# Introduction to HPC at MSU

CYBERINFRASTRUCTURE DAYS 2014

Oct/23/2014

Yongjun Choi

choiyj@msu.edu

Research Specialist,

Institute for Cyber-Enabled Research

MICHIGAN STATE UNIVERSITY

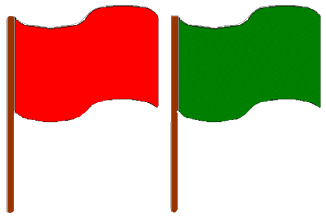
ICER

# Agenda

- Introduction to HPCC
  - Introduction to iCER
- How to Use the HPC
  - Get an account
  - Install needed Software (SSH, X11, Xming/XQuartz)
  - Basic navigation commands
  - Transfer files
  - Compile/Test programs on a developer node
  - Write a submission script
  - Submitting a job

# How this workshop works

- We are going to cover some basics.
  - Lots of hands on examples
- Exercises are denoted by the following icon in this presents:



# Green and Red Sticky

- Use the provides sticky notes to help me help you.
  - **No Sticky** = I am working
  - **Green** = I am done and ready to move on (yea!)
  - **Red** = I am stuck and need more time and/or some help

# Agenda

- Introduction to HPC
  - **Introduction to iCER**
- How to Use the HPC
  - Get an account
  - Install needed Software (SSH, X11, Xming/XQuartz)
  - Basic navigation commands
  - Transfer files
  - Compile/Test programs on a developer node
  - Write a submission script
  - Submitting a job

# iCER

- What does iCER stand for?
  - Institute for Cyber-Enabled Research
- Mission
  - Reducing the "Mean time to Science"
  - iCER's mission to help researchers with computational components of their research

# iCER Overview

- iCER is a research unit at MSU. We:
  - Maintain the university supercomputer
  - provide "software-as-a-service"
  - Organize Training
  - Provide 1-on-1 consulting
  - Help with grant proposal

# Funding From…

- The Vice President office for Research and Graduate Studies (VPRGS)
- Engineering College, College of Natural Science and College of Social Science
- This allows us to provides services and resources for **FREE!!!**

# Online Resources

- **icer.msu.edu**: iCER Home
  - **hpcc.msu.edu**: HPCC Home
- **wiki.hpcc.msu.edu**: HPCC User Wiki

# When would I use the HPC?

- Takes too long for computation
- Runs out of memory
- Needs licensed software
- Needs advanced interface (visualization/database)
- Read/write Lots of data

# What is a supercomputer?

MICHIGAN STATE UNIVERSITY

ICER

# What is a supercomputer?

- A computer at the frontline of contemporary processing capacity
- Introduced in 1960s, Seymour Cray

- MSU's supercomputer is a "collection" of computers that feature:
  - High FLOPS (Floating Point Operations/Second)
  - Fast Network
  - Fast/Reliable File Services

# Agenda

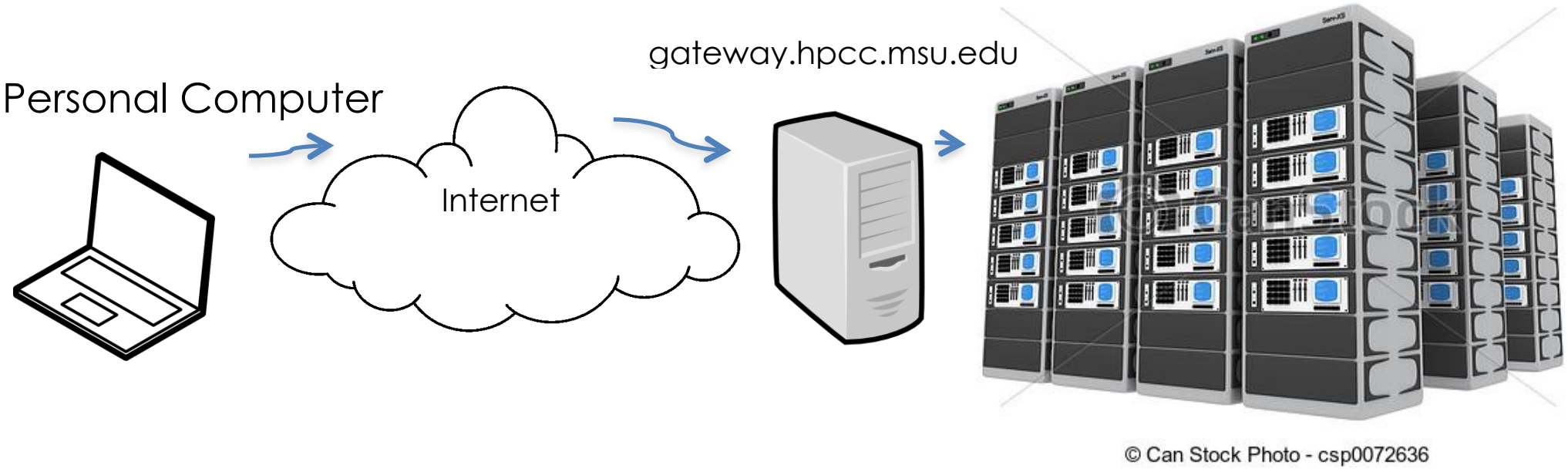
- Introduction to HPC
  - Introduction to iCER
- How to Use the HPC
  - **Get an account**
  - Install needed Software (SSH, X11, Xming/XQuartz)
  - Basic navigation commands
  - Transfer files
  - Compile/Test programs on a developer node
  - Write a submission script
  - Submitting a job

