

ACMMac Cluster Guide

Last updated: January, 2010

About This Guide

This guide is for anyone who is interested in using the ACMMaC High Performance Computing resources. This guide contains information about available resources and about how to connect to and use these resources. If there is additional information which you feel should also be included or if you have questions after reading this guide, please contact Duane Currie (duane.currie@acadiau.ca).

Contents

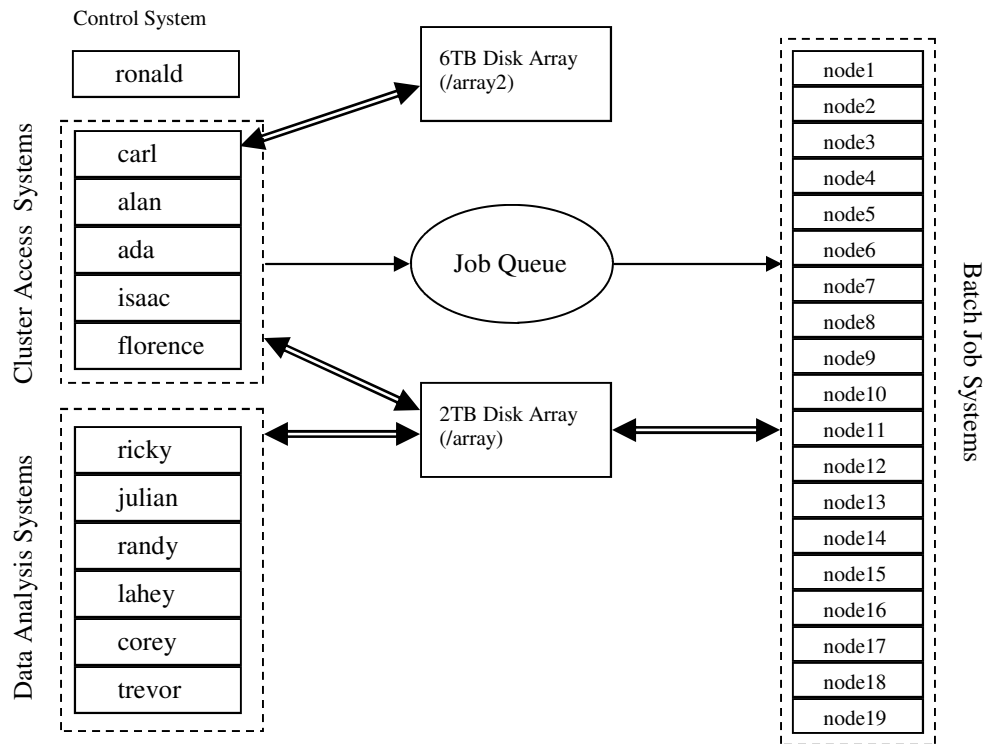
- Available Equipment
- Available Software
- Connecting to the Cluster
- Submitting and Managing Jobs
- Trouble-Shooting

Membership and Access

If you are interested in utilizing our resources, and are not currently a member of ACMMaC, please contact Hugh Chipman (hugh.chipman@acadiau.ca or x1525) or Duane Currie (duane.currie@acadiau.ca or x1773).

Available Equipment

To best describe our available computing equipment, we will begin with a high-level diagram of our computing systems and resources.



Overview

Our computing equipment consists of:

- **Cluster Access Systems**, which users can access and use directly, and from which batch jobs can be submitted to the Job Queue. Each of these systems possesses 2 Opteron cores and 4GB of RAM.
- **Data Analysis Systems**, which users can access and use directly, but these do not provide the ability to submit batch jobs. Each of these systems possesses 8 Opteron cores and 32GB of RAM.
- **Batch Job Systems**, which are not normally used directly. Instead, one submits batch jobs to the Job Queue, from which jobs are selected and scheduled to run on the batch systems. Each of these systems possesses 2 Opteron cores and 4GB of RAM.
- A **2TB disk array** (called /array) which is shared amongst all the machines. It is the sharing of this array which allows your home directory and scratch space to be accessible on every machine.
- A **6TB disk array** (called /array2) which is only accessible from carl. Its purpose is for storing data which does not exist elsewhere, or which cannot be re-generated.
- A **Job Queue**, which is what manages the running of batch jobs on the batch systems. Technically, the job queue is not actually a piece of equipment; rather, it coordinates how a large amount of our equipment is used.
- A **Control System**, ronald, which is generally unavailable for use. It maintains certain operations of the cluster which are not visible to end users.

