HPC Workshop

S. T. Balan, A. Palmese,D. Saadeh

Department of Physics and Astronomy
University College London

22 October 2015

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 subscirbe
- post any issues regarding splinter

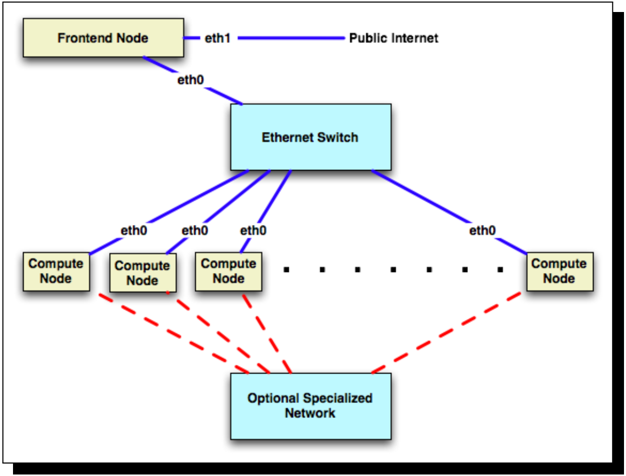
What will you learn?

- Running your programs in HPC machines
- Best practices

Splinter specs

- As of October 21, 2015, *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

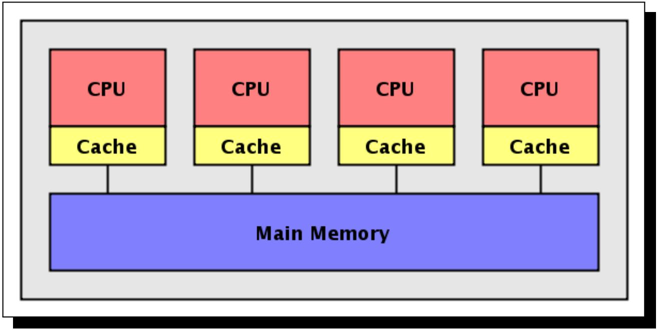
SPLINTER distributed



[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

`/share/splinter/user_name`

- can be used as a workspace
- no quota

Workspaces II

/share/data1

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

/share/apps

- for install 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.

example

```
# load my aliases
source ~/alisas.csh

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

Modules

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

commands

```
# 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 scheudler
- you will have to write a job script
- interactive job

commands

```
# submit a job
qsub jon_script
# sumit an interative 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_hob
#PBS -l nodes=1:ppn=6
#PBS -l mem=32gb
#PBS -l walltime=120:00:00
#PBS -j oe
#PBS -V

# set some environment variables
setenv OMP_NUM_THREADS 6

# source paths if needed
source /home/sbalan/libpaths.csh

# run my program
/home/sbalan/hello.exe
```

Using *Gaglia*

`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

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 resouces
- Minimise data transfer between nodes,
- Backup! Backup! Backup!

Exercises I

1. Login to your HCP machine and find the path to your `HOME` directory and your quota
2. Find the processor type and the version of your operating system
3. Request an interative queue and run the program `hello_world.exe`
4. Sumbit `hello_world.exe` using a job script, find its `jobid`, check the output log.
5. Compile `big_mem_example`, submit it using a `job-script` and see how much menory it uses
6. Compile `time_pause_example`, submit it using a `job-script` and kill this job using its `jobid`.
7. In the previous example see what happens when you play with the time requested.