# Accounts

- PIs can request accounts (for each group member) at http://contact.icer.msu.edu/account
- Each account has access to:
  - 50 GB of replicated file space (/mnt/home/userid)
  - 520 processing cores
  - 360 TB of high-speed scratch space (/mnt/scratch/userid)
- Also available: shared group folder upon request

# Nuts & Bolts

- Our supercomputer is a "remote service"

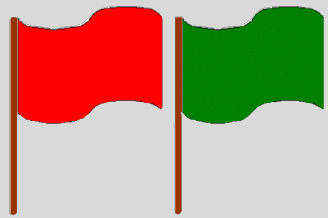



http://wiki.hpcc.msu.edu/x/DYAf

# Agenda

- Introduction to HPC
  - Introduction to iCER
- How to Use the HPC
  - Get an account
  - **Install needed Software (SSH, X11, Xming/XQuartz)**
  - Basic navigation commands
  - Transfer files
  - Compile/Test programs on a developer node
  - Write a submission script
  - Submitting a job

# Connecting to the HPCC using ssh



- Windows: we have "putty" pre-installed on iCER thumb drive.
  - Insert the thumb drive
  - Open appropriate folder
  - StartPortableApps.exe
  - Locate putty in the menu
  - double click on spcc default connection
  - Username=your netid; password = your netid password
- OSX:
  - Access spotlight from its menu bar icon (or ⌘ **SPACE**)
  - Type **terminal**
  - In terminal, type: ssh netid@gateway.hpcc.msu.edu

# Connecting to the HPCC using ssh


MICHIGAN STATE UNIVERSITY

ICER

# Types of Nodes

gateway.hpcc.msu.edu

eval.hpcc.msu.edu

GATEWAY NODES

dev-intel07

dev-intel10

dev-gfx10

dev-intel14

dev-intel14-k20

dev-intel14-phi

DEVELOPER NODES

© Can Stock Photo - csp0072636

COMPUTE NODES

MICHIGAN STATE UNIVERSITY

ICER

# Gateway Nodes

- Shared: accessible by anyone with an MSU-HPCC account.
- Shared resource -- hundreds of users on the gateway nodes
- Only means of accessing HPCC compute resources.
- Useful information (status, file space, messages)
- ** DO NOT RUN ANYTHING ON THESE NODES! **

# Developer/eval 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

# Compute Nodes

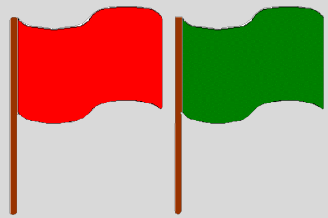
- Dedicated: you request # cores, memory, walltime. (also for advanced users: accelerators, temporary file space, licenses)
- Queuing system: when those resources are available for your job, those resources are assigned to you.
- Two modes
  - batch mode: generate a script that runs a series of commands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

# What is a Shell?

- A command interpreter that turns text into instructions
  - Text can be entered interactively in the command line
  - Text can be contained within a file
- On the HPCC, default shell is called BASH

  (aka Borne-Again shell)
- Note: use "bash" as search parameter in google!

# Secure shell



- If you're not connected to the HPCC, please do so now.
- From gateway, please connect to a development node

  ***ssh dev-intel10***
- ssh == secure shell (allows you to securely connect to other computers)

# Agenda

- Introduction to HPC
  - Introduction to iCER
- How to Use the HPC
  - Get an account
  - Install needed Software (SSH, X11, Xming/XQuartz)
  - **Basic navigation commands**
  - Transfer files
  - Compile/Test programs on a developer node
  - Write a submission script
  - Submitting a job

# Basic navigation commands: **ls**

| Command | Meaning |
|---|---|
| **`ls`** | list files and directories |
| `Some options for` **`ls`** `command` | |
| **`-a`** | list all files and directories |
| **`-F`** | append indicator (one of */=@\|) to entries |
| **`-h`** | print sizes in human readable format (e.g., 1k, 2.5M, 3.1G) |
| **`-l`** | list with a long listing format |
| **`-t`** | sort by modification time |

# Basic navigation commands: **cd**

| Command | Meaning |
|---|---|
| **`cd directory_name`** | change to named directory |
| **`cd`** | change to home-directory |
| **`cd ~`** | change to home-directory |
| **`cd ..`** | change to parent directory |
| **`cd -`** | change to the previous directory |
| **`pwd`** | display the path of the current directory |

# Basic Commands

- To find a command, use a search engine
- Complete options/instructions for that command

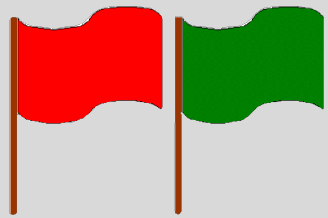
  ***man*** command_name
- An exhaustive list

  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/

# Exercise!



- Please type (in your terminal):

  ***module load powertools***

  ***getexample intro_workshop***

•This will copy some example files (for this workshop) to your directory.

