**OBDS Day 3: 29th April HPC with Ewan Mac Mahon**

**Intro to batch systems/SGE**

**Background**

Assuming HPC means high performance computing with sun grid engine

Sun grid engine is a batch queuing system, a way of sharing cpu between users/jobs

* Can give all these things a go on the ccb cluster too, somethings between clusters differ. We are set up on the CGAT training cluster during the course.

HPC – When you need more resources than you have available locally. Things will freeze/take ages etc to run, that’s how you’d know you’d need HPC

Downsides, less convenient, less interactive than local

How does the cluster work/connected?

Separate storage nodes, master node (coordinator). 2 networks. Split between designs of cluster, depending on how computational network is set up.

CGAT cluster designed Data off the storage nodes into the compute nodes rapidly

Other clusters act to move data quickly between nodes, depends how things are set up

**Slide 8 – definitions**

Traditional machine, single CPU, all jobs happen in sequence (one thing at a time), more modern computers can run things in parallel. Divide into chunks to run at same time

Amdahl’s law – how much can be parallelised at all.

How we kind of work out how much work can be paralllelised. Ie how much time you save, efficiency

As you increase processing units, the efficiency increase slows – the increase in speedup is not linear to no.processors (depending on how much can be parallelised)

Useful to give jobs a couple more cores but any increase on that doesn’t really help speed up.

24 CPU cores per node on our cluster, rare that a job would need more cpu nodes than that.

**Slide 12 – when to parallelise?**

If you have 20 input files, easy to parallelise 20 jobs, but harder if you need to do certain parts of job sequentially, ie dependency of one job on another

SMMP – my comp will have multiple cpu, sharing same main memory but individual cache per cpu. Good communications and efficient. Worry about synchronisation – openMP way of coordinating jobs. Will only encounter this really if it’s built into a certain tool.

**Slide 14 – Message passing**

Each CPU has its own memory/RAM (multiple nodes in a compute cluster)

Each CPU/node works on its own lil bit of data, passes a message to other nodes to say yes have done the work, here’s the reply/result (communication across nodeS)

Comes down to Amdahl’s law – message passing isn’t a parallel step as have to do it between steps, if you can speed this up, reduce the %job that isn’t parallelisable

This is all very complex, not sure I understand. Don’t know if I need to

**Slide 15**

Dividing job up into different chunks, what the CGAT pipeline does.

Multiple steps, pipeline code will get the jobs where they need to go to help parallel

Staggered starts – computer will submit multiple batch jobs for you when you submit one run

**Slide 16 – batch systems**

Take load off and execute work ina batch system (ie master host)

Give it a job, leave it to the master to give it to an execution host. Takes off aspects of figuring out which jobs/nodes to run/use

Ie. Could submit 500 jobs and master would stop you monopolising the system

**Slide 18 SGE**

Deva and klyn

Q- why is it called Sun Grid Engine

1. Idea of it being like plugging your job into the mains elec source. Doesn’t work quite as big as this, not quite grid-like – still quite local.
2. Cern etc have huge big grid networks, users may have no idea where data is stored etc
3. Sun because Sun wrote it.
4. Moving to a diff batch system from SGE as Sun now doesn’t exist so SGE hasn’t been updated.

**Slide 19**

Scheduling step, goes through queues (lucky dip pool of jobs, batch system selects jobs) based on priority, how busy system is, what job needs and a scoring system (job priority – who’s job, how long it’s been waiting. CGAT does look at how long a job has been waiting and also uses a fair share algorithm, more you do, less priority you get – ie so you don’t block/monopolise the system. Drops your priority, but if no one else usingsystem then can go for it)

Less the job asks of the system, generally quicker it will run.

**Slide 20**

Good to know your jobs, ie could run interactively first, see how much memory you need then sbmit batch?

