



# Introduction to HPCC at MSU

**Chun-Min Chang**  
***Research Consultant***  
***Institute for Cyber-Enabled Research***

Download this presentation:

<https://wiki.hpcc.msu.edu/display/TEAC/2016-10-18+Introduction+to+HPCC>



## Part1

Some useful information on ICER and HPCC

# Online Resources



- [icer.msu.edu](http://icer.msu.edu): iCER Home
  - [hpcc.msu.edu](http://hpcc.msu.edu): HPCC Home
- [wiki.hpcc.msu.edu](http://wiki.hpcc.msu.edu): HPCC User Wiki
  - Documentation and user manual
- <http://contact.icer.msu.edu>: Contact HPCC
  - Reporting system problems
  - HPC program writing/debugging consultation
  - Help with HPC grant writhing
  - System requests
  - Other general questions
- For training/workshops and registration details, visit:  
<http://icer.msu.edu/events>



# Getting Help



- HPCC user wiki -- <http://wiki.hpcc.msju.edu>
- Contact iCER — <http://contact.hpcc.msu.edu>
- Training information -- <http://icer.msu.edu/upcoming-workshops>
- Linux Command manual:
- An exhaustive list  
<http://ss64.com/bash/>
- A useful cheatsheet  
<http://fosswire.com/post/2007/08/unixlinux-command-cheat-sheet/>
- Explain a command given to you  
<http://explainshell.com/>




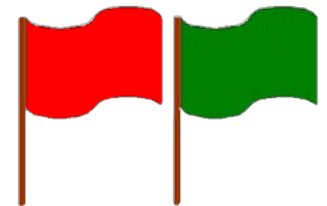
## Part 2 Connecting to HPCC and choosing the suitable nodes

- 1) After you connecting to HPCC, you are now in the gateway. You need to choose the node you want to use (from the development nodes list)
  
- 2) Use 'ssh NODE\_NAME ' to choose the node
  
- 3) Now you are at your home directory. You can also switch to 'Research Space' or 'Scratch Space'.

# Connecting to the HPCC using ssh



- Windows: we recommend “**mobaXterm**” as connection client.
  - Insert the thumb drive
  - Open appropriate folder
  - Looking for MobaXterm
  - Install MobaXterm
  - Server: hpcc.msu.edu or Remote Desktop (rdp.hpcc.msu.edu)
  - Username=your netid; password = your netid password
- OSX:
  - Access spotlight from the menu bar icon (or  **SPACE**)
  - Type ***terminal***
  - In terminal, type: ssh [netid@hpcc.msu.edu](https://hpcc.msu.edu)



# Successful connection HPCC Gateway



```
Session Servers Tools Games Sessions View Split MultiExec Tunneling Settings Help
Quick connect...
/mnt/home/billspat/
Name
.subversion
.TemporaryItems
.thumbnails
.vnc
.xemacs
anaconda3
Ansoft
Ansofttmp
code
Desktop
docs
examples
holekamplab
howto
images
intel
local
MdcsDataLocation
paintings
postgres
python
sasuser.v93
software
Follow terminal folder

5. billspat@hpcc.msu.edu
your own logo, your parameters, your welcome message and generate
either an MSI installation package or a portable executable.
We can also modify MobaXterm or develop the plugins you need.
For more information: http://mobaxterm.mobatek.net/versions.php
[2015-09-21 06:40:08] ~ [myuserid.mylaptop] ~ssh billspat@hpcc.msu.edu
Password:
Last login: Mon Sep 21 06:37:49 2015 from c-68-43-16-37 hsd1.mi.comcast.net

Welcome to Michigan State's High Performance Computing Center
** Unauthorized access is prohibited **

We recommend using dev-intel14 (or nodes with low usage).
For GPU development please use green nodes.
For MIC development please use underlined nodes

Development Nodes (usage)      Filesystem Information
-----
dev-intel07 (low)   dev-intel10 (low)   ${HOME} at 23% usage
dev-gfx10 (low)    dev-intel14 (low)   (used ~226G of 1.0T)
dev-intel14-phi (low)   dev-intel14-k10 (low)

Cluster Load (utilization)
-----
short jobs (< 4 hrs) ( 89%)   general jobs ( 90%)
nodes with > 256 GB ( 88%)   accelerated nodes ( 34%)

[billspat@gateway-00 ~]$
```