•Exercise: try to find the file '**youfoundit.txt**' in the "**hidden**" directory.

# (Advanced Tip)

- Tab Completion short cut
  - Typing out directory/file/program names is time consuming.
  - When you start typing out the name, hit tab key. The shell will try to fill in the rest of the directory name
  - E.g., return to home directory

    **cd**

    type ***cd l***, then press tab

# (Advanced tip 2)

- Access previous command by pressing up arrows
- See full history by typing

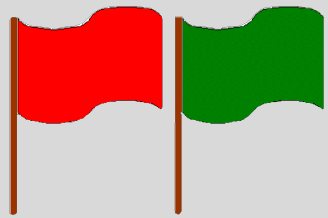
  ***history***
- Repeat previous command in history by using the exclamation sign, e.g.

  *[choiyj@dev-intel10] %* ***history***

  …

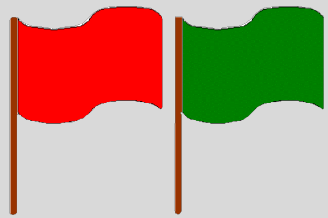
  ***8 ls -lat***
- ***!8*** will call the command "ls -lat"

# Practicing navigation again…



- Navigate to the data/ directory
- Use the "tab completion" feature
- Try to find the name and size of the file in that directory.

# Examining Files



- Print all contents of a file using cat

  ***cat*** *filename*

- When the file is really big, use "less"

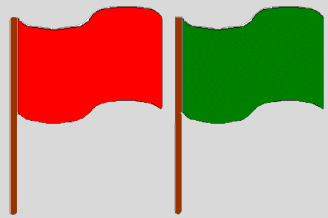
  ***less*** *filename*

  - Use arrow keys to scroll by line
  - Space to go to the next page
  - "b" to go backwards
  - "g" to go to the beginning
  - "G" to go to the end
  - "q" to quit
  - "/" to seach

# Basic manipulation commands

| Command | Meaning |
|---|---|
| **cp <from> <to>** | Copy files |
| **cp –r <from> <to>** | Copy recursively: files and directories |
| **mkdir *directory*** | Make named directory |
| **mv <from> <to>** | Move a file (can be used as a rename command!) |
| **rm filename** | Remove the file |
| **rmdir directory** | Remove an empty directory |
| **rm –r directory** | Remove a directory recursively (i.e. include all subdirs and files) |

# Short Exercise

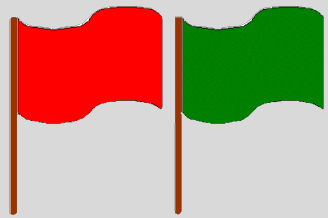


1. Rename the ***.hidden*** directory to ***not_hidden***
2. Create a new directory called new
3. Copy the file ***youfoundit.txt*** into the new directory

# Creating / Editing text files

- You "could" create/edit files on your local system and transfer them using filezilla
  - Downside: slow
  - Need to run command "dos2unix" if you are running a windows system
- Much "faster" in the long term to learn how to edit files in the command line

# Editors



- Choice of editor is really a religion
  - emacs (popular, powerful)
  - **vi / vim/ gvim** (popular, powerful)
  - Nano (simple, easy to learn)
- Today, we will learn nano. To start, type

  *nano newfile.txt*
- Commands are in the bottom. (**^** means control key)
- Create a file that says: this is a text file!

# Nano

- Nano does not leave any output on the screen after it exists.
- But ***ls*** now shows that we have a new file called ***newfile***
- Lets tidy up by deleting this new file:

  ***rm newfile***

** *NOTE: no undelete in Linux (unlike windows)*

# Other useful commands

- ***wget*** (get something from the www)

  wget http://…
- ***ps*** (get processes, i.e. running programs)
- ***top*** (get cpu/memory utilization)
- ***find*** (a way to locate files)
- ***grep*** (search for patterns)
- ***wc*** (get number of words/lines)
- ***bc*** (bash calculator)

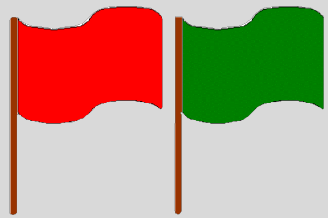
# Available Software

- Compilers, debugger,s and profilers
  - Intel compilers, openmp, openmpi, mvapich, totalview, GNU……
- Libraries
  - ACML, BLAS, FFTW, LaPACK, MKL……
- Commercial Softwares
  - MATLAB, Mathematica, FLUENT, Abaqus……
- For a more up to date list, see the documentation wiki:
  - http://wiki.hpcc.msu.edu/

# Module System

- To maximize the different types of software and system configurations that are available to the users, HPCC uses a Module system
- 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

# Short Exercise



- Unload all modules and load these modules
  - GNU, powertools
- Check which modules are loaded
- Several versions of MATLAB are installed on HPC. Find what versions are available on HPC. Load the latest version.

