High Performance Computing at Shahid Chamran University of Ahvaz

Vahid M. Saffarzadeh

Clustering of Computers

Advantages:

- High performance
- Large capacity
- High availability
- Incremental growth

Applications:

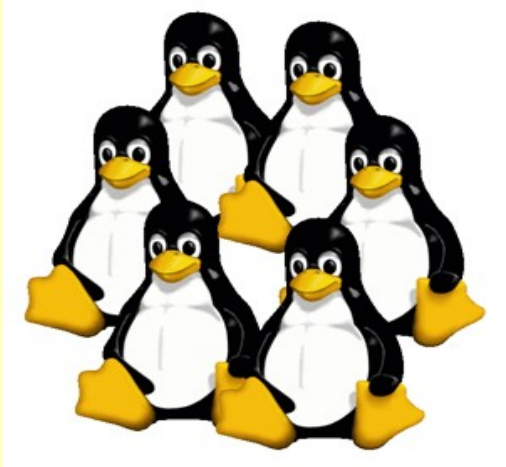
- Scientific computing
- Making movies
- Commercial servers (web/database/etc)

SCU HPC Cluster Overview

It's a starting Point:

- 5 computing nodes;
- 160 CPU cores, 32 per node;
- Designed for long-running computationally intensive jobs
- Optimized for batch jobs rather than interactive jobs
- Designed to run jobs that employ parallel processing or many single-processor jobs
- Runs Linux (CentOS Linux), Windows users should change some habits!

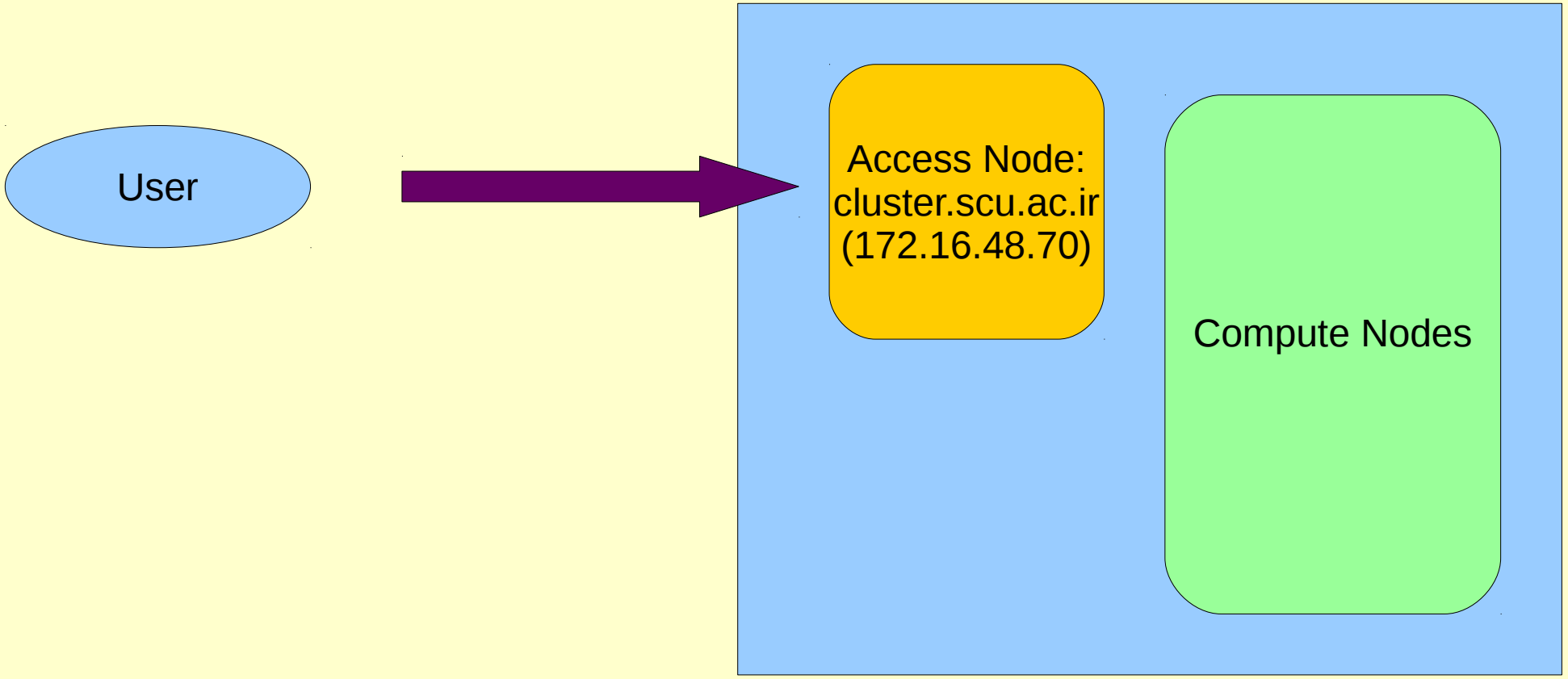
Why Linux Cluster?



