Introduction to IT for UCL Astrophysicists

L. Whiteway, T. Wilson

Department of Physics and Astronomy
University College London

13 & 20 October 2016

Where to find this presentation

url

`https://github.com/Astrophysics-UCL/HPCInfo/tree/master/training/workshop_2016`

Overall goals of presentation

- What software you might find useful
- Where to get more information (UCL courses, web, etc.)
- UCL-specific information (e.g. login details)
- Some hands-on work

Specific contents

13 October:

- Accessing Astrophysics group machines
- Using the Linux console
- Basics of Python

20 October:

- Commonly used programs (LaTeX, DS9, IRAF,...)
- Using High-Performance Computing (HPC) machines
- HPC best practices

Information on the Web

Astrophysics Wiki

`https://wiki.ucl.ac.uk/display/PhysAstAstPhysGrp/Main+Page`

This Wiki is freely viewable and editable by all members of the department. Please use it to record information that you think will be useful to others (including your future self). Be bold!

UCL Research Computing Platforms

`https://wiki.rc.ucl.ac.uk/wiki/Main_Page`

Stack Overflow

`http://stackoverflow.com/`

Computing Environment for Astrophysics

- Large datasets requiring substantial processing followed by sophisticated statistical analysis
- Calculations often done on specialised 'high-performance computing' (HPC) machines having large filesystems and large RAM; calculations are often broken into pieces that can be run simulataneously ('in parallel') across many processors.
- Much useful software is made freely available within the community. Software quality is usually high; documentation quality is more variable. As well, many users write their own software.

Local Computing Environment

You will have your own local machine, which will have one of these operating systems:

- PC (Windows)
- Mac
- Linux

In addition there are shared Linux machines:

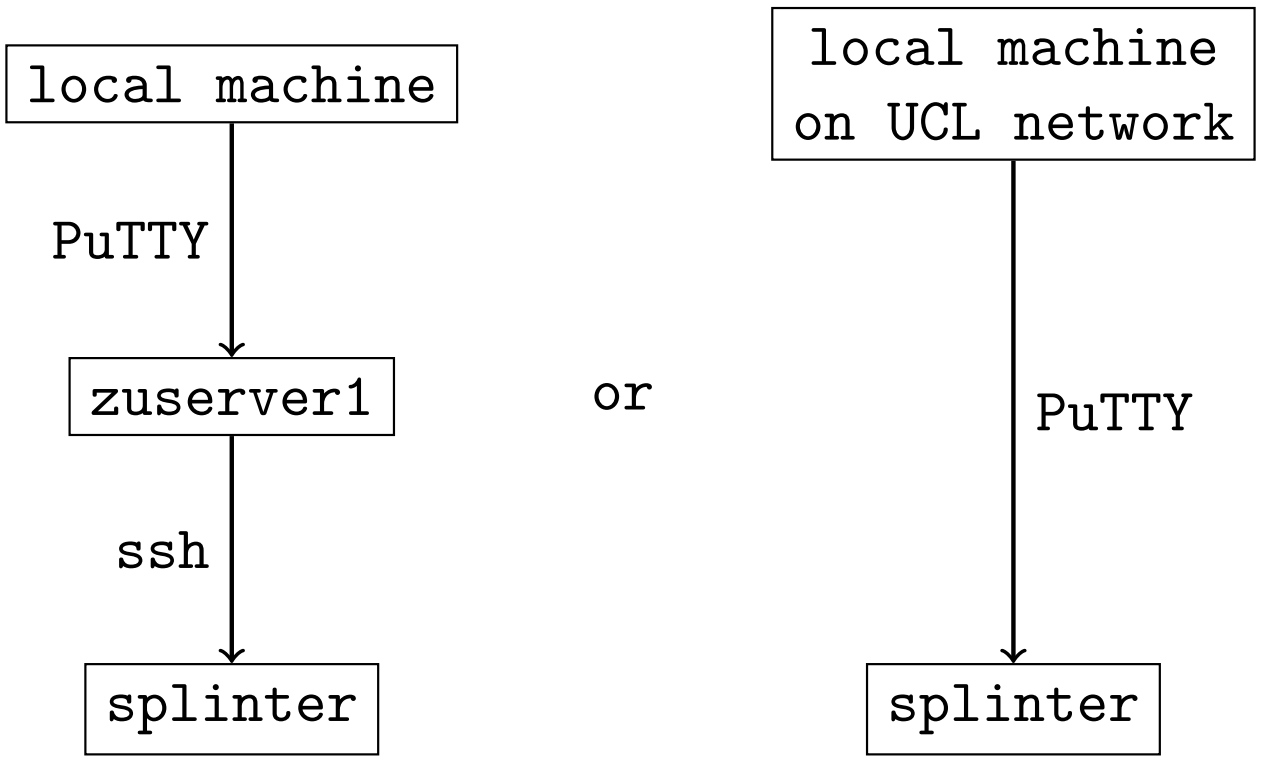
- zuserver1 (general purpose; accessable from outside UCL)
- splinter (HPC cluster)
- Others? Legion?