# Powertools

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

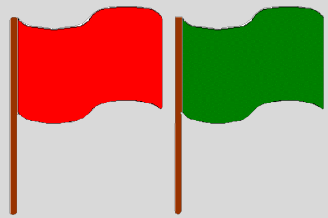
- ***module load powertools***

# 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
- https://wiki.hpcc.msu.edu/x/p4Bn

# *getexample*



- If you do not load ***powertools***, please do it now:
- Download the ***helloworld*** example using ***getexample***
- Check what you downloaded. What is the biggest file?

# *getexample*

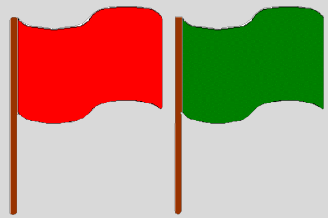
- You can obtain a lot of examples through ***getexample***. Take advantage of it!

```
Possible example names:
abaqus_example          cloudy_example          fortran_openmp_blas  makeflow            Molpro_example      qsub_arraydepend
ADMB_example            condor_advanced         FreeSurfer           MATLAB_basic        mothur              R_example
ADMB_example2           condor_basic            GAMESS_example       MATLAB_blcr         mothur2             R_plot
affinity                condor_Python           gmp_mpfr             MATLAB_compiler     mothur_example      SAS_example
allinea_map             condor_R                gromacs              MATLAB_compiler2    MPI_pi              simpleMatlab
Amber_CUDA_example      condor_simple           Hadoop_wordcount     MATLAB_many_jobs    multi_variable      SNPPipeline
Amber_example           cuda                    HEEDS_test           MATLAB_movie        myhadoop            splitBam
avida_blcr              cuda_clock              helloMPI             MATLAB_parameter_sweep NAMD_CUDA_example  STATA_example
basic_array_job         cuda_hybrid             helloOpenMP          MATLAB_parfor       NAMD_example        stata_parallel
blast                   CUDA_WORKSHOP_UIUC1208  helloworld           MATLAB_parfor2      Octave_basic        tbb_example
blender_farm            DDT_examples            HFSS_example         MATLAB_patternsearch OpenACC_example    TotalView_MPI_example
BOOST_example           econ_examples           ImageJ               MIC_examples        OpenCL_hello_world  Valgrind_example
bowtie                  espresso_benchmark      intro_workshop       MKL_benchmark       OpenMP_profiling    VASP_example
brother_test            fftw                    lammps_test          MKL_c_eigenvalues   paraview_basic      velvet_blcr
burn_heat               fluent3D                LAPACK_example       MKL_Example         pbdR_examples       XSEDE_MPI_WORKSHOP
Circuitscape_examples   fluentMPI               magma_example        MKL_mic             pcap_example
Clang_example           fortran_openmp          makefile_example     MKL_parallel        Python_MPI
```

# Standard in/out/err and piping

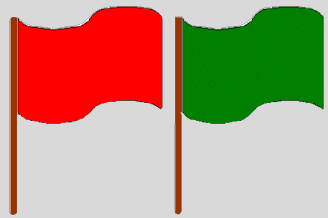
- You can redirect the output of a program to a file using “>” character:
- myprogram **>** output.txt
- You can also cause the output of the program to be the input of another program using the “|” pipe character:
- myprogram |myotherprogram

# Redirection and Piping



- Change to the ***helloworld*** directory
- Redirect the manual of the ***ls*** command to **ls.txt**:
- Print all content of **ls.txt** using ***cat*** and ***more*** using "|" pipe
- Print all content of **ls.txt** using ***cat*** and ***less*** using "|" pipe
- Can you see the difference between ***more*** and ***less***?