HPCC message

Hostname is  
"Gateway"

Command prompt at  
bottom

(Note: file list on left  
side does not  
present in Mac  
terminal)

# Development Nodes



- Shared: accessible by anyone with an MSU-HPCC account
- Meant for testing / short jobs.
- Currently, up to 2 hours of CPU time
- Development nodes are “identical” to compute nodes in the cluster
- Evaluation nodes are “unique” nodes
- Node name descriptive (name= feature, #=year)
- <http://wiki.hpcc.msu.edu/x/pwNe> (HPCC Quick Reference Sheet)





# Development Nodes



name	Processors	Cores	Memory	accelerators
dev-intel14	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	256GB	none
dev-intel14-k20	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	128GB	2x Nvidia Tesla K20 GPUs
dev-intel14-phi	Two 2.5Ghz 10-core Intel Xeon E5-2670v2	20	128GB	2x Xeon Phi 5110P accelerators
dev-intel16	Two 2.4Ghz 14-core Intel Xeon E5-2680v4	28	128GB	none
dev-intel16-k80	Two 2.4Ghz 14-core Intel Xeon E5-2680v4, and four Nvidia Tesla K80 GPUs	28	256GB	4x Nvidia Tesla K80 GPUs

# Compute Nodes



- Use them by job submission (reservation)
- Dedicated: you request # cores, memory, walltime (also for advanced users: accelerators, temporary file space, licenses)
- Queuing system: when those resources are available for your job, those resources are assigned to you.
- Two modes
  - batch mode: generate a script that runs a series of commands (use pbs job script)
  - Interactive mode (qsub -I)



# Summary of MSU HPCC File Systems



GLOBAL	function	path (mount point)	Env
Home directories	your personal	/mnt/home/\$USER	\$HOME
Research Space	Shared with working group	/mnt/research/<group>	
Scratch	temporary computational workspace	/mnt/scratch/\$USER <small>/mnt/ls15/scratch/groups/&lt;group&gt;</small>	\$SCRATCH
Software	software installed	/opt/software	\$PATH
LOCAL			
Operating system	local disk, don't use these	/etc, /usr/lib /usr/bin	
temp or local	fast in each single node only, some programs use /tmp	/tmp or /mnt/local	



# File Systems and static link



Files can have aliases or shortcuts, so one file or folder can be accessed multiple way,  
e.g.

`/mnt/scratch` is actually a static link to

`/mnt/ls15/scratch/users`



try this:

```
ls -l /mnt/scratch
```

```
cd /mnt/scratch/$USER # replace netid with YOUR netid
```

```
pwd
```

What do you see?

Note that "ls15" stands for "Lustre Scratch 2015",  
meaning the scratch lustre file system installed in 2015.



# Private and Shared files



Home directories are private; Research folders are shared by group

```
cd ~
```

```
ls -al
```

```
# -rw----- 1 billspat staff-np 3360642 Apr 9 10:35  
  myfiles.sam
```

Group Example: (you may not have a research group)

```
ls -al /mnt/research/holekamlab #this won't work for you
```

```
-rw-rw-r-- 1 billspat holekamlab 3360642 Apr 9 10:35 eg1.sam
```

*The Difference is read and write (rw) by group*

# /mnt/scratch/netid may be public to system unless you make it private

To share a file using your scratch directory:

```
touch ~/testfile
```

```
cp ~/testfile /mnt/scratch/<netid>/testfile
```

```
chmod g+r /mnt/scratch/<netid>/testfile
```