- Linux runs on a wide range of hardware
- Linux is exceptionally stable
- Linux source code is freely distributed.
- Linux is relatively virus free.
- Having a wide variety of tools and applications for free.
- Good environment for developing cluster infrastructure

How to Connect

- Due to network limitations users currently can only connect at university campus and via SSH connection (no desktop).

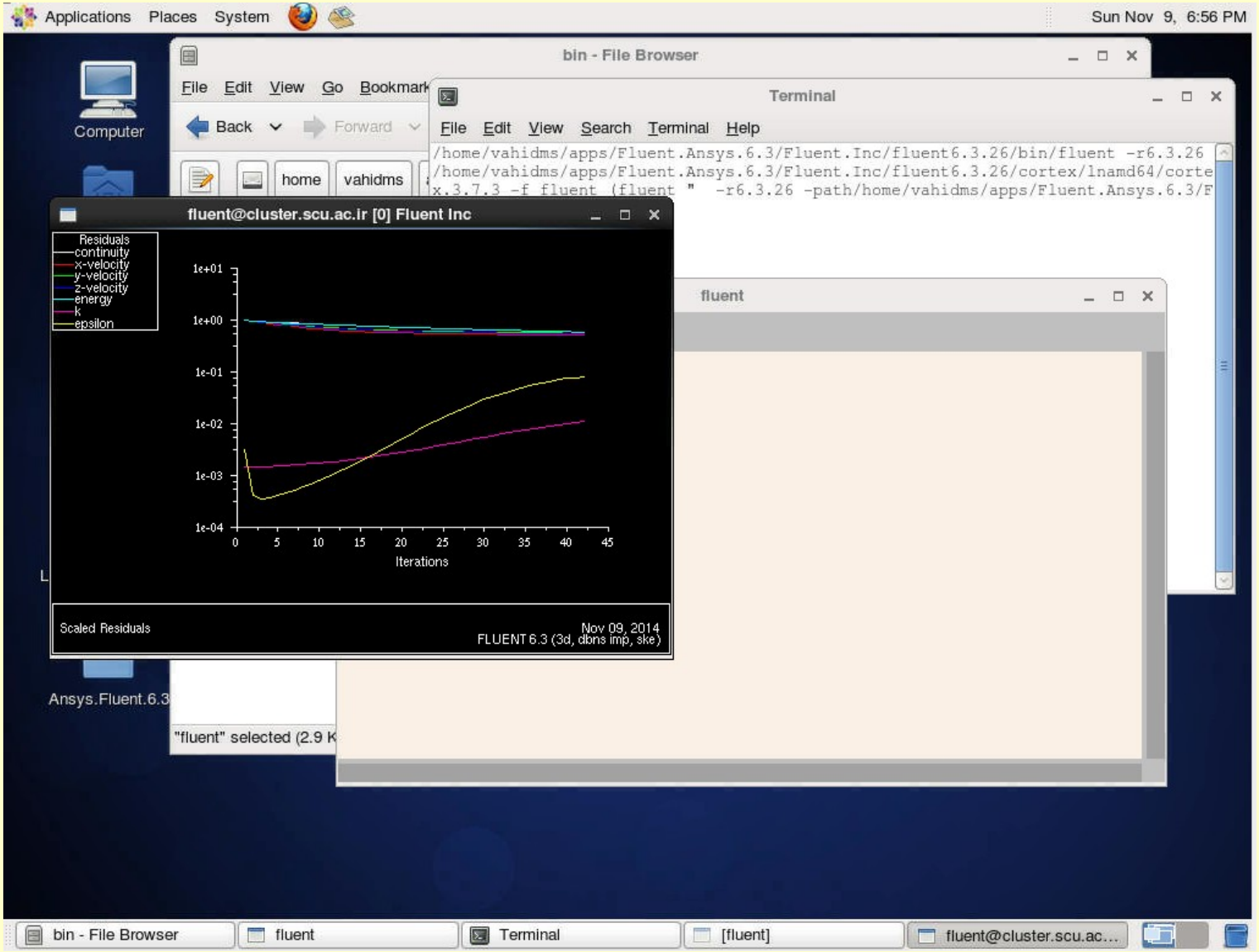


How to Use the HPC

Different types of applications can be run on the cluster:

- **Progrmans** written in C, C++, Fortran using supported compilers from GNU.
  - **Serial programs**
  - **Parallel programs** using shared parallelism (OpenMP) or distributed parallelism (MPI)
- **Pre-installed applications** including: MATLAB-like applications and R
- Molecular dynamics/Quantum: Gromacs
- Fluid dynamics: Fluent, Abaqus
- ...