# Redirection and Piping



- You can redirect the output of a program to a file using "**>>**" characters instead of "**>**":

1. Redirect the output of the ***ls*** command to **ls1.txt** using **">"**
2. Repeat 1
3. Redirect the output of the ***ls*** command to **ls2.txt** using **">>"**
4. Repeat 3
5. Check the difference between **ls1.txt** and **ls2.txt**

# Agenda

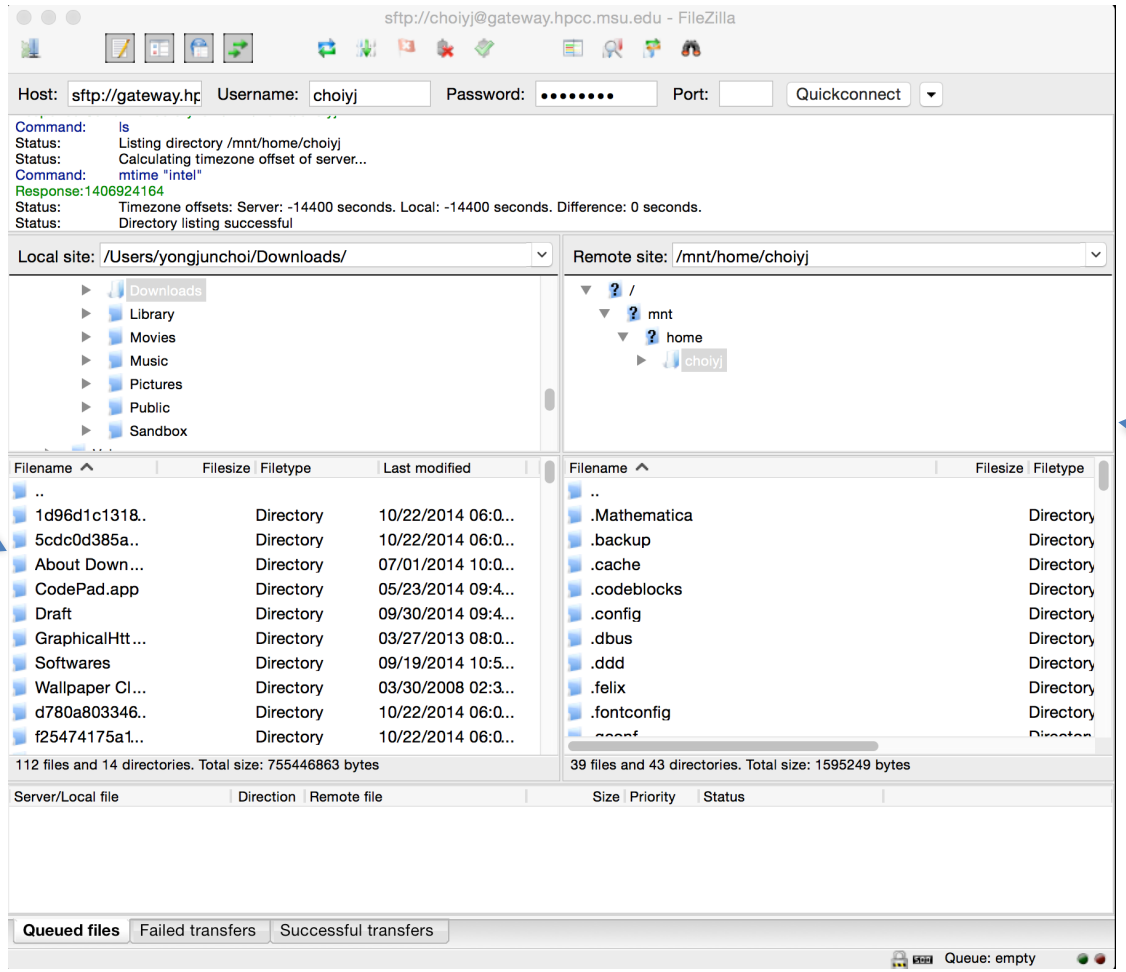
- Introduction to HPC
  - Introduction to iCER
- How to Use the HPC
  - Get an account
  - Install needed Software (SSH, X11, Xming/XQuartz)
  - Basic navigation commands
  - **Transfer files**
  - Compile/Test programs on a developer node
  - Write a submission script
  - Submitting a job

# Transferring Files (SFTP or SCP)

- Windows (iCER thumb drive)
  - Pre-installed filezilla
- OSX
  - download and install filezilla
  - http://filezilla-project.org
- Hostname: gateway.hpcc.msu.edu
- Username: msu netid
- Password: msu netid password
- Port : 22

# Transferring Files (SFTP or SCP)

LOCAL SYSTEM

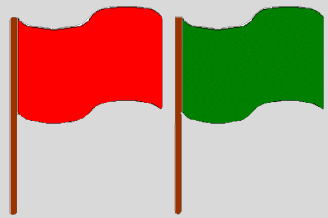



MSU's HPCC

MICHIGAN STATE UNIVERSITY

ICER

# Transferring Files (SFTP or SCP)



- Hostname: gateway.hpcc.msu.edu
- Username: msu netid
- Password: netid password
- Port : 22

Exercise: download this presentation from

**/mnt/scratch/choiyj/intro_hpcc/intro_hpcc.pdf**

to your local system

# Agenda

- Introduction to HPC
  - Introduction to iCER
- How to Use the HPC
  - Get an account
  - Install needed Software (SSH, X11, Xming/XQuartz)
  - Basic navigation commands
  - Transfer files
  - **Compile/Test programs on a developer node**
  - Write a submission script
  - Submitting a job

# Running Jobs on the HPC

- The developer (dev) nodes are used to compile, test and debug programs
- Two ways to run jobs
  - submission job scripts
  - interactive way
- Submission scripts are used to run (heavy/many) jobs on the cluster. We will be back here later.

