HPCC Basics

Wharton Research Computing Training

Resources

Documentation: http://research-it.wharton.upenn.edu/documentation/
Status Overview: http://hpcc.wharton.upenn.edu/ (from Upenn campus or VPN network only)
E-mail: research-compting@wharton.upenn.edu
Peers: Each other (look around you, these people can be your best resources)
The web: Google is amazing, particularly for CODE help … not so much for Grid Engine, though!

Hardware:

- 32 servers with a total of 512 cores
- 16 cores per server
- 8 TB of total cluster RAM, from 128GB to 512GB per server
- Many TB of HD space. Ability to grow easily

Lots of pre-installed software: Python, R, Julia, Matlab, Mathematica, SAS, Stata, GNU and Intel C/C++/Fortran
Other software or versions of the above available upon request, you may install personal software in your home directory

UNIX (Red Hat Enterprise Linux)

To use the Wharton HPCC environment, you will need to know at least some basic UNIX commands. There are numerous guides and cheat sheets to using UNIX from the command line. I prefer ones that are in a PDF card format, like:

http://research-it.wharton.upenn.edu/wp-content/uploads/2013/09/UNIX-Reference.pdf
For a deeper dive, this tutorial is great! https://swcarpentry.github.io/shell-novice/

UNIX command 'manual' pages: man *command*
You can also do an internet search for 'man *command*'. It's often easier to read in a browser. *NOTE: be aware of possible version differences if you read a man page from the web.*

What Limits Do Users Have?

CPU/cores: each team** may *in theory* (by default) use up to **256 cores** simultaneously, in **any combination** (100 single jobs, 2 x 50-core jobs, etc), with a maximum of 64 concurrent in the all.q (no time limit), and 4 concurrent in the interactive queue. Each user may also have a large number of jobs "queued up" and waiting to be launched as running jobs complete … up to 516 minus whatever you have running. However, we recommend instead using array jobs (-t option to qsub). More below.
RAM: 1024GB of total RAM (all running jobs combined) per team**
RAM: 250GB of RAM per compute host (for a limited number of hosts, the others have 125GB)
Disk: each users has (by default) a 100GB quota. See *your* team's quota with the 'quota' command
** What is a "team"? Generally it is a faculty member and all of their **RAs** and **Co-Authors**. Doctoral students *also* are their own 'team'. All limits are combined under the PI's limits. So if you are Prof X's RA, you and the Prof *both* share up to 256 cores and 100GB disk.

That said, these limits are a generous *baseline*. If you have specific requirements above and beyond these limits, just get in touch and we will discuss your several options, depending on what you may need.

Accessing Wharton HPC Resources

Fingerprints

When you first logon to Wharton HPC systems (from any different individual computer), you will be notified by your SSH or SFTP client that the

keys being received are 'untrusted'.
Below is the public key fingerprint that you should be offered, and which can be trusted (and saved):

SHA256 Fingerprint	BITS	Type
SHA256:nBq4P88AqBV8CizB2mguzdD09EF5EXOW6dvdcgURdKM	256	ECDSA

Command Line: SSH

NOTE: you *must* use one (and only one) of the Wharton or Penn VPNs for all non-Penn Campus network access!!

- Wharton VPN: https://support.wharton.upenn.edu/help/wharton-vpn
- Penn VPN: https://vpn.upenn.edu/

MacOS: built-in ssh (from Terminal)

- OPEN Terminal: Go > Applications > Terminal (I like to add that to my bar at the bottom)
- CONNECT (log on): in the Terminal window, type: `ssh username@hpcc.wharton.upenn.edu`
  - Don't forget to LOOK at the SHA256 fingerprint and make sure it matches the one from the *Fingerprints* table above
  - `exit` (or `logout`) to exit
  - Optional: Use Keys (instead of passwords):
    - In MobaXterm command line window, type: `ssh-keygen`
      - Defaults are fine
    - Append new public key to your authorized_keys file on the HPCC: `ssh username@hpcc.wharton.upenn.edu echo "$(cat ~/.ssh/id_rsa.pub) >> ~/.ssh/authorized_keys"`
    - You are now all set to ssh using keys (no password needed!)