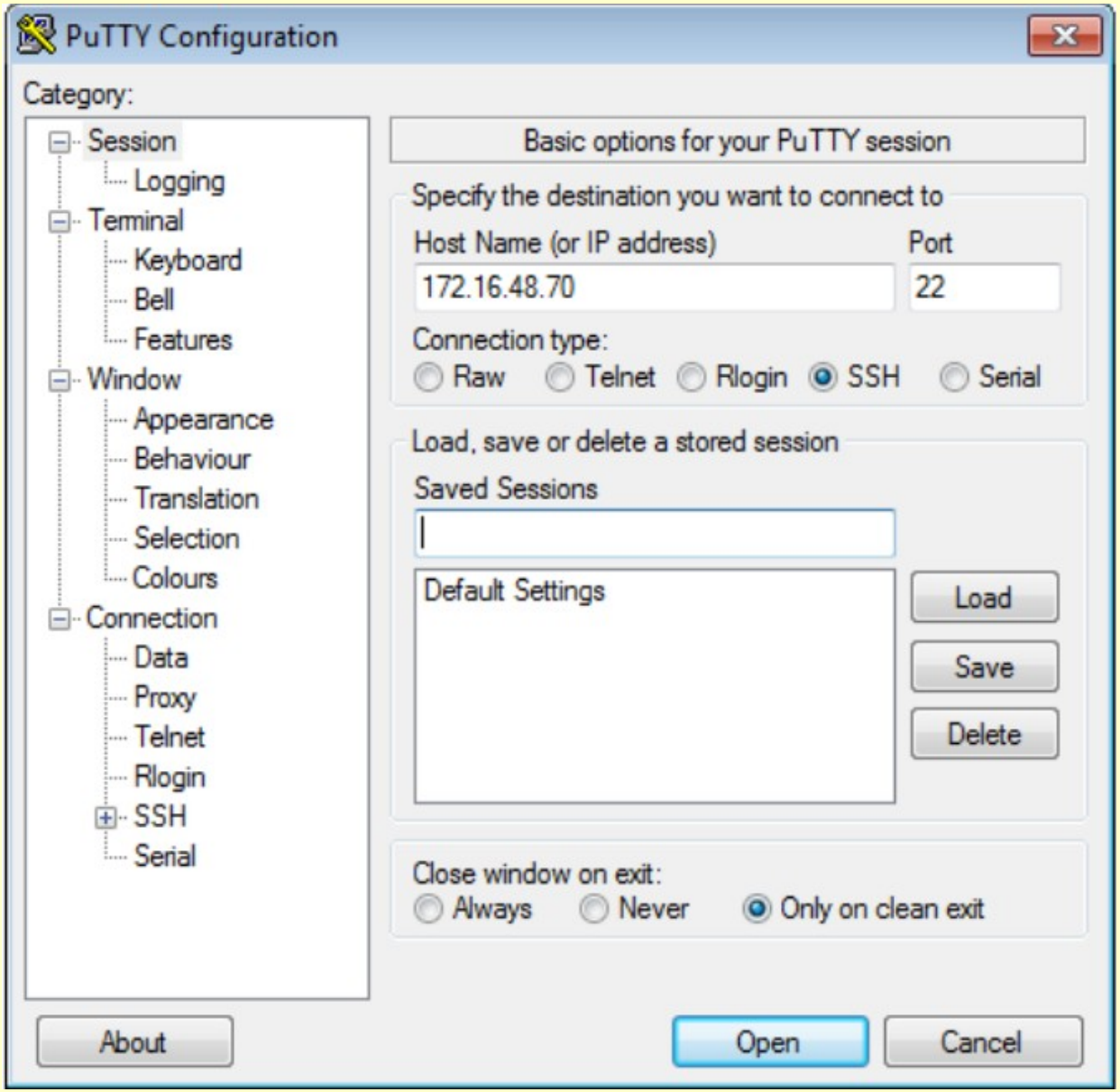
A Fluent Run Example



Logging in to the Cluster

- **Where you can login from:**
  - Campus computer (wired or via Wireless)
- **What you need to login:**
  - Use ssh utility such as Putty, cygwin (Windows) or Terminal (Mac). Telnet, RCP or Remote Desktop will not work.
  - An approved cluster account
  - Enter the UserID and the Single Sign-On password

Cluster Logging Example (for Windows users)(1)



Cluster Logging Example (for Windows users)(2)

- Once the SSH Connection is open, you should see a terminal prompt asking for your username:

  ```
  login as:
  ```

  Enter your primary account, username.

- Next, enter your password. Note that you will NOT see your any characters typed when typing your password.

  ```
  Using keyboard-interactive authentication.
  Password:
  ```

Cluster Logging Example (for Windows users)(3)

- You are now logged into the server with SSH. You see output like this:

```
username@cluster:~$
```

- Now, you are in the head node of the cluster and you can begin typing commands at the prompt.