# Advantages of running Interactively

- Yo do not need to write a submission script
- Yo do not need to wait in the queue
- You can provide input to and get feedback from your programs as they are running

# Disadvantages of running Interactively

- All the resources on developer nodes are shared between all uses
- Any single process is limited to 2 hours of cpu time. If a process runs longer than 2 hours it will be killed.
- Programs that overutilize the resources on a developer node (preventing other to use the system) can be killed without warning.

# Developer Nodes

| Names | Cores | Memory (GB) | Accelerators | Notes |
|---|---|---|---|---|
| *Dev-intel07* | *8* | *8* | | |
| *dev-gfx10* | *4* | *18* | *2xM1060* | *Nvidia Graphics Node* |
| *dev-intel0* | *8* | *24* | | |
| *dev-intel14* | *8* | *64* | | |
| *dev-intel14-phi* | *20* | *128* | *2xPhi* | *Xeon Phi Node* |
| *dev-intel14-k20* | *20* | *128* | *2xK20* | *Nvidia Graphics Node* |

# Compiling

- Most users use the developer nodes for developing their software
- If you are using a makefile you can compile using more processors with the "-j" option

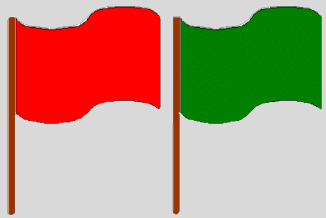
***make -j8***

  - Will compile using 8 core thread

# Compilers

- by default we use the GNU compilers. However, lots of other compilers are available including Intel and Portland compilers.
- The module system always sets environment variables such that you can easily test with other compilers.

# Compile Code



- change to “helloworld” directory
- run the gcc Compilers:

  **gcc -O3 -o hello hello.c**
- run the program:

  **./hello**

# Running in the background

- You can run a program in the background by typing an "**&**" after the command.
- You can make a program keep running even after you log out of your ssh session by using "**nohup** command"
- You can run an entire session in the background even if you log in and out of your ssh session by using the "**screen**" command
- All three of these options are common to linux and tutorials can be found online

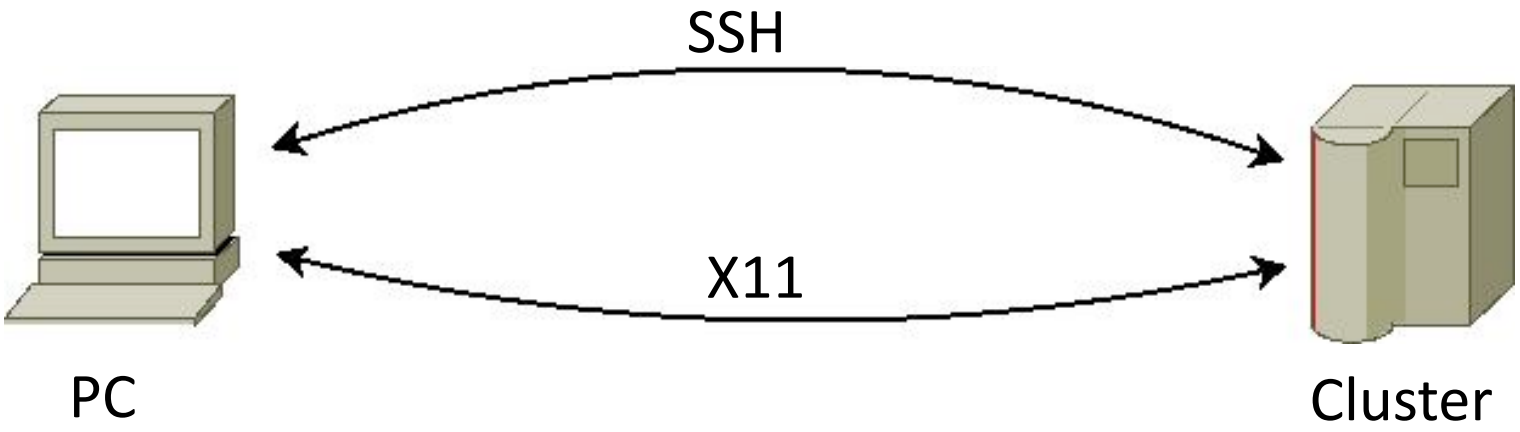
# CLI vs. GUI

- CLI: Command Line Interface

- GUI: Graphical User Interface

# What is X11?

- Method for running Graphical User Interface (GUI) across a network connection.