Available Software

Software Packages		
R 2.9.1	Statistical Programming	Our installation also includes many common libraries, and also parallel programming support
Matlab 2009a	Computational programming	Includes statistics, optimization, differential equations, and parallel computing toolkits.
Mathematica 5.1	Symbolic Math	
Maple 10	Symbolic Math and Arbitrary Precision Computation	
CPLEX 9	Numerical Programming	A CLI-based package for optimization tasks.
OPL Studio	Numerical Programming	A GUI-based package for optimization tasks.
Scilab	Computational programming	Uses a language similar to matlab, and is primarily for numerical computation.
Octave	Computational programming	A very matlab-like package, and will run many matlab codes as is.
OpenBUGS	Statistical Simulations	An implementation of the popular BUGS (Bayesian inference Using Gibbs Sampling) language.
GNUPlot	Simple plotting	A simple, common, 2D plotting utility
Compilers		
GNU Compiler Collection	C, C++, Fortran77, and Fortran 95 Compilers and the related utilities	Includes OpenMP support. MPI is supported with OpenMPI.
FreePascal Compiler	Pascal Compiler	
Sun Studio 12	C, C++, and Fortran Compiler and utilities	Includes OpenMP and MPI support.
Libraries		
Gnu Scientific Library	Scientific Routines for C, C++ and Fortran.	Library for a variety of scientific computing tasks—matrix and vector computation, statistical functions, differentiation, integration, and more
Scalapack	Parallel linear algebra	
OpenMPI	Parallel distributed-memory programming	
Fftw	Fast Fourier Transform routines	
Blas, lapack, atlas	Linear Algebra routines	

Connecting To The Cluster

There are multiple ways to access the cluster, depending on what you wish to accomplish:

Text-Only Login:

Useful when you do not need any graphics displayed to you, and everything can be done with text.

Graphics-Supporting Login:

Similar to the Text-Only login, except that you can also run commands which use graphics, like matlab, plotting in R, and GUI-based programs.

Transferring Files:

For when you need to transfer files between your own machine and the cluster.

Operating System	Login		Transfer Files
	Text-Only	Graphics	
Windows	PuTTY*	XMing* and PuTTY*	FileZilla*
Mac OS/X	ssh	ssh -X	FileZilla* or scp/sftp
Linux/Unix	ssh	ssh -X	FileZilla* or scp/sftp

* software requires installation, but is available for free.
PuTTY: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>
To install, just copy PuTTY.exe to your Desktop
XMing: <http://sourceforge.net/projects/xming/>
FileZilla Client: <http://filezilla-project.org/>

Logging In

Windows:

1. (optional) If you wish to log in and be able to use graphical programs, you'll have to run Xming first: Start->Programs->Xming->Xming
2. Run PuTTY.
3. For hostname, enter the name of your desired Cluster Access System or Data Analysis System
4. (optional) If you wish to log in and be able to use graphical programs, you'll have to select on the left, Connection->SSH->X11, and check the box for "Enable X11 Forwarding"
5. Click Open
6. Enter your username and password

Mac OS/X or Linux/Unix:

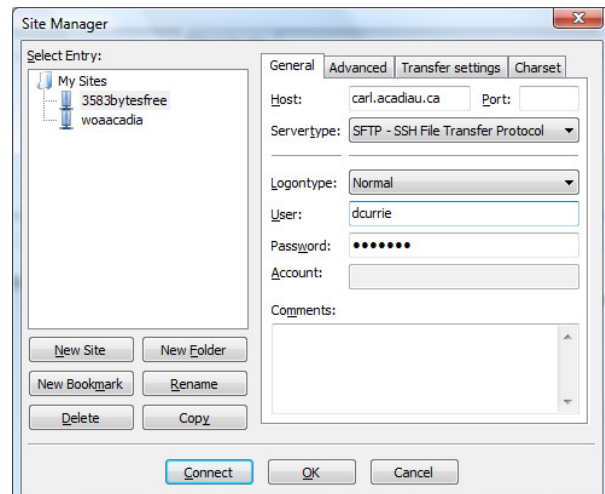
1. Open a terminal (in Mac OS/X, this is under Applications->Utilities)
2. In the terminal, if you wish to be able to log in and run graphics applications, run:
ssh -X <username>@<machinename>.acadiau.ca
or, for text-only mode, run:
ssh <username>@<machinename>.acadiau.ca
Note: replace <username> with your network username, and <machinename> with the name of the Cluster Access System or Data Analysis System you wish to log in to.
3. Enter your password when prompted.

Transferring Files

To transfer files to and from the cluster using FileZilla:

1. Open FileZilla
2. Select File->Site Manager...
3. Set Host to 'carl.acadiau.ca', ServerType to SFTP, Logontype to 'Normal', and provide your username and password in the appropriate boxes.
4. Click Connect

Now, when connected, the left side represents your own machine, and the right side represents the cluster. You can transfer files and folders by dragging and dropping from left to right (upload) or right to left (download).



