ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

LHCb

LPHE meeting, 1st Feb 2018, EPFL, Switzerland

LUCA PESCATORE

COMPUTING @ LPHE

Outline

- The LPHE server and websites
- EasyComment: the way we comment papers
- The LPHE cluster:
  - ✓ Structure
  - ✓ SLURM
  - ✓ submit.py
  - ✓ MC production
  - ✓ Best practices

The LPHE server

- The LPHE server is accessible as

  ssh user@lpheserv2.epfl.ch

- No password is needed. Access is manages via ssh key.

- You have a /home/public_html folder. You can put files or html into it and it will be accessible online as

  http://lphe.epfl.ch/~user

- Files for the Monday meeting should be put into /users/LHCbPhysicsMeetings and will be available at http://lphe.epfl.ch/LHCbPhysicsMeetings/

EasyComment

- A framework is now available to make comments to papers and notes in an interactive and collaborative way.

- Just go to: https://pluca.web.cern.ch/pluca/EasyComment/
- Then select the paper to comment on from the list or create a new one

- N.B.: Sometimes the page blocks. Reloading usually fixes it.

- You are not forced to use it but it's good if you do
- Feel free to report bugs or ask for improvements.

Cluster structure

Access nodes

lphelc1a lphelc1b

Work nodes
Queue: batch

lphe1 lphe2 … lphe20

Test nodes
Queue: debug

lphe21 lphe22

16 cores and 29 GB memory per node

Storage: /home and /panfs (for large storage)

Guides

Guides explaining all that is explained here too are available at

http://lphelcsrv2.epfl.ch/cluster/index.php

“Guides” link

How to use LHCb software

- On the cluster CVMFS is available which means that you can use LHCb software as if you were on lxplus.
- To setup:
  - Script are provided ".localrc" and ".bash_profile" provided at this link
  - Put these files in your cluster home. (Take care of not overwriting anything!)
  - Re-login and the LHCb login should automatically appear.

- The only thing that does not directly work is reading from EOS.
- To make it work you have to
  - Initialise a proxy

    ```
    export X509_USER_PROXY={Some filename where your certificate will be stored}
    lhcb-proxy-init -cert $X509_USER_PROXY
    chmod 750 $X509_USER_PROXY
    ```

  - Prepend "root://eoslhcb.cern.ch//" to your paths.

SLURM

- SLURM is the batch system of our cluster.
- You can find a quick start guide about how to use it here.
- Information on how to submit jobs and best practices are included in the next slides.

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

        {Write here your bash code as you'd normally do.}
```

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

        {Write here your bash code as you'd normally do.}
```

Output logs

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1
#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

        {Write here your bash code as you'd normally do.}
```

Output logs

Job name

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

        {Write here your bash code as you'd normally do.}
```

Output logs

Job name

Minimum number of nodes reserved

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

        {Write here your bash code as you'd normally do.}
```

Output logs

Job name

Minimum number of nodes reserved

Cores reserved per node

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

{Write here your bash code as you'd normally do.}
```

Output logs

Job name

Minimum number of nodes reserved

Cores reserved per node

Partition: batch, debug

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

{Write here your bash code as you'd normally do.}
```

Output logs

Job name

Minimum number of nodes reserved

Cores reserved per node

Partition: batch, debug

Summary mail

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

{Write here your bash code as you'd normally do.}
```

Output logs

Job name

Minimum number of nodes reserved

Cores reserved per node

Partition: batch, debug

Summary mail

Time after which job is killed

Typical SLURM script : my_job.sh

```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out
#SBATCH -e outputs/name.err
#SBATCH -J some_name

#SBATCH -N 1
#SBATCH -n 1

#SBATCH -A batch
#SBATCH -p batch

#SBATCH —mail-user user@sever.something
#SBATCH —mail-type ALL

#SBATCH -t 00:15:00
#SBATCH —mem-per-cpu 1000

{Write here your bash code as you'd normally do.}
```

Output logs

Job name

Minimum number of nodes reserved

Cores reserved per node

Partition: batch, debug

Summary mail

Time after which job is killed

Memory reserved per core

Run a SLURM script

```
sbatch my_job.sh

or

srun [options] myprogram
```

Other commands:

- squeue → to check the queues
- scancel → to kill a job
- scontrol show jobid -dd <jobid> → to show info

Let me help you

- First of all if any problem feel free to come to me
- A script is provided to help you: /home/pescator/setup/submit.py

```
python submit.py -d myjob -n subjob my_command
```

What will it do :

✓ It will automatically create a standard sbatch script

✓ Lets you choose if to run in batch or local mode (-- local)

✓ Automatically creates folders for your jobs where the launched configuration is saved.