# What is needed for X11

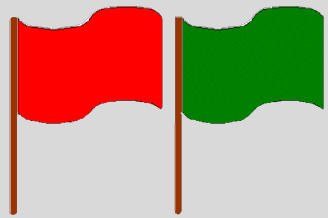
- X11 server running on your personal computer
- SSH connection with X11 enabled
- Fast network connection
  - Preferably on campus

# Graphical User Interface

- X11 Windows: Install Xming from iCER thumb drive
- ssh -X username@hpc.msu.edu
- Turn on x11 forwarding

- Note: Mac Users should use XQuartz

# Test GUI using X11



- run X11
- Try one of the following commands

***xeyes***

or/and

***firefox***

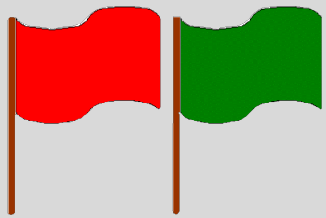
# Programs that can use GUI

- MATLAB
- Mathematica
- totalview : C/C++/fortran debugger especially for multiple processors
- DDD (Data Display Debugger) : graphical front-end for command-line debugger
- Etc, etc, and etc

# Remote Desktop Client

- RDP allows users to connect to our systems with superior performance relative to X11 forwarding over SSH.
- RDP is available at rdp.hpcc.msu.edu from on campus or via the MSU VPN.
- Windows: Microsoft Remote Desktop
- Mac: Need to install the Microsoft Remote Desktop from the App Store.
- The easiest way to connect to the HPCC via RDP is to download & run this template file: ICER HPCC RDP (https://wiki.hpcc.msu.edu/x/tABZAQ).

# Remote Desktop Client

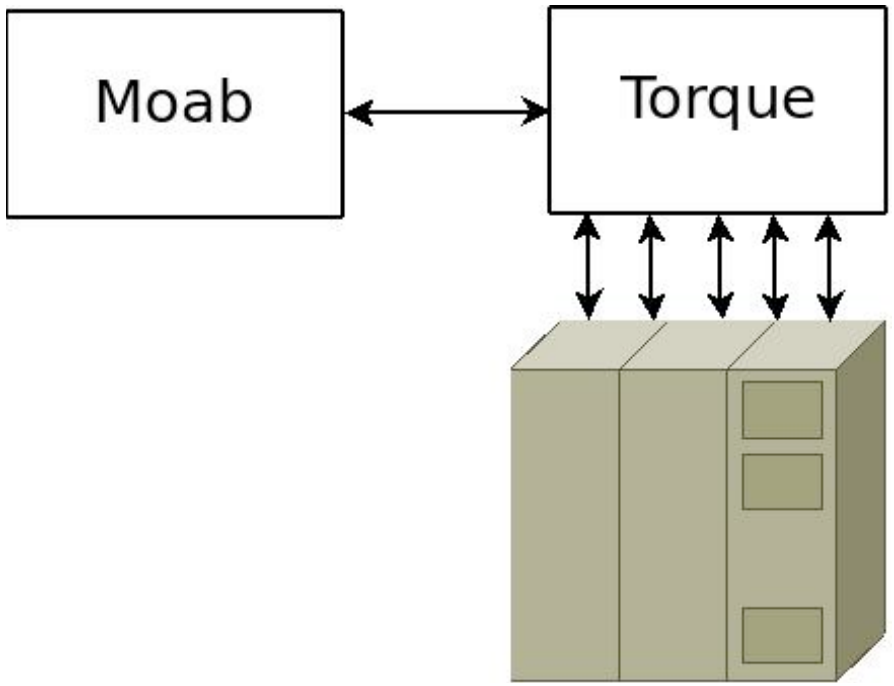


- Run RDP client and connect with the remote desktop client using ICER HPCC RDP template file.
- Open a terminal (**Application Menu => Terminal Emulator** or click the terminal icon (  ) on the upper left corner)
- Connect to **dev-intel10** and run ***xeyes***

# Agenda

- Introduction to HPC
  - Introduction to iCER
- How to Use the HPC
  - Get an account
  - Install needed Software (SSH, X11, Xming/XQuartz)
  - Basic navigation commands
  - Transfer files
  - Compile/Test programs on a developer node
  - **Write a submission script**
  - Submitting a job

# Resource Manager and Scheduler




Not First In First Out!!

