**MCRI HPC User Tutorial**

[](http://www.marshfieldresearch.org/)



Table Of Contents

**1 Introduction ..................................................................................................... 3**

**2 Cluster Access ................................................................................................ 3**

**2.1 User Accounts .......................................................................................................... 3**

**2.2 Connecting ................................................................................................................ 4**

**2.3 File Transfers ............................................................................................................ 4**

2.3.1 Linux/OS X ................................................................................................. 4

2.3.2 Windows ..................................................................................................... 5

**3 Cluster Environment ....................................................................................... 5**

**3.1 Reservation System……………………...................................................................... 5**

**3.2 Storage ....................................................................................................................... 5**

**3.3 Job Submission and Scripts .................................................................................... 6**

3.3.1 Submission Using a Script ........................................................................... 6

3.3.2 Submission Using Command Line .............................................................. 7

3.3.3 Job Arrays ................................................................................................... 8

**3.4 Job Management and Scheduler Commands ......................................................... 8**

3.4.1 Queue Status ............................................................................................... 8

3.4.2 Job Management ......................................................................................... 9

3.4.3 Killing Jobs ................................................................................................. 10

# Introduction

The MCRI HPC Cluster consists of the following hardware running CentOS 6.7 OS acquired from Milwaukee Institute in March 2017:

16 – Dell PowerEdge r620 compute nodes comprising of Intel Xeon E5-2660 2.20 Ghz dual-socket 8-cores (Sandybridge, 8.0 GT/s QPI, 20M L2) and 64GB of RAM

1 – Dell PowerEdge r820 Deep computing node nodes comprising of Intel Xeon E5-2660 2.20 Ghz 4-socket 8-cores (Sandybridge, 8.0 GT/s QPI, 20M L2) and 768 GB of RAM

1 - FDR (56gbps) Infiniband switch

1 – Dell PowerConnect 6248 network switch

1 – Dell PowerEdge r620 Login node

1 - Dell PowerEdge r620 Head node running Bright management software

2 – Dell PowerEdge r820 Storage nodes that control the GPFS file system

1 – DDN GPFS storage appliance

This guide is meant to both show how to access the MCRI HPC Cluster, as well as to provide various examples of how to submit jobs to run software using the Torque/Maui scheduler commands. It assumes a basic working knowledge of Linux command line functions. There are numerous tutorials online if you need a primer or a refresher. Here is one I recommend:

<http://linuxcommand.org/lc3_learning_the_shell.php>

The remainder of the guide uses “username” to denote where you would put your own username for purposes of the code examples. When running these commands yourself, substitute your username in for “username” wherever it appears, followed by your password whenever prompted.

Anytime you see a code reference that begins with a “$” this denotes that it is a shell command. The “$” represents the command prompt you would see while logged in, so be sure if copying and pasting not to copy the “$” symbol but everything after it.

Lastly, this is meant to be a reference for you to get started and show you how to access the

cluster but is by no means all encompassing.

This should be sufficient for most users but for those who would like a more comprehensive list of

Torque/Maui functionality, you can reference the user manual from Adaptive Computing here:

Torque

<http://docs.adaptivecomputing.com/torque/4-0-2/help.htm>

Maui

<http://docs.adaptivecomputing.com/maui/index.php>

# Cluster Access

## User Accounts

When you initially receive your account you will receive an e-mail with your username and a temporary password. Once you log in to the system for the first time, you can change your password to whatever you like by running the following command:

$ passwd

Follow the prompts to create and save your new password. If you need it to be reset or if you forget it, email Ryan Frahm at [frahm.ryan@mcrf.mfldclin.edu](mailto:frahm.ryan@mcrf.mfldclin.edu).

## Connecting

All access to MCRI’s HPC cluster is currently done securely via SSH by default. (FTP and telnet are both disabled.) If you have a terminal application, you can use that directly, or download an SSH terminal emulator (like Putty at [\\mcrfnas2\bigdata\Software\Tools\Putty\putty.exe](file:///\\mcrfnas2\bigdata\Software\Tools\Putty\putty.exe)).

The address for the login server is: **hpclogin1-lc**

### Linux/OS X

Pull up a terminal window and type:

$ ssh hpclogin1-lc

You will be prompted for your username and password.

### Windows

You will need to install a terminal emulator of your choice. I recommend PuTTy which can be installed by running [\\mcrfnas2\bigdata\Software\Tools\Putty\putty.exe](file:///\\mcrfnas2\bigdata\Software\Tools\Putty\putty.exe) :

Enter the following credentials;

Host Name: **hpclogin1-lc**

Port: **22**

Connection type: **SSH**

(You can type a name under “Saved Session” and hit save so next time you won’t have to enter it again.) After hitting “Open” you will be prompted for your username and password.