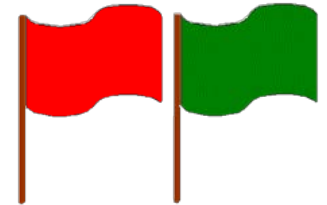
## Part 3 Transferring files and Mapping HPC drives

- 1) This part covers several transferring methods.
- 2) You can even mapping your HPC drives to a campus computer

# Transfer files



- Load to/from your (use hostname: [rsync.hpcc.msu.edu](https://rsync.hpcc.msu.edu))
  - MobaXterm
  - Filezilla or Cyberduck (thumb drive)
  - Mapping to HPCC
  - rsync, sftp commands
- Load to/from HPCC
  - wget
  - curl



<https://wiki.hpcc.msu.edu/display/hpccdocs/Transferring+Files+to+the+HPCC>

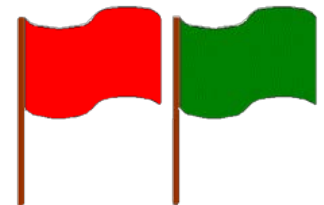
<https://wiki.hpcc.msu.edu/display/Bioinfo/UNIX%2C+SSH+and+SCP>



# Transferring Files



- SFTP program
  - download and install **cyberduck**
  - <https://cyberduck.io/>
- Hostname: **rsync.hpcc.msu.edu**  
**or: rdp.hpcc.msu.edu**
- Username: **your netid**
- Password: **your netid password**
- Port : **22**





# Copying files to/from HPCC



Can copy files from system to laptop with secure ftp = sftp or secure copy = scp

Example

1. create a file to copy, please type

```
cd ~/class
```

```
echo 'Hello from $USER' > greeting.txt
```

```
echo 'Created on `date`' >>greeting.txt
```

```
cat greeting.txt
```

# make sure you are in the directory with `pwd` command, and it's lower case

2. exit from the HPCC

```
exit; exit
```

3. in terminal or Moba issue following command scp => secure copy

```
scp netid@rsync.hpcc.msu.edu:class/testfile.txt
```

```
testfile.txt
```

```
open testfile.txt
```



Windows: MobaXTerm auto connected, files are listed on the side



# Mapping HPC drives to a campus computer



You can connect your laptop to the HPCC home & research directories  
Can be very convenient and work on Windows 10!

For more information see: <https://wiki.hpcc.msu.edu/x/VYCe>

1. determine your file system using the command

```
df -h ~
```

Filesystem	Size	Used	Avail	Use%	Mounted on
------------	------	------	-------	------	------------

ufs-10-b.i:/b10/u/home/billspat					
---------------------------------	--	--	--	--	--

2. server path

Mac: smb: //ufs-10-b.hpcc.msu.edu/billspat

Windows: \\ufs-10-b.hpcc.msu.edu\billspat

3. connect

Mac : Finder | connect to server | put in server path | netid and Password

Windows : My Computer | Map Network Drive | select letter

Windows fix: user name = hpcc\netid (same password)



## Part 4 Modules

- 1) Modules relate to built-in functions in HPCC. You can load/unload your own modules if needed
- 2) 'Powertools' is a very useful module in HPCC. After loading this module, follow the guide in this part, and you can get a package called 'intro2hpcc'. The scripts in 'intro2hpcc' would be helpful in submitting jobs to HPCC.