✓ Works also on lxplus LSF

✓ Can actually do much more

```
|-jobs
|   |-myjob
|   |   |-subjob
|   |   |   |-test.sh
|   |   |   |-run.sh
|   |   |   |-out
|   |   |   |-err
```

The script you'd normally launch locally

sbatch script

submit.py

- I suggest an alias: alias submit='python path/to/submit.py '
- You can define an environment variable $JOBDIR to define the main folder for the jobs. Otherwise they'll go to $HOME/jobs
- Main options available:

  --local : runs locally

  -s 'some stuff' : will add the stuff on top of the script. Used to so some setup e.g. -s 'SetupProject root'

  --in [list_of_files] : copies the files to the job folder

  --noClean: normally if you re-submit a job with same name the same folder will be used but cleaned up before. This will avoid the cleanup.

MC production

You want to produce MC on the cluster? We have a framework for it!

- ✓ It provides Sim09b and Sim09c steps for 2011,2012, 2015 and 2016
- ✓ It will produce folders in an orderly fashion for subjobs
- ✓ It gives you monitoring of jobs sent, completed and failed
- ✓ Is sets automatically fair share parameters to avoid that one person will take the whole cluster
- ✓ And last but not least: it's easy to use!!!

https://github.com/marinang/SimulationProduction

MC production: basic usage

- git clone https://github.com/marinang/SimulationProduction.git
- Go into a screen session to avoid problems (Just type “screen”)
- Set the output location to panfs. (/panfs/user/yourfolder)
  You can do this modifying the SIMOUTPUT variable (look in setup.sh)
- The launch your jobs

```
python LaunchProduction.py {Decfile ID} {Nevents} {year} --simcond Sim09c
```

- Enjoy!

Available EventIDs can be found at http://lhcbdoc.web.cern.ch/lhcbdoc/decfiles/

Best practices on the cluster

Long queues and memories

- Point 1: Please do not bypass SLURM! Namely do not do this:

  "`ssh lphe1`" and then laugh a jobs on the node directly

  If you do this you are just making queues longer for others

- Point 2 : set memory and cores responsibly
  - In other words do not do this!

    ```
    #SBATCH -n 2
    #SBATCH —mem-per-cpu=14500
    ```

  - If you reserve all memory the node will be blocked for others
    → long queues for everyone: try to be a responsible user
  - A maximum should be set. Never more then half memory and half cores ?

Excursus: Optimising memory

- Most likely you use more memory than needed
- Memory could be optimised per each job.
- I'm investigating if there is an intelligent way to figure it out
- Cluster experts don't know our codes and can't provide solutions…

- Rule of thumb:
  - Jobs use less memory than you think.
  - Set the memory to 2000 (2GB) and send a test job
  - If SLURM kills it then increase it

Excursus: Optimising memory

- There are memory profilers at least for python very simple to use
  - Wrap you code in a function decorated with @profile

```
@profile
def my_func():
  #Do here all your fancy physics stuff

if __name__ == '__main__':
   my_func()
```

  - Profile! python -m memory_profiler myscript.py

```
Line #    Mem usage    Increment   Line Contents
================================================
     1   28.840 MiB    0.000 MiB   @profile
     2                             def my_func():
     3   36.473 MiB    7.633 MiB       a = [1] * (10 ** 6)
     4  189.062 MiB  152.590 MiB       b = [2] * (2 * 10 ** 7)
     5   36.473 MiB -152.590 MiB       del b
     6   36.473 MiB    0.000 MiB       return a
```

- There's valgrind for C++: valgrind —tool=memcheck myprogram
- Gaudi has valgrind built in:

  gaudirn.py —profilerName=valdrindmemcheck myOptions