behind the AberAU firewall
- Accessed using SSH
  - Windows, download PuttySSH
    - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  - GNU/Linux/Mac
    - ssh username@bert.ibers.aber.ac.uk

What is Sun Grid Engine (SGE)

A queue scheduler that allows users to submit jobs.

It deals with finding resources for your job to be run, e.g. CPU and Memory

Allows users to monitor the progress of their jobs.

SGE knows how much memory each node has and how many CPU cores, represented by 'slots'.

And a queue is?

Nodes are assigned to a queue. This means that the user submits a job to a queue for a specific resource.

Bert - Three queues

- large.q
- intel.q
- amd.q

Holly - Three queues

- interactive.q
- intel.q
- large.q

Bert & Ernie Compute Nodes

- 1xAMD node
  - 32 core
  - 512GB RAM
  - large.q
- 3xIntel nodes
  - 8 core
  - 192GB RAM
  - intel.q
- 3xAMD nodes
  - 32 core
  - 98GB RAM
  - amd.q
- 4xAMD node
  - 64 core
  - 256GB RAM
  - amd.q
- 2xAMD nodes
  - 64 core
  - 512GB RAM
  - large.q

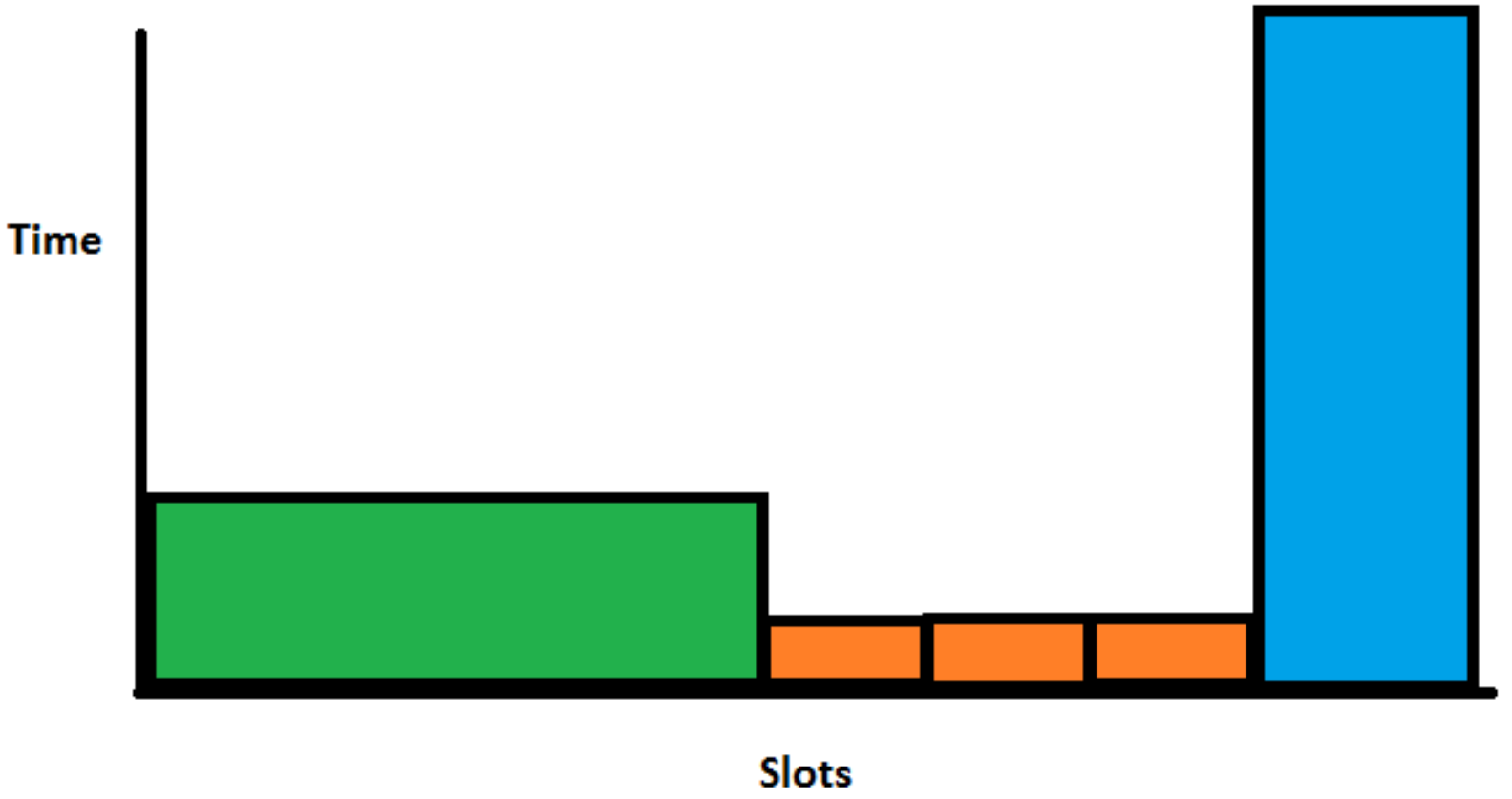
Holly Compute Nodes

- 2xIntel nodes
  - 8 core
  - 16GB RAM
  - interactive.q
- 12xIntel nodes
  - 8 core
  - 16GB RAM
  - intel.q
- 2xIntel nodes
  - 16 core
  - 128GB RAM
  - large.q