# Module System



- For maximizing the different types of software and system configurations that are available to the users, HPCC uses a Module system to set Paths and Libraries.
- Key Commands
  - ***module list*** : List currently loaded modules
  - ***module load*** modulename : Load a module
  - ***module unload*** modulename : Unload a module
  - ***module spider*** keyword : Search modules for a keyword
  - ***module show*** modulename : Show what is changed by a module
  - ***module purge*** : Unload all modules



# Using Modules



```
module list      # show modules currently loaded in  
                  your profile
```

```
module purge     # clear all modules
```

```
module list      # none loaded
```

```
# which version of Matlab do you need?
```

```
module spider Python
```

```
module spider Python/2.7.2
```

```
# after purging (no modules), find the path to Python with the Linux "which"  
command
```

```
which python; python --version
```

```
module load python
```

```
which python
```

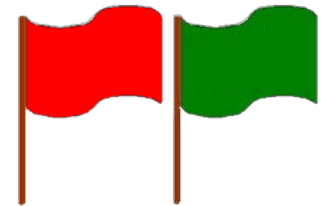


# Powertools



- ***Powertools*** is a collection of software tools and examples that allows researchers to better utilize High Performance Computing (HPC) systems.

- ***module load powertools***



- quota # list your home directory disk usage
- poweruser # Set up your account to load powertools by default
- priority\_status # IF part of a buy-in group; Shows the status of an individuals (buy-in nodes)
- userinfo <\$USER> # get details of a user (from public MSU directory)



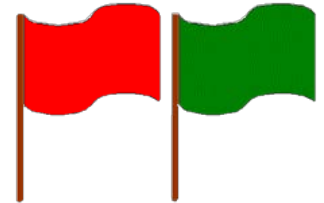
# Load supply from HPCC



- Please Load Software modules type (in your terminal):

***module load powertools***

***getexample intro2hpcc***



See the changes before and after module load

- This will copy some example files (for this workshop) to your directory.
- Exercise: try to find the file '**youfoundit.txt**' in the "**hidden**" directory.



# Useful *Powertools* commands



- ***getexample*** : Download user examples
  - ***powertools*** : Display a list of current powertools or a long description
  - ***licensecheck*** : Check software licenses
  - ***avail*** : Show currently available node resources
- 
- For more information, refer to this webpage (HPCC Powertools)  
<https://wiki.hpcc.msu.edu/x/p4Bn>





## Part 5 Submitting Job to HPCC

- 1) This part includes how to submit a job to HPCC
- 2) The guide to write the submitting file is included. 'hello.qsub' is a simple example.

# Need to run Jobs with more resources?



- We've focused so far on working interactively (command line)
- Need dedicated resources for a computationally intensive task?
  - List resource requests in a job script
  - List commands in the job script
  - Submit job to the scheduler
  - Check results when job is finished



# The Queue



The system is busy and used by many people  
may want to run one program for a long time  
may want to run many programs all at once on multiple nodes or cores

To share we have a 'resource manager' that takes in requests to run programs, and allocates them

you write a special script that tells the resource manager what you need, and how to run your program, and 'submit to the queue' (get in line) with the qsub command

We submitted our job to the queue using the command  
qsub hello.qsub

And the 'qsub file' had the instructions to the scheduler on how to run



# Scheduling Priorities



- NOT First Come First Serve!
- Jobs that use more resources get higher priority (because these are hard to schedule)
- Smaller jobs are backfilled to fit in the holes created by the bigger jobs
- Eligible jobs acquire more priority as they sit in the queue
- Jobs can be in three basic states: (check states by `showq`)
  - Blocked, eligible or running
  - Jobs in BatchHold

# Submission of Job to the Queue



A Submission Script is a mini program for the scheduler that has :

1. List of required resources
2. All command line instructions needed to run the computation including loading all modules
3. Ability for script to identify arguments
4. collect diagnostic output (qstat -f)

Script tells the scheduler how to run your program on the cluster. Unless you specify, your program can run on any one of the 7,500 cores available.



# Submission Script



- List of required resources

<https://wiki.hpcc.msu.edu/display/hpccdocs/Scheduling+Jobs>

- All command line instructions needed to run the computation



# Compile, Test, and Queue



The README file has instructions for compiling and running

```
gcc hello.c -o hello
```

```
./hello
```

```
qsub hello.qsub
```

*# compile and output to new program 'hello'*

*# test run this hello program. Does it work?*

*# submit to queue to run on cluster*

```
2. ssh
[billspat@dev-intel10 helloworld]$ gcc hello.c -o hello
[billspat@dev-intel10 helloworld]$ qsub hello.qsub
29444624.mgr-04.i
[billspat@dev-intel10 helloworld]$
```

This 'job' was assigned the 'job' id 29444624 and put into the 'queue.'