Windows: MobaXterm

Available from: http://mobaxterm.mobatek.net/download-home-edition.html

- INSTALL & LAUNCH MobaXterm
- CONNECT (log on): in the MobaXterm window, type: `ssh username@hpcc.wharton.upenn.edu`
  - Don't forget to LOOK at the Fingerprint and make sure it matches one from the *Fingerprints* table above
  - `exit` (or `logout`) to exit
- Optional: Use Keys (instead of passwords):
  - In MobaXterm command line window, type: `ssh-keygen`
    - Defaults are fine
  - Append new public key to your authorized_keys file on the HPCC: `ssh username@hpcc.wharton.upenn.edu echo "$(cat ~/.ssh/id_rsa.pub) >> ~/.ssh/authorized_keys"`
  - You are now all set to ssh using keys (no password needed!)
- MULTIPLE WINDOWS: click the '+' to create a new window tab, or choose Split type from the Split icon at the top (I love 4!)

File Transfer (SFTP)

Mac OSX: Fetch

University licensed, available at: http://www.upenn.edu/computing/product/ (FTP)

Windows: WS_FTP Professional

University licensed, available at: http://www.upenn.edu/computing/product/ (FTP)

- Install and launch
- "Open a Remote Connection" link
  - Create Site …
    - Site Name: Wharton HPC (or whatever you like, this is just a 'name')
    - Connection Type: SFTP/SSH
    - Server Address: hpcc.wharton.upenn.edu
    - Username: your Wharton Username
    - Password: your Wharton Password
  - CONNECT

- LOOK at the "Untrusted Public Key" Fingerprint and make sure it matches one from the ***Fingerprints*** section on page one. If so, you can "Trust this key". If NOT please contact hpc-admin@wharton.upenn.edu and let them know that there is an SSH key mismatch, and do not continue to connect
- You can drag-and-drop in and out of the windows, or use the arrows to do transfers
- **ASCII tip:** Linux and Windows use different end-of-line characters, which can cause problems when trying to move scripts between Windows and Linux. You may wish to add .DO, .M, .R, .SAS and any other "text" (script) files that you commonly use to the ASCII Filenames list in Tools menu > Options… > Transfers > ASCII Filenames list. This will 'translate' the files during transfer. .SH, .LOG, and many others are already configured to 'translate' during transfer.

File Transfer (Windows Share /network drive)

FROM ON CAMPUS ONLY or VIA WHARTON VPN CONNECTION (see your departmental rep): \\hpcc.wharton.upenn.edu*username*
You can map this as a network drive.

- Windows: My Computer > Map Network Drive > Folder: \\hpcc.wharton.upenn.edu*username*. If the system is not on the Wharton domain (a Wharton-owned system), you should check the "Connect using different credentials" checkbox, and use User name: "wharton*username*" and your normal Wharton password
- Mac OSX: Go menu > Connect to Server … Server: smb://hpcc.wharton.upenn.edu/*username*

NOTE: if you upload files via Windows Share/Samba they may need conversion before running. If you experience an error when you run a program that you've uploaded this way, try 'dos2unix <filename>' and resubmitting.

File Transfer (Dropbox / Box / AWS S3 / Google Drive / Others) using rclone

`rclone` integration allows you to *manually* sync or copy to and from your Dropbox / Box / etc to and from Wharton's HPC environment.

Prerequisite: Dropbox Account (http://dropbox.com/) … Wharton has *unlimited* Dropbox. If you don't have one, e-mail your departmental IT staff for assistance with getting one set up.

See our rclone page for setup and usage details at https://research-it.wharton.upenn.edu/tools/rclone/, and https://rclone.org/ as well.

Graphics: HPCC Desktop

BEST option is to use our Graphical HPCC environment at https://hpcc-desktop.wharton.upenn.edu/. This precludes all of the setup, below, and is a nice place to work.

Portable Devices, other OSes, etc.

Many portable devices have SSH apps. We like Terminus for the iPad and iPhone. While we will make an effort to support these means of access, we can't promise much support, as the volume of apps in this space is quite large. But often quite useful, if you're on the go!