Asking for resources

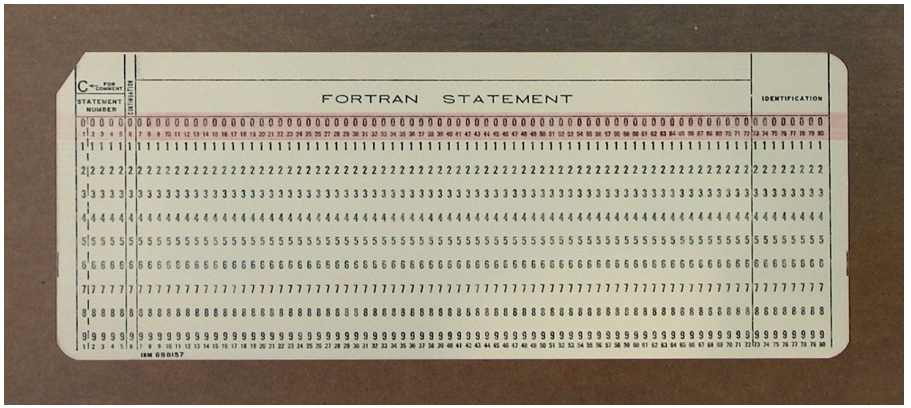
- SGE needs to know three things
  - How many CPU cores you want
  - How much Memory you want
  - How long you expect it to run for
- It cannot work this out for you, and it cannot be changed once it's in the queue.
- By telling SGE to use certain resources, doesn't mean your program understands this
- Based on what you have asked for (CPU cores, memory, length of time), what resources are available and if you're running anything else, it will try to fit your job in

A bit like Tetris



The Important concept!

- It's not immediate!
- Other people are using it
- Your queuing for resources to be available



Lets login

username@bert.ibers.aber.ac.uk

Copy the example directory to your home directory

- Location
  - /ibers/repository/public/courses/intro-HPC-course

```
[mjv08@bert]$ cd /ibers/repository/public/courses
[mjv08@bert]$ cp -r intro-HPC-course ~/
[mjv08@bert]$ cd
```

A simple script – first-script.sge

```
#specify the shell type
#$ -S /bin/sh
#run in the current working directory
#$ -cwd
#specify which queue you wish to use
#$ -q course.q
#Limit memory
#$ -l h_vmem=512M
#How long we think it will run for (2 mins)
#$ -l h_rt=00:02:00

#run a program command
sleep 60
hostname && uptime
```

Sun Grid Engine (SGE)

- You submit your job to the queue.
- When appropriate resources are available, it will run.
- Example
  - qsub first-script.sge

Your running jobs

- Display your running jobs
  - `[mjv08@bert]$ qstat`

Once your job is complete

- `qstat returns nothing`
- `*.oJOBID file`
  - stdout
- `*.eJOBID file`
  - Stderr
- qacct -j JOBID
  - Gives you information about the job that ran

I want more cores!!!

- In order to specify more cores, you need to add;
  - #$ -pe multithread 2
- NOTE:
  - Your program will probably need to be told the number of cores. $NSLOTS will do this.
- The multithread Parallel Environment (pe) requests N cores on the same node
  - Other parallel environments are available. Beyond the scope of today

blast-script.sge

```
#$ -S /bin/sh
#$ -cwd
#$ -q course.q
#$ -l h_vmem=3G
#$ -l h_rt=00:10:00
#$ -pe multithread 2
#run a program command
module load BLAST/blast-2.2.30+
blastx -db BLAST_db -query query.fa -num_alignments
10000 -outfmt 6 -out BLAST.ouput -num_threads $NSLOTS
```

Modules (1)

Software is installed in modules. This allows you to have lots of programs installed, but only available when you want them to be.

- Maintains multiple versions
- Avoids clashes in names
- Switch versions quickly (good within your work-flow)

Lets have a look….

Modules (2)

- See what modules are available
  - `module avail`
- See what modules you have already loaded
  - `module list`
- Load a particular module
  - `module load program/version`
- Remove a particular module
  - `module unload program/version`

Getting your jobs running

- Remember, the more resources you ask for, the longer it takes for those things to be scheduled
- Taking a look at what is happening on the HPC
  - Visual Representation of Load average
    - http://bert.ibers.aber.ac.uk/ganglia
  - Node view
    - `qstat -f`
  - Other peoples running jobs
    - `qstat –u “*”`
    - `qstat -u “*” | grep “ r “`

Advanced reservations

- If you want a lot more than one CPU core
  - `qsub -R y script.sge`
- Reserves slots until you get the number you want
- Needs time to be well defined
- Favours short running jobs
- Only so many reservations allowed on the system at any one time