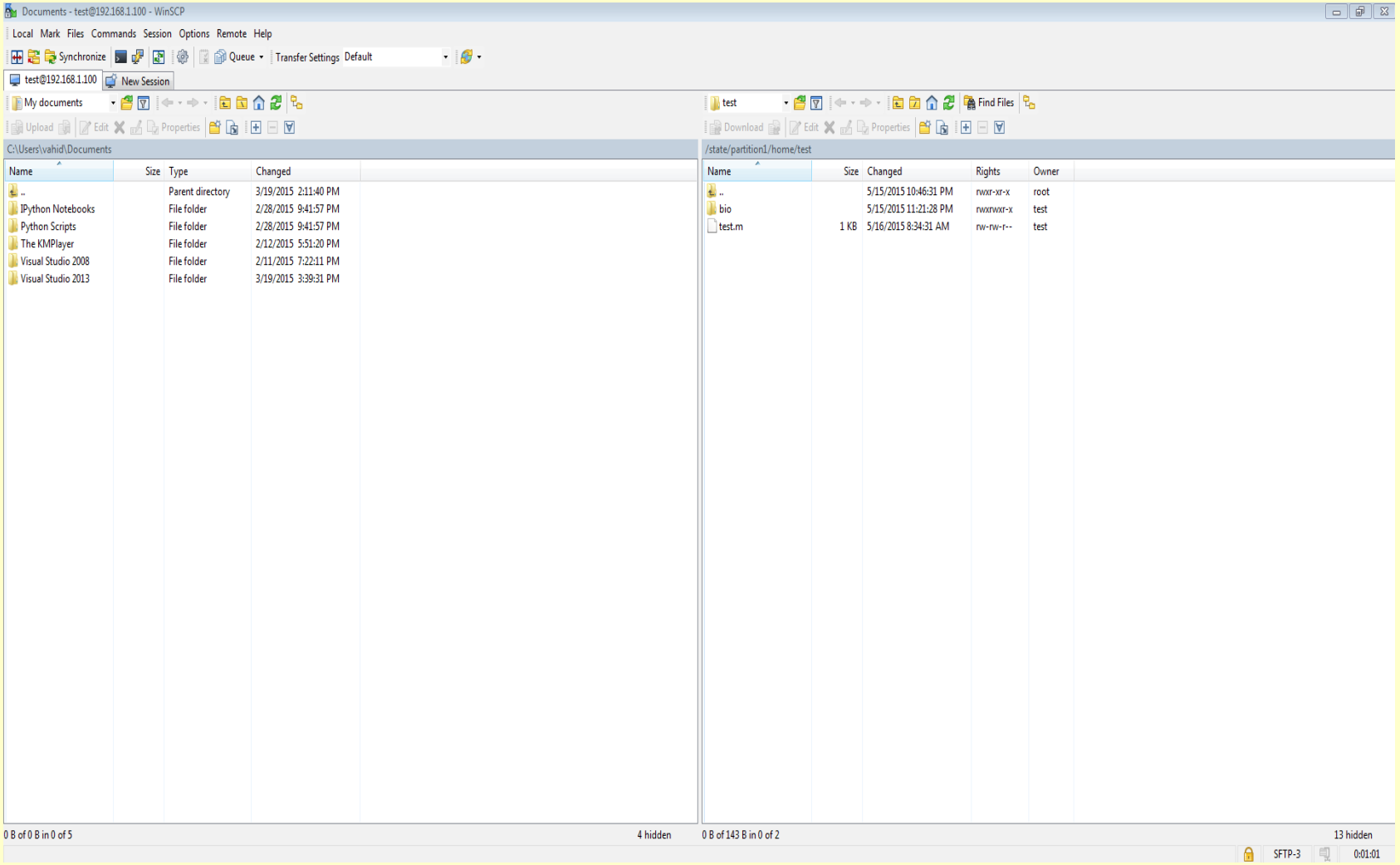
Cluster Logging example (for Linux & Mac users)