Now, we wait for it to be scheduled, run, and output files created...



# hello.qsub



```
#!/bin/bash -login
#PBS -l walltime=00:05:00
#PBS -l nodes=1:ppn=1

cd ${PBS_O_WORKDIR}
./hello

qstat -f ${PBS_JOBID}
```





# Details about job script



- “#” is normally a comment **except**
  - “#!” special system commands
  - ***#!/bin/bash***
  - ***#PBS*** instructions to the scheduler

***#PBS -l nodes=1:ppn=1***

***#PBS -l walltime=hh:mm:ss***

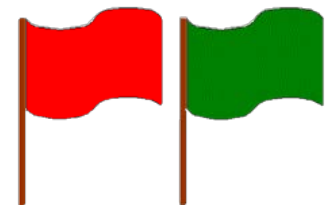
***#PBS -l mem=2gb*** (!!! Not per core but total)
- <http://wiki.hpcc.msu.edu/x/Np-T> (Scheduling Jobs)



# Submitting and monitoring



- Once job script created, submit the file to the queue: ***qsub hello.qsub***
- Record job id number (#####) and wait around 30 seconds
- Check jobs in the queue with: ***qstat -u netid & showq -u netid***
- What about using the Commands ***qs & sq*** ?
- Status of a job: ***qstat -f jobid***
- Delete a job in a queue: ***qdel jobid***



# Typical Submission Script



```
#!/bin/bash -login
```

```
### define resources needed:
```

```
#PBS -l walltime=00:01:00
```

```
#PBS -l nodes=5:ppn=1
```

```
#PBS -l mem=2gb
```

```
### you can give your job a name for easier identification
```

```
#PBS -N name_of_job
```

```
### reserve license for commercial software
```

```
#PBS -W x=gres:MATLAB
```

```
### load necessary modules, e.g.
```

```
module load MATLAB
```

```
### change to the working directory where your code is located
```

```
cd ${PBS_O_WORKDIR}
```

```
### call your executable
```

```
matlab -nodisplay -r "test(10)"
```

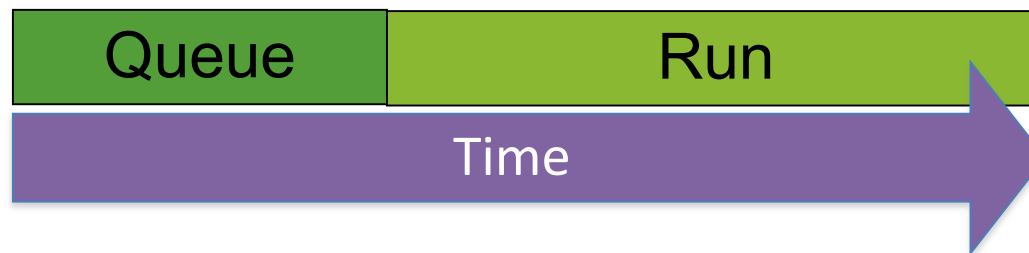
```
# Record Job execution information
```

```
qstat -f ${PBS_JOBID}
```

# Scheduling Tips



- Requesting more resources does not make a job run faster unless you are running a parallel program.
- The more resources you request, the “harder” it is for the job manger to schedule the resources.
- First time: over-estimate how many resources you need, and then modify appropriately.
- ***qstat -f \${PBS\_JOBID}*** at the bottom of your scripts will give you resources information when the job is done)
- Job finished time = Job waiting time + Job running time