## File Transfers

File transfers are handled using SCP (secure copy protocol).

### Linux/OS X

This example assumes you would like to transfer “myfiles.zip” from your computer to the cluster.

To copy to your base home directory which is in /gpfs/home/username:

$ scp myfiles.zip username@hpclogin1-lc

To copy to the directory myfiles/test/ in your home directory:

$ scp myfiles.zip username@hpclogin1-lc:myfiles/test/

Likewise, you can grab files from the cluster as well by flipping the origin and destination in the command. To grab “myfiles.zip” from your home directory to the cluster:

$ scp username@hpclogin1-lc:myfiles.zip

### Windows

You will need to download an SCP client. I recommend WinSCP, which can be found here:

[\\mcrfnas2\bigdata\Software\Tools\WinSCP\winscp554setup.exe](file:///\\mcrfnas2\bigdata\Software\Tools\WinSCP\winscp554setup.exe)

The login dialog will ask for:

Host name: **hpclogin1-lc**

Port number: **22**

User name: **<yourusername>**

Password: **<yourpassword>**

Some example screenshots:

<http://winscp.net/eng/docs/screenshots>

This will open the Application Interface that will allow you to navigate on both the local server and remote server and drag/drop files between the two.

### From hpclogin1-lc

The MCRI Isilon NAS [\\mcrfnas2\bigdata](file:///\\mcrfnas2\bigdata) is mounted at /mnt/bigdata on every server in the HPC cluster. Files/folders can be copied to/from this location into your home directory at /gpfs/home/username while logged into hpclogin1-lc.

# Cluster Environment

Logging in to hpclogin1-lc will give you access to all the software and compilers we have available. When you login you will be placed initially in your home user directoy.

The cluster utilizes **Torque**, the open source version of PBS (portable batch system) for resource

management, and Adaptive Computing’s **Maui** scheduler that uses information from Torque to

figure out where to schedule jobs on the cluster.

## Reservation System

Maui operates on a reservation system that allocates the number or processors and/or nodes

that you request to your job. The scheduling system is only as smart as you the information you

provide it; it has been set up to schedule based on requested resources, and not utilized resources.

What this means for you: only request the number of processors that you need. If you request 4

processors, but your job only ends up using one, the scheduler will still carve out a 4 processor

block for your job and reserve it completely until the job is complete. Nobody else can use the

other three processors while your job has them reserved, so reserve wisely!

## Storage

The cluster utilizes a highly parallel low latency storage appliance from DDN that stripes data across numerous drives to achieve maximum bandwidth and redundancy. GPFS (general parallel file system) implements the file system on the storage.

Below is a list of common storage locations within the cluster:

*/gpfs/home/* – Location of all home directories

*/mnt/bigdata* – NFS mount for [\\mcrfnas2\bigdata](file:///\\mcrfnas2\bigdata) share

*/gpfs/apps* – Location for shared large software installs

*/cm/shared/apps* – Location for shared software packages and utilities

*/cm/shared/modulefiles* – Location where modules files are installed to define which software to load.

*cm/local* – Location where local torque job info is stored for each node

The /gpfs, /cm/shared, and /mnt/bigdata mounts are available on all nodes in the cluster, and in this way the scheduler can run jobs based on the same absolute directory path for data from anywhere on the system. The safest way to reference your own data is by using the absolute path, such as

/gpfs/home/username/directory/to/my/data

The /gpfs mount has approximately 130 TB of storage.

## Submitting and Running Jobs

The cluster utilizes Torque, the open source version of PBS (portable batch system) for resource management. Maui runs the scheduling system and uses information from Torque to figure out where to schedule jobs on the cluster.

What this means is that in order for you to run jobs on the cluster, you must create a Torque script which you then submit to the scheduler. The scheduler takes your script, reads the info, and runs the commands within then writes a log file and completes.

Torque job scripts are essentially bash scripts with extra pragmas at the top beginning with “#PBS”. (PBS is the original branch of Torque, standing for “public batch system”.

The script can contain any command you would normally run on the command line or in a script—Torque will then run the script when you submit the job. This makes it useful for setting up environment variables, renaming files before and after your executable finishes, or changing directories, etc.