- Open a terminal or xterm. Then type this command

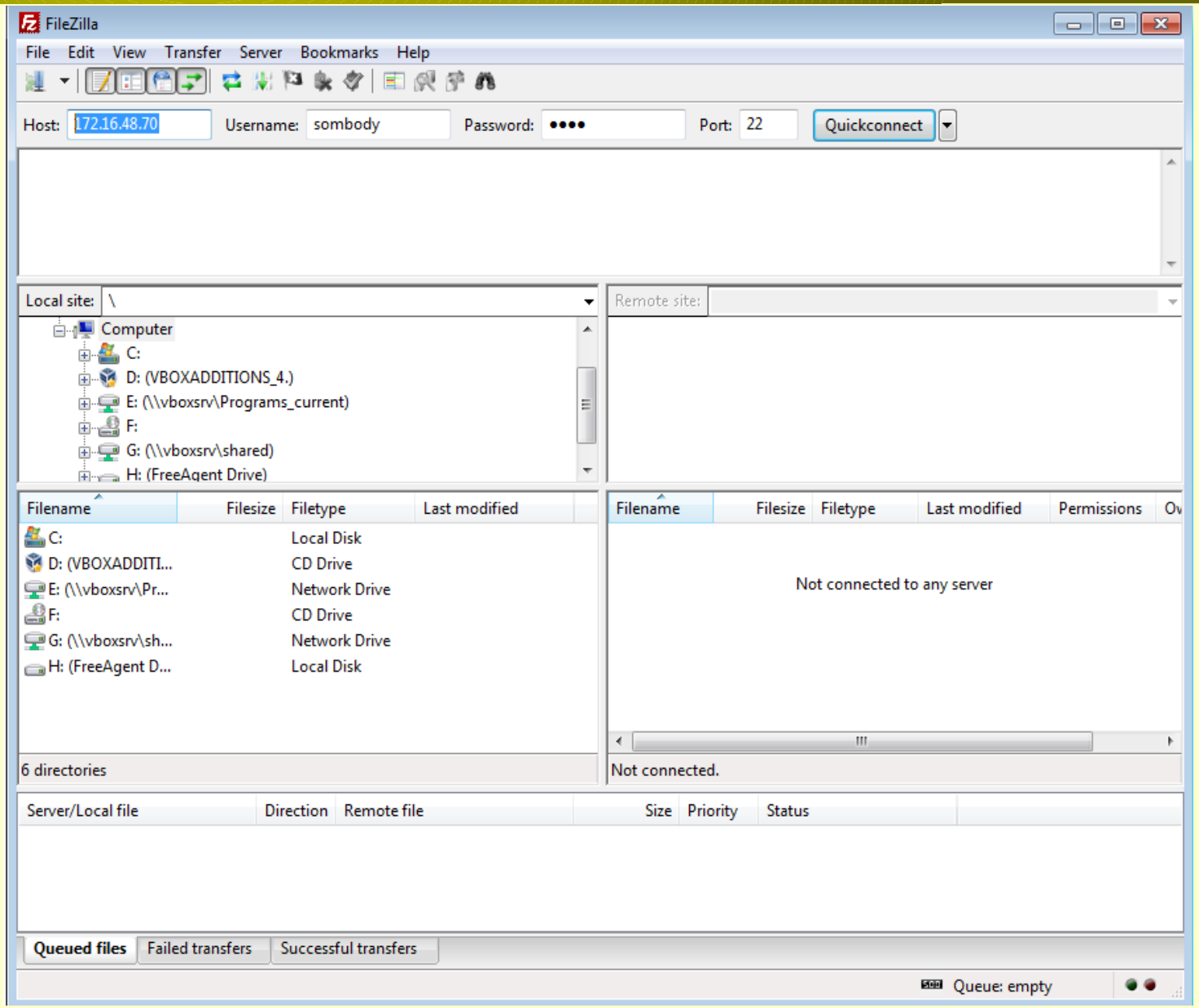
```
ssh  username@172.16.48.70
```

- After entering your password you will be logged into the head node of the cluster.

Copy files to and from cluster using winscp



Copy files to and from cluster using Filezilla



Linux Basic on the Cluster (1)

Using the Linux shell:

- Linux shell is basically where you enter the commands to manipulate files or to run a code or to execute a script

Files and directories

- The root directory is /, and the directories underneath this root are /home, /usr, /tmp, /etc
- Your home directories is located at /home/Username
- You can create subdirectories underneath your home directory (use mkdir command). Each job usually runs on its own directory
- You can create file, copy file (with cp command), move or rename files (with mv command), or remove files (with rm)

Linux Basic on the Cluster (2)