Sharing Files, Working Collaboratively

You may wish to share files or work with other users. rclone via Dropbox (recommended) is a great solution, and as Ras you will likely be working in shared folders on the system. For details see http://research-it.wharton.upenn.edu/documentation/sharing/. If you're working for someone, DO NOT set up project folders for them in your home directories … set them up in their team projects space. That way when you leave there won't be lost work, or challenges with re-writing code paths, etc.

Working Efficiently

Organization
Use an organized directory structure. The format of this structure will depend on your project(s), but it might look something like:

```
projects/common
projects/common/data
projects/common/data/datafile1
```

```
projects/common/code
projects/common/code/codefile1
projects/project1
projects/project1/data
projects/project1/code
projects/project1/logs
projects/project2
projects/project2/data
projects/project2/code
projects/project2/logs
```

Relative Paths
Use relative paths. In other words, try to *not* have /home/*dept*/*username/projects/project1/* at the beginning of any file paths in your code. ~ is better, and *nothing* is even better. This makes the code much more portable!

Software Tricks
There are some tricks in specific software packages to translate from Windows to Linux/MacOS. For example Matlab has both the fullfile and the filesep functions:

```
f = fullfile('dog','cat','frog')

f = dog/cat/frog

>> f = ['dog' filesep 'cat' filesep 'frog']

f = dog/cat/frog
```

Linux/MacOS vs Windows (DOS) text file types

If you're working on a Mac, the text files that you create (job scripts, data files, code files) will all be 'ready to go' in our Linux environment. If you're working in Windows, you *may* need to translate line endings in the files to Linux style. The easy way to do this is to use the `dos2unix filename` command. If you see 'command not found' errors, this is a likely fix.

If you're looking for ways to make your code more portable, reach out to us! We can likely help.

Setting Up and Running Jobs!

*You may not run **intensive processes*** on the login nodes. This is to prevent users from creating problems for other users. To this end, there is very little usable research software on the login nodes.

Gaming the system: our hardware and software does a pretty good job of keeping you from doing any harm to other users' research activities. If we suspect this kind of activity, you may lose your account. This is particularly relevant to you RAs, who may have access to more than one set of faculty resources.

Consequences: if we notice you doing intensive things on the login nodes, or gaming the system, we may restrict your access, and will certainly kill (cancel) your running processes without warning. **Work with us!** If you have questions about your methods, or need resources, let us know.

Interactively
`qrsh -now no` – run a single command on a compute node
`qlogin -now no` – log onto a compute node (generally used for code development and on-the-fly testing)

Batch
This is what Wharton HPCC is all about! At its simplest, without a job script:

```
qsub -N mytestjob -j y -b y 'hostname; sleep 120'
```

A bit more complex, let's get the example code for an R job and submit a job:

Get the demo code:

```
cd ~  # <- go to your home directory from wherever you currently are
cp -r /usr/local/demo/R .  # <- copy the demo code so that you can edit and submit
cd R  # <- enter the demo directory
cat job-script.sh  # <- look at the file!
```

Method 1:

```
qsub job-script.sh
```

Method 2:

```
qsub -N Rdemo -b y 'R --no-save < demo.R'
```

Modify your command so we can examine running job details:

```
qsub -N Sleeper -b y 'R --no-save < demo.R; sleep 300'
```

```
qstat
qstat -j job-ID
qdel job-ID
qstat -s z
qacct -j job-ID
```

Job Arrays

```
qsub -t 1-4 job_script.sh
qsub -N array1 -t 1-4 'R --no-save < demo.R; sleep 300'
```

You can 'continue' an array job with higher numbers by changing `'-t 1-4'` to `'-t 5-100'`, etc.

WHY would you want to run the same thing over and over? SGE_TASK_ID is the answer to some tricky job setups. Consider these two use case examples:

Example #1: Create 4 script files named mycode-1.R (or mycode-1.m, etc.) through mycode-4.R. Now run:

```
qsub -N MyRArray1 -t 1-4 -b y 'R --no-save < mycode-$SGE_TASK_ID.R'
```

So there you go, we're launching all ten with one command.