Work patterns

Several work patterns are possible:

- Write and test a program on your local machine; use the local machine to remotely connect to splinter; upload the program to splinter and run it there;
- Or do all your work locally (requires small data sets);
- Or use the local machine to remotely connect to splinter and do all your work there.

Accessing splinter



Accessing splinter

- You will need a *username* and *password* for splinter (and perhaps zuserver1).
- Step 1: Logon to zuserver1:
  - From Windows: Use PuTTY (`http://www.putty.org/`). On the 'Connection/SSH/X11' tab, click on 'enable X11 forwarding' and set 'X display location' to 'localhost:0'. On the Session tab, set the Host Name to zuserver1.star.ucl.ac.uk.
  - From Mac: ?? Find out what to do
  - From Linux: ?? Just do ssh?
- Step 2: From zuserver1 you can logon to splinter: `ssh -YC username@splinter-login.star.ucl.ac.uk`.
- If you are connecting from a machine on the UCL network then in step 1 go straight to splinter-login.star.ucl.ac.uk and omit step 2.
- Say something about X Windows forwarding.

Command shell in Linux

- In Linux you will use a 'command shell'.
- This is a text-based environment in which you type commands and receive text output.
- Not GUI! Reflects the hardware limitations current when Unix was created. Low-tech and reliable e.g. for remote access.
- Various command shell programs in use: *bash*, *csh*, *tcsh*,...

Directory structure

- Everything is organised around files (which may be data files or program files i.e. instructions to be executed).
- Files live in directories. There is a hierarchical tree structure of directories.
- Sample file name: /share/splinter/ucapwhi/des/foo.txt
- Note use of slash '/', not backslash '\' as in Windows.
- Case sensitivity: 'Foo' and 'foo' are different strings.

Special symbols for directories

Symbol	Meaning
/	Top of the directory tree (the root directory)
.	Current directory
..	Parent of the current directory
~	User's 'home' directory

Structure of commands

Structure

```
[command] -[option(s)] [argument]
```

Examples

```
ls -la
mkdir hello_world
cp hello.cpp new_hello.cpp
```

Linux command reference

There is a very useful summary of Linux commands at:
`http://www.computerhope.com/unix.htm`

Basic Linux commands I

navigation and help

```
ls -la
cd dir_name
man command_name
pwd
exit
```

copy or move

```
cp src dest
mv src dest
scp usr@host:file dest
```

create or delete

```
touch file.txt
mkdir dir_name
rm -i file.txt
```

find and system info

```
whereis file
which
echo $VAR_NAME
```

file contents

```
cat file
more file
head file
```

Basic Linux commands II

& ; | ¿ ¡

& (background)
; (combine)
\ (next line)
| (combine)
* (wildcard)
> (output)
< (input)

Text editors

emacs
vi
gedit

process management

kill
top
nohup

compressed files

gunzip
tar

images

gthumb
ds9

publishing

latex
bibtex

Exercises I

1. Go to your home directory and create a directory called `linux_hpc_workshop`.
2. Change directory to `linux_hpc_workshop`.
3. Find the name of the present working directory.
4. Make a directory `level_1/level_2`, and move to `level_1/level_2` in one command.
5. Move back to the previous directory.
6. Remove the directory `level_1` (and its contents).
7. In the current directory make a symbolic link to `usr/lib` called `my_sybolic_link`.
8. Create a file called `foo.txt` with contents "This file contains the word foo".
9. Add another line in `foo.txt` with contents "This is the second line".
10. Check to see if it worked.
11. Search for the phrase *foo* in `foo.txt`.

Exercises II

1. Find the location of your python installation.
2. Find the installation location(s) of `liblapack.a`.
3. Find whether an object `daxpy` is in `liblapack.a`.
4. Find the value the environment variable `PATH` and `LD_LIBRARY_PATH`.
5. Set the environment variable `MY_LINUX_HPC_VAR` to equal the absolute path to `linux_hpc_workshop`.
6. Add (i.e append) to the `PATH` the absolute path to `linux_hpc_workshop`.
7. Use the *source* command do the last two steps from source file.
8. Use the *man* command to find the option of `ls` that shows the output in Kilobyte,Megabyte.