Basic Linux shell commands related to files and directories

(type man *command* for help on each commands below)

cat : to list the content of a text file	**locate**: to list all the files in the local machine with the given name
cd : to change the current directory	**ls** : to list the files and directories in the directory
cp : to copy files	**mv**: to move the files or rename them
df : to list out the disk usage of the filesystems	**pwd** : to list the current directory
less : to list the content of a text file by page	**rm** : to delete files
ln : to create a symbolic link to another file or a directory	**which**: to indicate the location of the executables/binaries

Modules and Environment
Module command

This sets up the environment necessary to run your applications (binary, libraries, shortcuts)

- To see all applications you can load in your environmnet type:

```
module avail
```

- To load a module into your current environment type:

```
module load modulename
module load octave3.08
```

Using the nano Editor

- **nano** is a text editor.
- We will use nano as an example on how to write a short Fortran code

```
cd ~/examplePrograms
nano squares.f
```

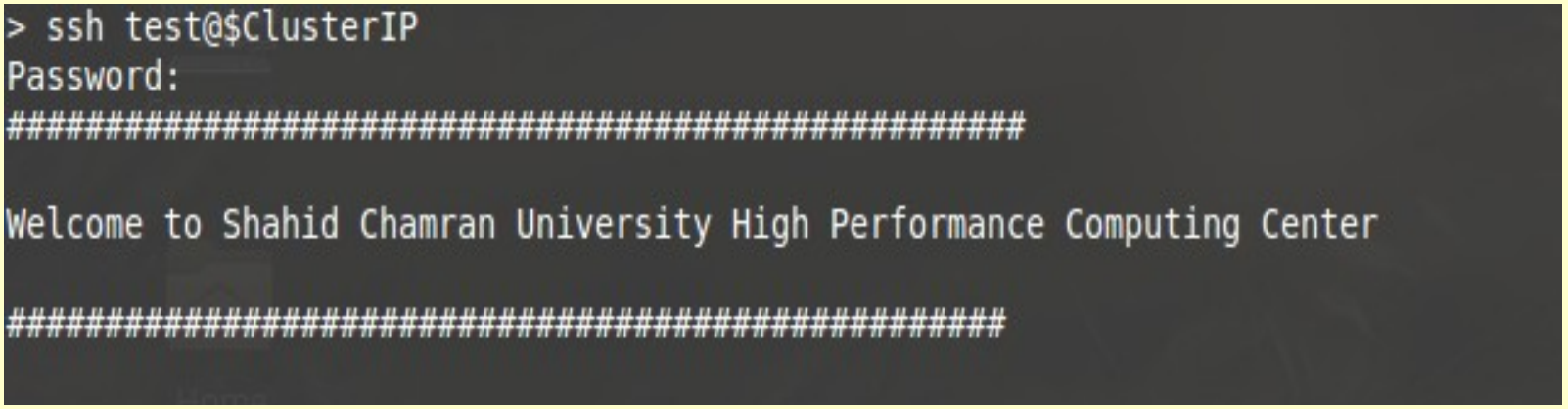
```
      PROGRAM SQUARE
        do i = 1,10
          write(*, *) i*i
        enddo
        END
```

- Press Ctrl+x and Enter to save and exit the editor

An Example of Loging and Using MATLAB (1)

- After you logged in to your account on the cluster type:

```
module avail
```



An Example of Loging and Using MATLAB (2)

- You now see a list of available modules containing MATLAB module, you select your desired version (eg. 2012a) and type:

```
module load matlab2012a
```

```
##################################################

[test@cluster ~]$ module avail

------------------------------------------------------------------------------------
dot             module-git      module-info     modules         null            op
------------------------------------------------------------------------------------
openmpi-x86_64

------------------------------------------------------------------------------------
matlab2012
[test@cluster ~]$ module load matlab2012
```

An Example of Loging and Using MATLAB (3)