Example #2: *more* useful: let's say you have 10 tab-separated text data files to evaluate with the same code ... name them mydata-1.txt through mydata-10.txt, and within a Matlab script file called 'mydataread.m':

```
cd ~
cp -r /usr/local/demo/job_array .
cd job_array
qsub -N array -t 1-10 mydataread.sh
```

Each of the ten jobs will read the data file generated with strcat, getenv, and the SGE_TASK_ID environment variable, which is different in each array sub-job.

Parallel Jobs

Do stuff in parallel. While this *can* be useful, it's often *much* slower than Array Jobs because of time to start (all cores must be available) and inter-process communications (array jobs are independent).

Create a script using cat (or some other method that you prefer):

```
cat > mpitest.sh
#!/bin/bash
hostname
sleep 120
Ctrl-d
```

Make sure it looks good:
`cat mpitest.sh`

Change permissions so it will run:
```
chmod +x mpitest.sh
ls -l mpitest.sh
```

Run the script via mpiexec in the mpi parallel environment ('pe'):
```
qsub -pe openmpi 4 -N mpitest -j y -b y 'mpiexec ./mpitest.sh'
qstat
qstat -f
```

Using More RAM
To use more than the default 5GB of RAM for a job, modify the `m_mem_free` option (default 5) of the job, either in your job script:

```
#$ -l m_mem_free=12G
python myPythonCode.py
```

Or as an option to your qsub command: `qsub -l m_mem_free=12G`

That option will be passed to all slots used for a job, so if you're doing a parallel job, each worker will have the adjusted value.

Keep in mind the RAM limits:

- **RAM:** 1024GB of RAM total (all jobs combined) per team**
- **RAM:** 500GB of RAM per compute host *for a very limited number of hosts*, others have less)

For example: if you request 50GB of RAM (`-l m_mem_free=50G`) 100 task Array Job, you will have at most 20 running jobs (1024/50=20 rounded down), and 80 queued jobs.

Job Status and Monitoring

- What is the cluster doing? `hpccstatus (-v for greater detail, -p for team (project) view)`
- What jobs am I currently running? `qstat`
- How can I CANCEL a running job? `qdel` *`job-ID`*
- What jobs are complete? `qstat -s z`
- How can I get stats for a running job? `qstat -j` *`job-ID`*
- How can I get stats for a completed job? `qacct -j` *`job-ID`*
- How can I look at job output? `cat` *`jobname`*`.o`*`job-ID`*
- How can I look at job error? `cat` *`jobname`*`.e`*`job-ID`* (see ***Logging*** below for how to combine output and error logging)

Logging

- SGE creates an output and an error log. Can I combine them? Yes! Add the following option to qsub: `-j y`
- SGE names the output and error log files based on the script name. Can I change this? Yes! Add this option to qsub: `-o` *`outputfilename`*
- SGE names the job after the script name. Sometimes I use the same script for different jobs, or I use the '`echo "command" | qsub`' submission method, which uses STDIN as the job name. Can I change the job name? Yes! Add this option to qsub: `-N` *`jobname`* (cannot start with a number)

E-Mail Notification

- To send an e-mail at job completion, add these options to qsub: `-m e -M` *`e-mail@email.com`*

Practicing and Examples

There are various demo scripts in the /usr/local/demo directories for your use. These directories generally include a `README.txt` file with

details on running the demo, a demo script written in the language being demoed, and a job launch script to launch the job with qsub. Please feel free to make use of these scripts! This is accessible via the command line, or SFTP.

WRDS Data

WRDS data is available at the WRDS website (along with a lot of great ways to get and query it!): http://wrds.wharton.upenn.edu/, in the /wrds directories, and (best!) available via the WRDS PostgreSQL server. If you're using WRDS and SAS exclusively, and have SAS on your local system, you can use the SAS/CONNECT method to connect and use our SAS/CONNECT server (sastcpd.wharton.upenn.edu). Take a look at the SAS Page in the Research Wiki (https://wiki.wharton.upenn.edu/researchcomputing/SAS).