Exercises III

1. Find hostname, processor type and operating system version and write this info into a text file called `info.txt`.
2. List the people who are currently logged into the system.
3. Find the process that is taking most of the CPU at the moment.
4. Find the IDs of the processes that you are running.
5. Make a directory called `to_be_compressed`. Add the files `hello.cpp` and `hello.py` in this dir. Then compress this directory using *tar* and *zip*.
6. Delete the directory `to_be_compressed` and extract the files from `to_be_compressed.tar.gz`.
7. Use *wget* to download files from `ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/cfitsio3370.tar.gz`.
8. Find the size of the item you just downloaded in MB.
9. Extract all files from this downloaded archive file.
10. In the extracted files, find all occurrences of `ffopentest` in all the files with extension `.h`.
11. Remove all the files with extension `.h`.
12. Copy the files with extension `.c` into a new directory `c_files`.

Information on the Web

Documentation

`http://scipy.org/`
`http://matplotlib.org/`
`http://www.astropy.org/`

SciPy Tutorials (Also NumPy and Matplotlib)

`https://conference.scipy.org/scipy2013/tutorials.php`

SciPy Lectures (Also NumPy and Matplotlib)

`http://www.scipy-lectures.org/`

Stanford's Introduction to Scientific Python

`http://web.stanford.edu/~arbenson/cme193.html`

Python

Base Python

Dictionaries, functions, classes?

Numpy

Basic statistics, numpy arrays, slicing, sorting, matrices?, masked arrays?, I/O

Scipy/Astropy

constants, more stats, fitting, interpolation, pyfits, world coordinate systems, symbolic calculus

Matplotlib

different styles, image plotting, contour?, multiple subplots

Common and Useful Programs

LaTeX

Talk about Tikz in here

DS9

Introduce and show various features including aligning, panning, colour sets, scaling (also using right click), blinking, regions and annotating (producing a finder chart?)

IRAF

Introduce modularity and how to get help (?) and edit parameter files (maybe do a quick aperture photometry demonstration?)

