

Biology Intel Cluster Information

Overview

The Biology Intel Cluster comprises five Apple dual quad-core 2.8-GHz Harpertown Xserves (all with 16 GB of RAM) and two Apple single quad-core 2.26-GHz Gainestown Xserves (both with 24 GB of RAM) connected to a head node. Job management is handled by the Sun Grid Engine (SGE) package from Sun Microsystems, and the iNquiry Suite from the BioTeam. The Java version is 1.6.

Access and Accounts

Access to the cluster is limited to the head node (sable.biol.mcgill.ca) via SSH (including sFTP) and AFP (Mac file sharing) from the 132.192.0.0/11 subnet only. User accounts are hosted on the head node, which has (for now) sparse disk space, so users are encouraged to limit their accounts to a size of approximately 5 GB.

Submitting Jobs

The SGE, which schedules and distributes jobs to the compute nodes, is a command-line scheduler. Jobs, by default, are scripts or script-wrappers calling other programs, but may also be actual binaries (see 'qsub -b'). Here is an example of a basic script:

```
---
#!/bin/bash
#IMAtest.sh

/common/custom/IMa/ima -i /common/custom/IMa/imtest_unix.u -o /Users/scottyb/IMAtest.txt -
q1 10 -m1 10 -m2 10 -t 10 -b 1000 -L100 -s123 -p45

exit 0
---
```

All scripts must be written with Unix line breaks (!!). Here is how you submit a job to the SGE: 'qsub ./IMAtest.sh'. See the qsub man page for more info. (there are many options). Note that you can also embed the qsub options within the script, as so:

```
# Request shell
#$ -S /bin/bash

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
#$ -a 12241200

# If I run on dec_x put stderr in /tmp/foo, if I
# run on sun_y, put stderr in /usr/me/foo
#$ -e dec_x:/tmp/foo,sun_y:/usr/me/foo
```

```
# Export these environmental variables
#$ -v PVM_ROOT,FOOBAR=BAR
```

```
# The job is located in the current
# working directory.
#$ -cwd
```

etc.

To check that your job has been successfully submitted and is running on a node, use: 'qstat -f -u '*''. You will see under the node identifier your job ID (the first number), your job name, your user name, the job status ("r" for running), and the date and time that it was started. The job ID is important, because if you decide to kill a job you need that number: 'qdel <job ID>' (you may need to add '-f' to force deletion). Completed jobs generate stdout (.o) and stderr (.e) files (by default in your home directory, unless '-cwd' is specified) in addition to any user-defined output files. More complicated script examples can be found at /common/sge/examples/jobs. Always use qsub to submit jobs (i.e., DO NOT EVER run jobs on the head node, as it does not have the resources both to manage the cluster and function as a compute node).

Other Useful SGE Commands

'qhold <job ID>' : prevent a job scheduled for execution in the queue from being considered.

'qrls <job ID>' : release a previously defined job hold.

Parallel Computing and Memory-Intensive Jobs

The cluster is designed as a distributed-computing environment, where individual jobs are accorded a dedicated core for execution (note that since each node has multiple cores, programs written to effect multi-threaded parallelism will take advantage of multiple cores, if they are available). Explicit parallelism (i.e., a single job executing over many nodes via a parallel-programming API) is not possible on this cluster. If you have particularly memory-intensive jobs, send them specifically to the two 24-GB nodes using the '-q' flag with qsub (e.g., 'qsub -q gns.q ./highMemJob.sh').

NFS Mounts

The following head-node directories are shared with the compute nodes:

/common

/Users

/Library/Perl --> Note: This mount is seen by the compute nodes as /RemotePerl

NFS Caveat (IMPORTANT!!!)

The Network File System (NFS) is the weak point of the cluster foundation. It can tolerate large bursts of, but not consistent, activity. Whenever possible, avoid having persistent, open file handles across the NFS. Too much continuous NFS activity between the head node and the compute nodes can bring the NFS down, which necessitates a cluster reboot. This is bad. If your jobs are I/O heavy, the best thing to do is to have your jobs perform local I/O: at the start of your scripts, copy the data you need to /scratch (each compute node has a /scratch for this purpose), write any temp. files to /scratch, and then at the end copy what you need back to your ~ (in /Users). Please do not forget to delete your data or temp. files in /scratch afterward!!!