$TORQUE\_HOME = /cm/shared/apps/torque/var/spool

**3.3.1 Submission Using a Script**

Job submission on the cluster is done using the qsub command.  Basic syntax for qsub using a

submission script is as follows:

$ qsub my\_script.sh

The script "my\_script.sh" should contain a number of imperatives that start with #PBS to tell torque how to handle the job.  Take the following example of a simple R batch job:

#PBS -N myjobname  
#PBS -l nodes=4:ppn=8  
#PBS -M [your.email@address.com](mailto:your.email@address.com)  
#PBS -m abe  
#PBS -e ~/pbs\_logs/stderr.txt  
#PBS -o ~/pbs\_logs/stdout.txt

cd $PBS\_O\_WORKDIR

# $PBS\_O\_WORKDIR is set to the directory you ran qsub from  
# In this case, runme.R is in the same directory as “my\_script.sh” and  
# we run ‘qsub’ from this directory  
#  
# This could also be “cd ~” for your homedirectory, or any other directory

# execute program  
R CMD BATCH runme.R

Here's a break down of what the lines in this batch file mean:

* **#PBS -N**  
  Tells Torque what to name your job (otherwise it gets a generic name)
* **#PBS -l procs=1**  
  Asks Torque for one CPU. This means that when your job starts you will have exclusive access to one CPU. For multithreaded jobs, you would switch that number to 4, 8, 16, etc, depending on the number of threads you are working with. Alternate method:  
   **#PBS -l nodes=4:ppn=8.**   
  The other way to request it is by specifying nodes and ppn (procs per node) as above. This will give you 8 procs, and will always try to put them on the same machine. This is the way you must specify format for MPI jobs. For example, if you know your job wants at least 32 cores, but running on ideally 4 nodes, you would specify nodes=4:ppn=8. This will at the minimum give you 8 processors on each of 4 nodes, but will try to give you 16 processors on each of 2 nodes. (It will always preserve the number or processors requested but attempt to pack the job on to as little nodes as possible.)
* **#PBS -M your.email@address.com**Tells Torque what e-mail to use for job notifications *(If you don't use this it might try to append the wrong domain on to your username, so please make sure to use this when submitting jobs)*
* **#PBS -m abe**Tells Torque that you would like job notifications if your job aborts (a), when it starts/begins (b), and when it ends (e).  If you would just like notifications for when your job begins, that would read #PBS -m b, or just notifications for when your job ends, that would read #PBS -m e.  If you want notifications for multiple events, you must group them on one line behind a#PBS -m directive or Torque will only obey the last flag you have in your script.
* **#PBS -e ~/pbs\_logs/**   
  Tells PBS to store all output that would normally be put in stderr into a file in your pbs\_logs directory. This file's name will contain the PBS job number and will have suffix .ER. This enables you to check whether there were any errors running your R program.
* **#PBS -o ~/pbs\_logs/**Tells PBS to redirect all output to a .OU file in your pbs\_logs directory, similarly to the location of the error file in the previous line.
* **#comment**   
  The other lines in the sample script that begin with '#' are comments. The '#' for comments and PBS directives must be in column one of your script file. The remaining lines in the sample script are executable commands.

**3.3.2 Submission Using Command Line (Advanced)**

Any of the above pragmas can be included on the command line for qsub as well. If the same

pragma/option appears as a qsub flag on the command line as well as in the script, the

command line will take precedent. This makes it easy to customize running scripts on the fly by

using a base ‘master’ script and superseding different options when you use qsub.

Running completely from the command line without a script is also possible—but this option is for

more seasoned users. All of the above pragmas may be used on the command line, either in

conjunction with a script or by themselves. To use qsub without a script, you can pipe another

command to the qsub command to have it run via the scheduler. To run “sleep 30” for example,

which will simply spawn a job that sleeps for 30 seconds, you could do

$ echo “sleep 30” | qsub -l nodes=1:ppn=1

**3.3.3 Job Arrays (Advanced)**

If you have a bunch of jobs that you would like to submit based on a number of similarly named

input files or command line parameters, you can use the Torque’s job array function to automate

submitting a number of jobs at once from a single script.

The option you need to use is the “#PBS -J” which will set an iterator on the variable

$PBS\_ARRAY\_INDEX.

#!/bin/bash  
#PBS -l nodes=1:ppn=1  
#PBS -l walltime=00:30:00  
#PBS -J 1-10  
echo "This is job # $PBS\_ARRAY\_INDEX in my job array."

When you do a qsub on this script, it will submit 10 jobs, and in each iterative submission, the

bash variable $PBS\_ARRAY\_INDEX will increment from 1 to 10, specified by the “-J 1-10” in

the script.

This will result in the above example 10 jobs each printing out their PBS\_ARRAY\_INDEX. While

the above example is fairly rudimentary, this is a powerful tool that can be used to implement

filename modifications as inputs to the same program.