Information on the Web

This presentation

`https://github.com/Astrophysics-UCL/HPCInfo/`

Splinter on the UCL Astrophysics Wiki

`https://wiki.ucl.ac.uk/display/PhysAstAstPhysGrp/Splinter+User+Guide`

UCL Research Computing Platforms

`https://wiki.rc.ucl.ac.uk/wiki/Main_Page`

DiRAC

`http://www.dirac.ac.uk/`

Mailing list

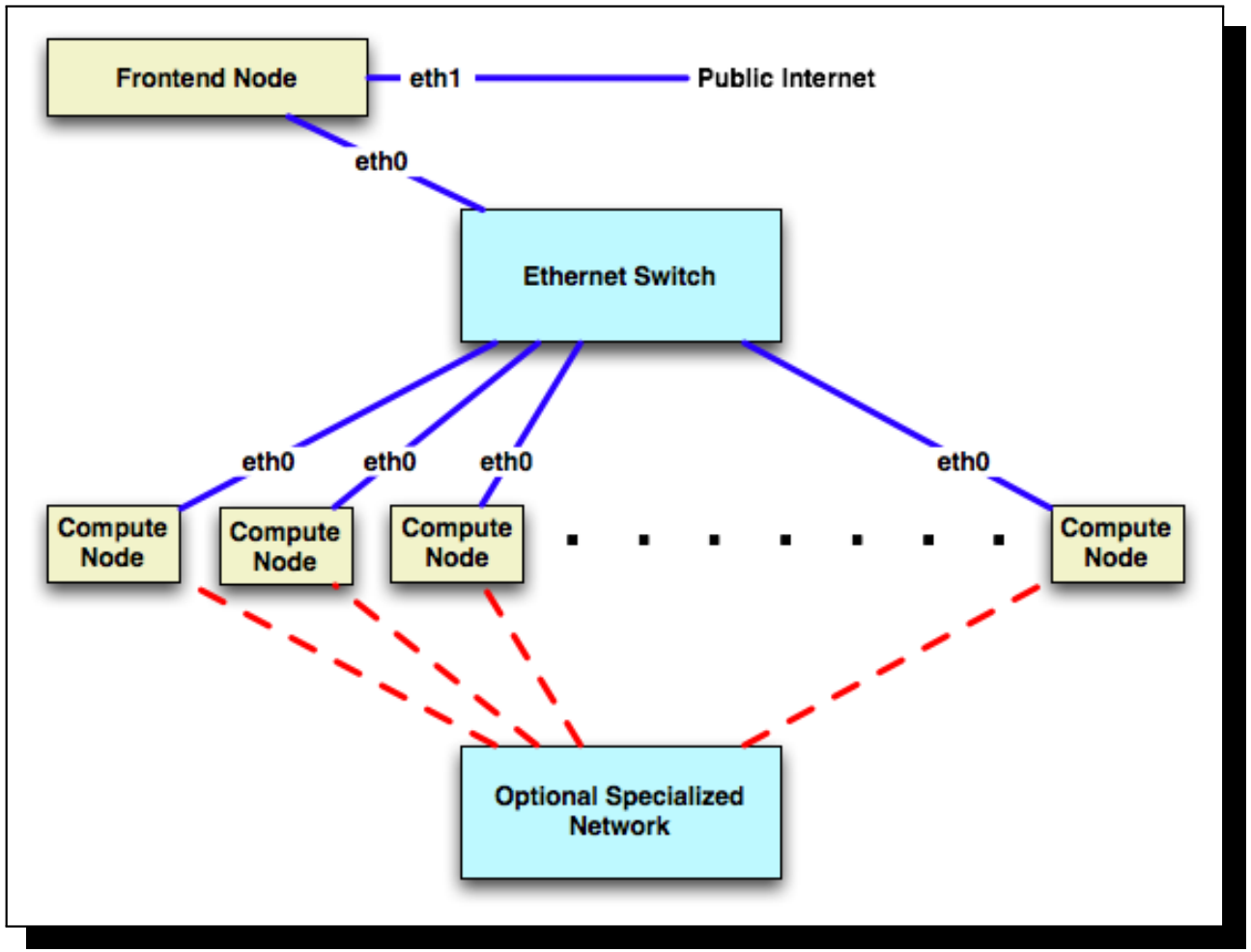
`https://www.mailinglists.ucl.ac.uk/mailman/listinfo/splinter-users`

- please subscribe
- post any issues regarding splinter

Splinter specs

- As of October 8, 2016, *Splinter* has 528, 4TB memory
- 8 nodes, dual 6-core 2.8GHz, 48GB memory
- 20 nodes, dual 8-core 2.0GHz, 128GB memory
- SMP node, 40 2.4GHz cores, 1TB memory
- login node, dual 10-core, 2.4GHz 98GB memory
- head-node, dual 8-core, 2.4GHz, 164GB memory