# Scheduler vs. Resource Manger

- Scheduler
  **(Moab)**
  - Tracks and assigns
    - Memory
    - CPUs
    - Disk space
    - Software Licenses
    - Power / environment
    - Network
- Resource Manager
  **(PBS/Torque)**
  - Hold jobs for execution
  - Put the jobs on the nodes
  - Monitor the jobs and nodes

# Common Commands

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

# Submission Script

- List of required resource
- All command line instructions needed to run the computation

# Typical Submission Script

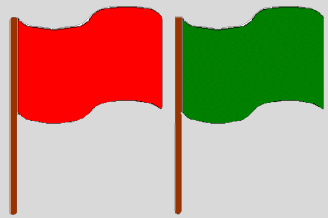
```
#!/bin/bash -login
#PBS -l walltime=10:00:00,mem=3Gb,nodes=10:ppn=1
#PBS -j oe

cd ${PBS_O_WORKDIR}

./myprogram -my input arguments

qstat -f ${PBS_JOBID}
```

# Submit a job



- go to the helloworld directory

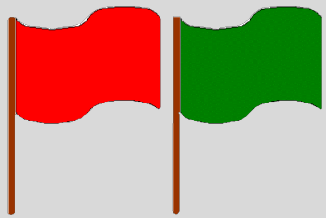
  ***cd ~/helloworld***

- Create a simple submission script

  ***nano hello.qsub***

- See next slide to edit the file…

# hello.qsub



```
#!/bin/bash -login
#PBS -l walltime=00:01:00
#PBS -l nodes=1:ppn=1

cd ${PBS_O_WORKDIR}
./hello

qstat -f ${PBS_JOBID}
```

# Details about job script

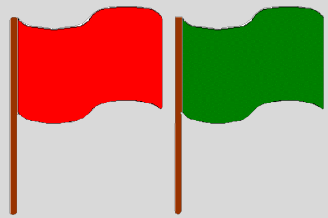
- "**#**" is normally a comment except
  - "**#!**" special system commands
  - ***#!/bin/bash***
  - "***#PBS***" instructions to the scheduler"

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

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

  ***#PBS -l mem=2gb*  (!!! Not per core but a whole)**
- http://wiki.hpcc.msu.edu/x/Np-T

# Submit a job



- 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* userid**

- Delete a job in a queue:

  ***qdel* jobid**

- Status of a job

  ***qstat –f* jobid**

# Monitoring

- Submit the file to the queue:

  ***qstat -f*** "jobid"

- When will a job start:

  ***showstart -e all*** "jobid"

# Scheduling Priorities

- NOT First Comes First Serves!
- 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:
  - Blocked, eligible or running

# Cluster Resources

| Year | Name | Description | ppn | Memory | Nodes | Total Cores |
|---|---|---|---|---|---|---|
| 2007 | intel07 | Quad-core 2.3GHz Intel Xeon E5345 | 8 | 8GB | 126 | 1008 |
| 2009 | amd09 | Sun Fire X4600 (Fat Node)<br>AMD Opteron 8384 | 32 | 256GB | 3 | 96 |
| 2010 | gfx10 | NVIDIA CUDA Node (no IB) | 8 | 18GB | 32 | 256 |
| 2010 | intel10 | Intel Xeon E5620 (2.40 GHz) | 8 | 24GB | 191 | 1528 |
| 2011 | intel11 | Intel Xeon 2.66 GHz E7-8837 | 32 | 512GB | 2 | 64 |
| | | | 32 | 1TB | 1 | 32 |
| | | | 64 | 2TB | 2 | 128 |
| 2014 | intel14 | Intel Xeon E5-2670 v2 (2.6 GHz) | 20 | 64GB | 128 | 2560 |
| | | | 20 | 256GB | 24 | 480 |
| | | 2 NVIDIA K20 GPUs | 20 | 128GB | 40 | 800 |
| | | 2 Xeon Phi 5110P | 20 | 128GB | 28 | 560 |
| | | Large Memory | 48 or 96 | 1 - 6 TB | 5 | 288 |

# Scheduling Tips

- Requesting more resources does not make a job run faster unless you a running a parallel program
- The more resources you request, the "harder" it is for the scheduler to reserve those resources.
- First time: over-estimate how much resources you need, and then modify appropriately.
- (***qstat –f ${PBS_JOBID}*** at the bottom of my scritps will give you resources information when the job is done)

# 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: 6tb
- Scheduling
  - 10 eligible jobs at a time
  - 512 running jobs
  - 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.

***#PBS –l feature=gpgpu,gbe***
***#PBS –l nodes=2:ppn=8:gpu=2***
***#PBS –l mem=16gb***

- Email address (-M)

***#PBS –M choiyj@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:

***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}**

# Softwares (Modules) again

- iCER has over 2500 software titles installed
- Not all titles are available by default
- We use "modules" to setup the software
- Some modules are loaded by default

***module list***

- To see different version:

***module spider MATLAB***

- To search, also use the module spider command

# Getting Help

- Documentation and User Manual – wiki.hpcc.msu.edu
- Contact HPCC and iCER Staff for:
  - Reporting System Problems
  - HPC Program writing/debugging Consultation
  - Help with HPC grant writing
  - System Requests
  - Other General Questions
- Primary form of contact - http://contact.icer.msu.edu/
- HPCC Request tracking system – rt.hpcc.msu.edu
- HPCC Phone – (517) 353-9309
- HPCC Office – 1400 PBS
- Open Office Hours – 1-2 pm Monday/Thursday (PBS 1440)