Memory

- The hardest thing to get correct
  - Easier if you're writing your own code
  - Most are using programs others have written
  - User doesn't always have control over this
- Check memory allocation
  - `qstat -F h_vmem,mem_free`
- There is no answer to getting memory correct, it's often a function of input file
  - Sometimes just a very bad piece of software

Other SGE Commands

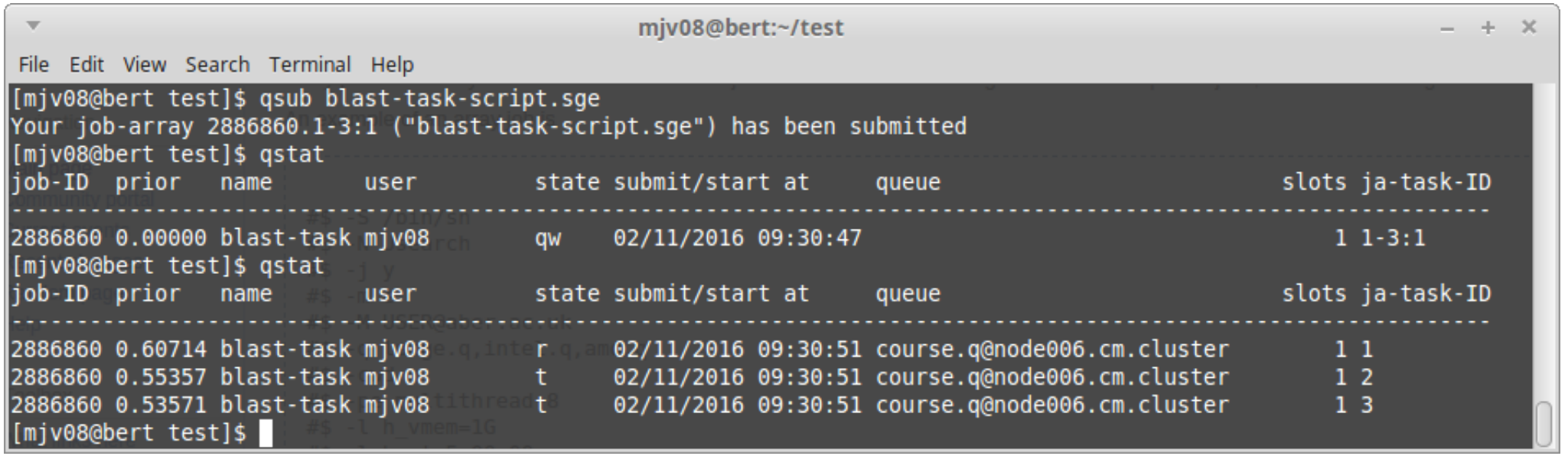
- Deleting your job from the queue or while running
  - `qdel job-ID`
- Find out what the job did after it's completed
  - `qacct -j job-ID`

Submitting 1000s of jobs

- Don't do this individually
  - Courses slow down of SGE spool
  - Difficult for everyone to read what's in the queue
  - SGE takes a long time to schedule new jobs
- Use tasks instead
  - 1 job submission, run multiple times

blast-task-script.sge

```
#$ -S /bin/sh
#$ -cwd
#$ -q course.q
#$ -l h_vmem=3G
#$ -l h_rt=00:10:00
#run on three different files
#$ -t 1-3
#run a program command
module load BLAST/blast-2.2.30+
blastx -db BLAST_db -query query_$SGE_TASK_ID.fa \
-num_alignments 10000 -outfmt 6 \
-out BLAST_$SGE_TASK_ID.ouput
```


mjv08@bert:~/test
File Edit View Search Terminal Help
[mjv08@bert test]$ qsub blast-task-script.sge
Your job-array 2886860.1-3:1 ("blast-task-script.sge") has been submitted
[mjv08@bert test]$ qstat
job-ID prior name user state submit/start at queue slots ja-task-ID
2886860 0.00000 blast-task mjv08 qw 02/11/2016 09:30:47 1 1-3:1
[mjv08@bert test]$ qstat
job-ID prior name user state submit/start at queue slots ja-task-ID
2886860 0.60714 blast-task mjv08 r 02/11/2016 09:30:51 course.q@node006.cm.cluster 1 1
2886860 0.55357 blast-task mjv08 t 02/11/2016 09:30:51 course.q@node006.cm.cluster 1 2
2886860 0.53571 blast-task mjv08 t 02/11/2016 09:30:51 course.q@node006.cm.cluster 1 3
[mjv08@bert test]$

Best Practice

- Use scratch space for temporary files
  - /ibers/ernie/scratch/USERNAME
- Don't submit lots of jobs without ensuring 1 will actually complete
- If you don't specify the resources you use, it will use the default (e.g. 1GB RAM, 144years runtime, 1 CPU core)
- Use tasks for many small jobs

Access and Support

- Read and Contribute to the wiki
  - https://bioinformatics.ibers.aber.ac.uk/wiki
- Email Me
  - mjv08@aber.ac.uk