Preparing Cluster Jobs

In order to submit a cluster job, you must first create a “job script”, which tells the cluster how to run your job. A number of templates are given below for different sorts of jobs. Normally, you should give these script files a ‘.sh’ extension.

Single Job

This is appropriate for general programs that run using only one processor. This is also how you would submit parallel matlab jobs—matlab itself will request the resources your program needs.

```
#!/bin/sh

# Run from current directory
#$ -cwd
# Use this shell
#$ -S /bin/bash
# name that appears in qstat
#$ -N my_job_name
# Email a notice when done (e = ended)
#$ -m e
#$ -M email.address@acadiau.ca

# Load any local settings.
. /etc/profile

# Run my command.
command_name arg1 arg2

# Note: If it's an R program, you might use
# R CMD BATCH program.R
```

Parallel Mathematica Job

This is appropriate for using Mathematica’s Parallel Toolbox. Note: In your program, you only need to use LaunchSlaves[] at the beginning and CloseSlaves[] at the end. The cluster does the rest of the setup for you.

```
#!/bin/sh

# Run in the current directory
#$ -cwd
# Run using this shell
#$ -S /bin/bash
# name that appears in qstat
#$ -N my_math_program
# Mail me after its done
#$ -m e
#$ -M email.address@acadiau.ca
# Use 5 CPUs (1 Master, 4 slave)
#$ -pe mathematica 5

. /etc/profile

# Run the job. $TMPDIR/init.m initializes
information for LaunchSlaves[]

# NOTE: Literally use '$TMPDIR/init.m'.
# NOTE: in the following command, in.m is
# your mathematica program, and the output
# from running it will be in out.txt

math -initfile $TMPDIR/init.m < in.m > out.txt
```

Array of Single Jobs

This is appropriate for when you have a single processor job to run many times on different input files. The example below would run the command ‘model’ 10 times, each on a different input file (in.1, in.2, ...) and save to different output files (out.1, out.2, ...).

```
#!/bin/sh

# Run from current directory
#$ -cwd
# Use this shell
#$ -S /bin/bash
# name that appears in qstat
#$ -N my_job_name
# Email a notice when done (e = ended)
#$ -m e
#$ -M email.address@acadiau.ca
# Run ten times with SGE_TASK_ID going
# from 1 to 10, stepping by 1
#$ -t 1:10:1

. /etc/profile

model < in.${SGE_TASK_ID} > out.${SGE_TASK_ID}
```

Parallel Job using MPI

This is appropriate for when you have a program which runs in parallel using MPI.

```
#!/bin/sh

# Run from current directory
#$ -cwd
# Use this shell
#$ -S /bin/bash
# name that appears in qstat
#$ -N my_job_name
# Email a notice when done (e = ended)
#$ -m e
#$ -M email.address@acadiau.ca
# Run using 8 processors
#$ -pe mpi 8

. /etc/profile

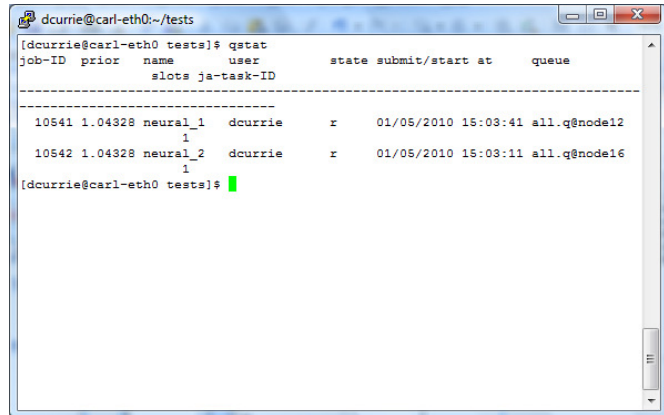
mpirun command
# Note: If you are running a parallel R
# program, use:
# mpirun -np 1 program.R
```

Managing Jobs

Viewing the Job Queue

Jobs in the cluster are managed in a queue. When you submit a cluster job, it is put in the queue. When it is completed, it is removed from the queue. To view what jobs are currently in the queue, use the command 'qstat', as shown on the right.

In the example, we see two jobs in the queue. A status of 'r' means they are running, and a status of 'qw' would mean 'queued and waiting'. A job has a status of 'qw' until there are enough resources for it to run, and then once it is being executed, it gets a status of 'r'.



```
[dcurrie@carl-eth0:~/tests]$ qstat
job-ID prior  name      user      state submit/start at   queue
-----
10541 1.04328 neural_1  dcurrie   r    01/05/2010 15:03:41 all.q@node12
10542 1.04328 neural_2  dcurrie   r    01/05/2010 15:03:11 all.q@node16
[dcurrie@carl-eth0:~/tests]$
```

You can also see from the output of qstat all the jobs in the queue, who their owner is, their name (from the # $\$$ -N line in the job script), the number of cores used (last column), and their Job ID. The Job ID is a value that you may need for cancelling jobs and troubleshooting problems.