Consider the following example. You have 15 input files, all named my\_input\_1.txt through

my\_input\_15.txt. You want to run myProgram with all 15 inputs, so you can use job arrays

to create a single script that will do that for you:

#!/bin/bash  
#PBS -l nodes=1:ppn=1  
#PBS -l walltime=00:30:00  
#PBS -J 1-15  
myProgram –input my\_input\_${PBS\_ARRAY\_INDEX}.txt

This script will submit 15 jobs, each one referencing the file numbers based on the current value

of $PBS\_ARRAY\_INDEX.

When combined with some more advanced bash scripting, this can really help automate job

submission in certain situations. For more advanced examples, please contact me directly.

**3.4 Job Management and Scheduler Commands**

The scheduler has a number of commands that will allow you to check on the status of your jobs,

the position in the queue, and also kill or modify jobs.

**3.4.1 Queue Status**

To see the list of jobs currently in the queue by all users, use the “showq” command:

$ showq

active jobs------------------------

JOBID USERNAME STATE PROCS REMAINING STARTTIME

17005 userA Running 4 3:22:28:13 Mon Jan 24 15:05:05  
17007 userB Running 4 3:22:28:14 Mon Jan 24 15:05:06

This output shows the JOBID of each job, which can be referenced in the next section to get more

details on each job, and that userA, userB, and userC all have running jobs with 4 requested

processors that were submitted on January 14 at 15:05. The “REMAINING” shows how much

walltime is left (default is 10 days, or 10:00:00:00) unless otherwise specified.

There are a number of flags that can change the behavior of this command, all of which can be

combined.

To see a list of all running jobs, add “-r” to the command:

$ showq -r

Or conversely to only see idle jobs waiting in the queue add the “-i” flag:

$ showq -i

To see only the jobs by a certain user:

$ showq -u USERNAME

**3.4.2 Job Management**

The scheduler allows you to check the status of your job using the ‘checkjob’ command.

Adding “-v” to the command gives you detailed information about the job. This command will

only work for your own jobs.

Let’s say you check the queue and want to know more about job 17005.

$ checkjob 17005 –v

job 17005 (RM job '17005.bright.cm.cluster')

AName: my-test-job

State: Running

Creds: user:userA group:userA clas:batch

WallTime: 6:01:59:24 of 10:00:00:00

SubmitTime: Fri Mar 14 15:57:45

(Time Queued Total: 9:23:07:20 Eligible: 9:23:07:18)

StartTime: Mon Mar 24 15:05:05

TemplateSets: DEFAULT

Total Requested Tasks: 1

Total Requested Nodes: 1

Req[0] TaskCount: 1 Partition: torque

Dedicated Resources Per Task: PROCS: 4

Utilized Resources Per Task: PROCS: 0.99 MEM: 2494M SWAP: 2792M

Avg Util Resources Per Task: PROCS: 0.99

Max Util Resources Per Task: PROCS: 0.99 MEM: 2494M SWAP: 2792M

Average Utilized Memory: 2416.81 MB

Average Utilized Procs: 3.96

NodeCount: 1

Allocated Nodes:

[compute-043:4]

SystemID: Maui

SystemJID: 17005

Notification Events: JobEnd Notification Address: [your.email@address.com](mailto:your.email@address.com)

Task Distribution: compute-043,compute-043,compute-043,compute-043

UMask: 0000

OutputFile: login:/gpfs/home/userA/my-test-job/my-test-job.stdout

ErrorFile: login:/gpfs/home/userA/my-test-job/my-test-job.stderr

StartCount: 1

User Specified Partition List: SHARED,torque

Partition List: torque

SrcRM: torque DstRM: torque DstRMJID: 17005.bright.cm.cluster

Submit Args: my\_test\_job.bash

Flags: RESTARTABLE

Attr: checkpoint

StartPriority: 14347

PE: 4.00

Reservation '17005' (-6:01:59:39 -> 3:22:00:21 Duration: 10:00:00:00)

A few things to note are that this will tell you the location of the outputfile, inputfile, as well as the

“Submit Args:” which shows the input script.

**3.4.3 Killing Jobs**

If you submitted a job or a few jobs and later decide you’d like to kill them, simpy run a ‘showq’

command to find the job numbers and use the ‘canceljob’ command

$ canceljob 17005

To cancel jobs 18034, 32483, and 19384, just specify them all at once.

$ canceljob 18034 32483 19384

To cancel all your current jobs, simply run (use with caution!)

$ canceljob ALL