- Now when you type "`matlab`" command, MATLAB application will be open in **nodisplay mode**.

```
------------------------------------------------------------------------------------------------------ /share/apps/modulefiles -------------
matlab2012
[test@cluster ~]$ module load matlab2012
[test@cluster ~]$ matlab
Warning: No display specified.  You will not be able to display graphics on the screen.

                                                                          < M A T L A B (R) >
                                                                Copyright 1984-2012 The MathWorks, Inc.
                                                                 R2012a (7.14.0.739) 64-bit (glnxa64)
                                                                           February 9, 2012


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>>
```

MATLAB Parallel Computing
An Introduction - Step1

Is your code correct and have your code ready and running in ordinary MATLAB?

MATLAB Parallel Computing
An Introduction - Step2

Prepare your ordinary-running code to become a parallel-running code to run using multiple processors. You need two main instructions in your code for this purpose:

MATLAB Parallel Computing
An Introduction - Step2(matlabpool)

- **matlabpool**

  It tells MATLAB that you want to use multiple processors in your run

  At the beginning of the code, type:

  ```
  matlabpool('open',#);
  ```

  where # is the number of cores you want to use. This instruction **opens the parallel computing toolbox.**

MATLAB Parallel Computing
An Introduction - Step2(matlabpool)

At the end of the code, type:

```
matlabpool('close');
```

This will close the parallel computing, however, the license will still be in use until you close your MATLAB session.

MATLAB Parallel Computing
An Introduction - Step2 (parfor)

- **parfor**

  parfor is the parallel computing version of the traditional "for" command for a loop. Use it in place of "for", just as simple as that:

```
parfor i = 1 : 100
    (your code)
end
```

  The loop variable (i) can only be a vector of consecutive integers. Also, the code inside this loop is the portion of your code that will be executed in parallel.

Check “MATLAB help” for particular conditions for both “matlabpool” and “parfor”.

MATLAB Parallel Computing
An Introduction – Step3 – Some Tips(1)

- When using "parfor":
  - all the executions of the loop need to be independent from each other.
  - "parfor" loop doesn't even execute consecutively (i = 1, 2, 3, 4, …), but quite randomly (i = 3, 4, 59, 37, …) **decided by MATLAB**.
  - The "parfor" loop can only be done for one level, i.e., you cannot have a "parfor" loop inside another "parfor" loop.

MATLAB Parallel Computing
An Introduction – Step3 – Some Tips(2)

- To have "for" loops inside the "parfor" loop, you need to pre-allocate all the variables that grow inside the "for" loops.

- The best way to get your results out is to print to file and save data inside the "parfor" loop, this is also a good way to take results out of the “parfor” loop.

A Simple MATLAB Code

```
matlabpool ('open',2);   % Call to open the distributed processing
x = zeros(5,10);         % Initialize the main variable
parfor i = 1:5           % Parallel loop
    y = zeros(1,10);     % Initialize the secondary variable
    for j = 1:10         % Inner loop
        y(j) = i;
    end
    y                    % Display the inner variable
    x(i,:) = y;          % Get values from loop into the main
variable
end
x                        % Display main variable
matlabpool close;        % Close the distributed computing
```

A Simple MATLAB Code - Result

```
Starting matlabpool using the 'p1' profile ... connected to 2 labs.

y =

     2     2     2     2     2     2     2     2     2     2


y =

     1     1     1     1     1     1     1     1     1     1


y =

     4     4     4     4     4     4     4     4     4     4


y =

     3     3     3     3     3     3     3     3     3     3


y =

     5     5     5     5     5     5     5     5     5     5


x =

     1     1     1     1     1     1     1     1     1     1
     2     2     2     2     2     2     2     2     2     2
     3     3     3     3     3     3     3     3     3     3
     4     4     4     4     4     4     4     4     4     4
     5     5     5     5     5     5     5     5     5     5

Sending a stop signal to all the labs ... stopped.
>>
```