Submitting Jobs

To submit a job, you use the command 'qsub', and give it the name of a job script. Example:
qsub neural_run1.sh

Afterwards, the job will be added to the queue, and executed when resources are available.

For more information on options for qsub, run:
man qsub

Cancelling Jobs

You may have times when you want to cancel a job—you might need to change something, you might not need it anymore, or there might be a bug and it's running forever and needs to be stopped.

To cancel a job, run 'qdel' and give it the Job ID of the running job. For example, to cancel the job called 'neural_2' above, you would run:
qdel 10542

TIPS

- Always give your jobs short unique names.**
Later, if you need to cancel or troubleshoot specific jobs, it is much easier if you can identify them individually in the output of qstat. qstat only shows 8 character names, so keep them short.
- Keep the number of CPUS used in a parallel job modest.**
In our case, try to use 16 as a maximum. Most parallel programs perform, per CPU, more efficiently on a smaller number of CPUs. i.e. although a 16-CPU job will end faster than an 8-CPU jobs, 2 8-CPU jobs, using those same 16 CPUs will complete faster than two 16-CPU jobs.
- Tell the cluster when jobs are low priority.**
If a job is not particularly urgent, you can tell the cluster to reduce its priority by adding the following line to the job script:
$\$$ -p -500
- Tell the cluster about you job's memory needs.**
A job may require very little memory, or a lot. To help schedule jobs, include this in your job script:
$\$$ -l h_vmem=200M
The above assumes your job requires at most 200M. To find out what might be an appropriate setting, after running your job once, you can run 'qacct -j JobID | grep h_vmem', and increase the value by 20%-50%.

Troubleshooting / FAQ

My job is taking longer than expected. Can I tell if something's wrong and fix it?

There are three possible situations:

1. The job is running infinitely and incorrectly, but doing nothing.
2. The job is running infinitely and incorrectly, using resources.
3. The job is running fine, but taking longer than you thought.

Situation #1 can usually be diagnosed. If you run 'qstat -F', you will get a listing, machine by machine, of what jobs are running per machine. If you examine the entries for machines for on which your job is running, and the load value per process on the machine is less than (2/3), or especially if near zero, it is likely the case. If it is, you can cancel the job, attempt to determine and fix the cause of the problem, and try again.

Situation #2 is sometimes almost impossible to distinguish from situation #3. The only way to tell is by the other side effects of your job. If your job is supposed to be regularly producing output, you may be able to check the output to identify if the output is valid, and you can use this to determine if the job is running correctly. If there are no such side effects, you might want to build these into your job (such as periodically outputting diagnostic data), and run your job again.

If you can diagnose that situation #2 has occurred, you should cancel the job, and re-submit.

Graphics (or matlab) from the cluster will not display on my computer. What might be wrong?

First, make sure you're on campus. Off-campus connections are too slow to display the graphics screens from the cluster reasonably. Actually, other local universities should be okay, but home network connections will be too slow.

Then, ensure you've connected to the cluster in such a way as to show graphical applications. Instructions are provided earlier in this document.

If neither of the above work, please contact Duane Currie (duane.currie@acadiau.ca), and provide details such as time, location, cluster machine name, and which operating system you are using.

I'm having problems logging in. What's wrong?

Try again, and make sure you're using your current network password. If you've changed your network password, this will affect the cluster as well.

Next, try logging in to carl.acadiau.ca (make sure the acadiau.ca is there).

Otherwise, make sure you can log into other Acadia network programs (e.g. Acorn, email). If those fail, it's likely there is some issue with your network account, and you should contact the Service Desk, in person, or at 585-4357.

If none of these resolve, or help resolve the issue, please contact Duane Currie (duane.currie@acadiau.ca)

My job is queued and waiting, but there should be enough processors able to run it. What can I do?

Currently, we have 36 processors available. First, make sure that the number of processors left is enough to run your job (subtract the number qstat reports as used by running jobs from 36 to determine the number free).

If there should be enough free to run your job, please contact Duane Currie (duane.currie@acadiau.ca), as this can indicate a known minor system problem.

Where should I put the data for my job?

If your job generates much data at all (greater than 10M is a good baseline), you should store the generated data somewhere in your directory either in `/array/data1/` or `/array/data2/`. These are directories for scratch space, i.e. data which is not backed up, but can be re-generated or re-downloaded if necessary.

If the data is a data set which can not easily be re-generated/re-downloaded, you should store the data in a directory on `/array2`. If so, please contact Duane Currie (duane.currie@acadiau.ca) to make sure a suitable location in that directory is created for you. This area is periodically (but not daily) duplicated to a second disk array in another building to avoid potential loss.

Otherwise, for small volumes of data, use your home directory.