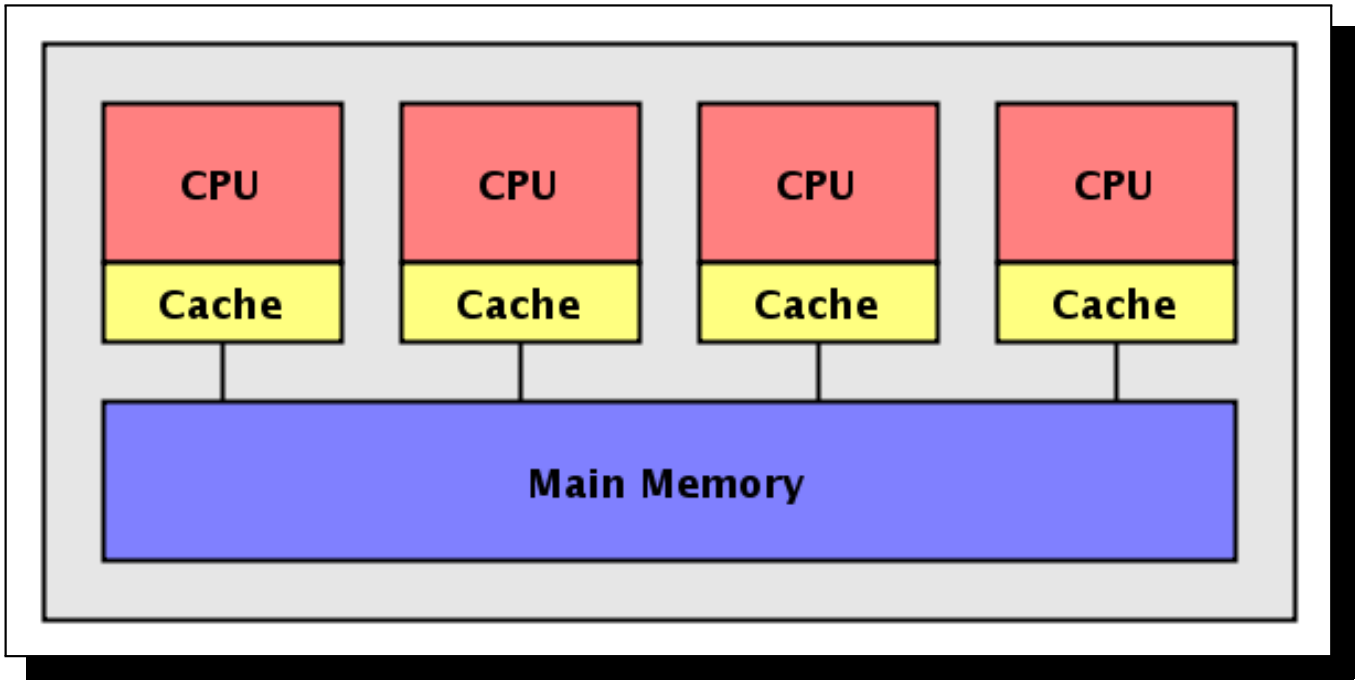
SPLINTER distributed



[1]

[1] http://www.rocksclusters.org/

SPLINTER shared



2

[2] http://www.cs.rit.edu/

Workspaces I

/home/user_name

- this is your home directory
- login scripts can be put here
- 1GB quota
- private

/share/splinter/user_name

- create the directory if not already there
- can be used as a workspace
- no quota
- public unless made private

Workspaces II

`/share/data1`

- for storing large data
- you can create a directory for your, .e.g, `/share/data1/SKA`

`/share/apps`

- for installing software
- module-files

Login script

- everytime you login this file will be executed
- this file is in your `$HOME`
- it is called `.login`
- you can load modules, envvars, etc.

Examples

Load my aliases

```
source ~/aliases.csh
```

Load python

```
module load dev_tools/nov2014/python-anaconda
```

Modules

- easy and flexible way use software
- available to everyone in splinter

Examples

Print the available modules

```
module avail
```

Load a module

```
module load module_name
```

List the loaded modules

```
module list
```

Unload a module

```
module unload module_name
```

Unload all modules

```
module purge
```

Help

```
module --help
```

Submitting jobs

- computing jobs should be submitted to the scheduler
- you will have to write a job script
- interactive job

Examples

Submit a job

```
qsub job_script
```

Submit an interactive job

```
qsub -I
```

Check the status of a job

```
checkjob job_id
```

List the status of all jobs

```
qstat
```

Show the queue

```
showq
```

Delete a job

```
qdel job_id
```

Queues

- `compute`
- `cores16`
- `cores12`
- `smp`

Structure of a job script

```
#!/bin/tcsh
# PBS -q cores12
# PBS -N a_name_for_your_job
# PBS -l nodes=1:ppn=6
# PBS -l mem=32gb
# PBS -l walltime=120:00:00
```

Set some environment variable

```
setenv OMP_NUM_THREADS 6
```

Source paths if needed

```
source /home/username/libpaths.csh
```

Run my program

```
/home/username/hello_world.exe
```

Jobscripts: things to remember

- Submit the job to the right queue
- Request the correct number of `nodes` and `ppn`
- Specify the memory required
- Always specify the walltime
- If your program is not parallel, please use `nodes=1,ppn=1`
- Use `-q compute` for single processor jobs
- Use `qsub -I` for interactive job
- If using most of the resources, please send an email to the mailing list.

More PBS commands

Specify output

```
PBS -o path/to/file.out
```

Specify error output

```
PBS -e path/to/file.err
```

Mail alert at (b)eginning, (e)nd, and (a)bortion of execution

```
PBS -m bea
```

Send mail to the following address

```
PBS -M your_email_id@ucl.ac.uk
```

Using *Ganglia*

`http://splinter.star.ucl.ac.uk/ganglia/`

- is tool for analysing splinter
- can only be loaded from splinter (using firefox)
- will give you load/memory information
- can look into nodes

Collaborative projects

- collaboration between two splinter users
- can share common data in `/share/data1/my_collaboration`
- give read/write permission to other users using `chmod`

Best practices

- Choose the machines that are suited for your problem
- Read the User Guide
- Do not run your programs in the login node
- Install common software locally if and only if absolutely necessary
- Request optimum resources
- Minimise data transfer between nodes,
- Backup! Backup! Backup!

Exercises

https://github.com/Astrophysics-UCL/HPCInfo/tree/master/training/workshop_2016/