# Advanced Scheduling Tips



- Resources
  - A large proportion of the cluster can only run jobs that are four hours or less
  - Most nodes have at least 24 GB of memory
  - Half have at least 64 GB of memory
  - Few have more than 64 GB of memory
  - Maximum running time of jobs: 7 days (168 hours)
  - Maximum memory that can be requested: 6 TB
- Scheduling Limit
  - total 520 cores using on HPCC at a time
  - 15 eligible jobs at a time
  - 1000 submitted jobs

# Job Completion



- By default the job will automatically generate two files when it completes:
  - Standard Output:
    - Ex: jobname.o5945571
  - Standard Error:
    - Ex: jobname.e5945571
- You can combine these files if you add the join option in your submission script:  
***#PBS -j oe***
- You can change the output file name  
***#PBS -o /mnt/home/netid/myoutputfile.txt***

# Other Job Properties



- resources (-l)
  - Walltime, memory, nodes, processor, network, etc.

<https://wiki.hpcc.msu.edu/display/hpccdocs/Advanced+Specification+of++Resources>

***#PBS -l feature=gpgpu,gbe***

***#PBS -l nodes=2:ppn=8:gpu=2***

***#PBS -l mem=16gb***
- Email address (-M)

***#PBS -M yourNetID@msu.edu***
- Email Options (-m)

***#PBS -m abe***

Many others, see the wiki:

<http://wiki.hpcc.msu.edu/>

# Advanced Environment Variables



- The scheduler adds a number of environment variables that you can use in your script:

<https://wiki.hpcc.msu.edu/display/hpccdocs/Advanced+Scripting+Using+PBS+Environment+Variables>

## ***PBS\_JOBID***

- The job number for the current job.

## ***PBS\_O\_WORKDIR***

- The original working directory which the job was submitted

- Example

```
mkdir ${PBS_O_WORKDIR}/${PBS_JOBID}
```



# Common Commands



- ***qsub*** “submission script”
  - Submit a job to the queue
- ***qdel*** “job id”
  - Delete a job from the queue
- ***showq -u*** “user id” , ***qstat -u*** “user id”
  - Show the current job queue of the user
- ***checkjob -v*** “job id” , ***qstat -f*** “job id”
  - Check the status of the current job
- ***showstart -e all*** “job id”
  - Show the estimated start time of the job



## part 6      Using GUI

1) In order to run a GUI software (e.x. Matlab), you should have X11 running on your computer

2) To enable X11, using the command line 'ssh -XY netid@hpcc.msu.edu' when log in.

# What is needed for X11?



- X11 server running on your personal computer
- SSH connection with X11 enabled
- Fast network connection
  - Preferably on campus



# Run X11 on Windows

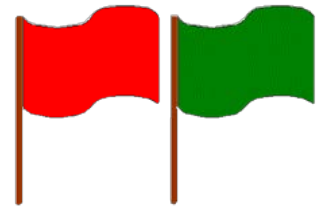


- Install MobaXterm

<http://mobaxterm.mobatek.net/download.html>

- at MobaXterm,

– `ssh -X username@gateway.hpcc.msu.edu`

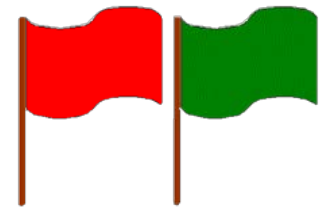


Run remote desktop on

[rdp.hpcc.msu.edu](http://rdp.hpcc.msu.edu)



# Run X11 on Mac (OSX)



- Mac users:
- Install XQuartz from iCER thumb drive
- Run XQuartz
- Quit terminal
- Run terminal again
- `ssh -X username@gateway.hpcc.msu.edu`

Download remote desktop from Apple store (free)  
and run it on

[rdp.hpcc.msu.edu](http://rdp.hpcc.msu.edu)



# Test GUI using X11



- run X11
- Try one of the following commands

***xeyes***

or/and

***firefox***