I.e run a mini interactive version, downsampling – basically check your code and the job details

**Slide 21**

Run on batch system, monitor cpu usage and monitor within code yourself

**Exercise 1**

Log into cluster, type following commands

Qhost #Overview of hardware on the system. Gives you info on different systems, numbered nodes like cgat005 arebatch system worker nodes. Dashed outs on like load and memory use – currently offline nodes (switched off) but batch knows they exist. Reason is as they’re old nodes, low mem. Cgat016 on the other han has more memtot – load column gives you info on how busy machine is on (0.05 so empty) 12g memuse

Qconf -shgrpl #Shows groups on system. Allhosts means everything. On cgat cluster, don’t get anything as the nodes aren’t grouped

Qconf -sql #show me the queues as a list.

Qconf -sgl

On CBRG, qhost gives you some nodes with 252g and big mem machines have 1000g each.

I just logged into cbrg klyn

On CBRG – can give jobs with bigger memories requirement to the bigmemq

**Slide 27 – Cluster**

Normally just want dedicated, all of the cores will be on a single node. Generally only use the dedicated

Make n mpi (mpi will try and give 4 sltos on different nodes, make will give on one node unless spillover. Gives you a list of nodes)

**Slide 28**

Is the cluster busy?

Qhost -j #lists jobs on machine

Qstat -f -u “\*” #Lists jpobs that are running n qhere they are (-f means full output) (-u means “user”, using “\*” will give all users)

Qstat #on cbrg can just use qstat as they’ve set this up as an alias

Qstat | less #this will pipe the output of qstat into less so it’s more manageable to read

**Slide 30**

Qsub pipeline.py #pipeline will submit batch jobs for you from this pipeline

Interactive sessions

Qrsh #remote shell, login from 1 machine to another but qrsh will schedule login like a batch job. Ie it will find a good node for you to do an interactive heavy R session (eg). This will be useful

Slide 31

Adaptations of qsub to define where to do the job, redirect output etc. Tbf you can add this into your bash script

Better to put this info into the bash script, but its quick n dirty to include in command (pretty sure the command will override scrpt tho)

**Slide 34**

Pe dedicated #will give a certain number of slots if you know your job needs more memory

Skipped 35 on array jobs, he’ll come back to this

**Slide 36**

Submitting this info into the script rather than the command

Lines starting in # are comments that bash will ignore (so will SGE)

But #$ are comments for bash but SGE will see them as commands!

**Exercise2 Slide 38**

Run the commands on the slide (can do both cgat traing

If you ls-l gives you the following:

Gives you serial.error, serial.output

Cat serial.output #gives you which node it used, queue it used and job name, and how many machine it used.

Qsub -l h=cgat016 sge-serial.sh #Will force it to run on that node

Qsub -l mem\_free=2G sge.serial.sh #Says it needs this much memory

Default is that the operating system will choose an output file. But make sure if you specify an output file, you change it each time so you don’t write over the output file from previous job

If you submit a job with more memory than available, it will appear as qw on qstat

Parallel script just runs 4 threads

Qrsh #interactive batch session

Will change which node you’re on (can see with username at start) – each user will be on a diff node

Exit #to logout of qlogin

Can put job requirements on this too

Qrsh -pe dedicated 8 #opens the login session on a node that memory is available (basically reserves the nodes for you and you can just caryr on working normally or submit batch jobs from here)

Can qrsh a particular command ie. Try this and look at user name

Qrsh “echo \$HOSTNAME”

Qdel #delete jobs (you can’t delete other users’s jobs)

Vim #text editor like nano, can quit with :q!

: #command

Q #quit

! # JUST DO IT

**Slide 44-45: DRMA API**

Data structure, that has the info for a qsub command and then make a function for the job submission. Python script available but currently we don’t have the conda environment set up

Portable api across batch systems, but drma gives a site independent way of doing it (I think) DRMA is a good think of communicating between batch systems but can’t gloss over all the site specific batch system things (argh!)

Slides will be available on files tab