LPHE meeting, 1st May 2017, EPFL, Switzerland

LUCA PESCATORE on behalf of angry cluster administrators

CLUSTER AND WEB COMMENTS: living in a community
The problem

- Cluster administrators spotted that some people were bypassing the cluster’s batch system: SLURM

  ```
  ssh lphecl05 ‘my_nasty_script.sh’
  ```

- This should not be done:
  - It disturbs SLURM potentially blocking nodes for others
  - The risk of “loosing” jobs increases

- They threatened to reduce users privileges if we continue
The solution

- Use SLURM for everything! Not just CPU intensive jobs, or jobs that have to run in parallel. Everything!
- In particular: do not launch CPU intensive job on the access nodes 1a and 1b or you’ll block the cluster for everyone.

- There are test nodes: lphe21 and lphe22
- But according to administrators these are also handled by SLURM
  ```bash
  #SBATCH -A debug
  #SBATCH -p debug
  ```
- One issue is Ganga. When open it can block a node.
- Try use it only on test nodes as much as possible.
The survey

• Cluster survey: https://lucapescatore.typeform.com/to/ih9cqd
• Thanks to all who answered already! I got very useful input

• Issues raised:
  • Many people, especially new students do not know how the cluster is organised and how to use slurm. An introduction lecture should be setup.
  • Add possibility to write to eos (investigating)
  • Install gedit (done!)
  • Slurm: long queues and memory
  • Slurm: one user can take over the full cluster for long times
Cluster structure

Access nodes

Work nodes
Queue: batch

Test nodes
Queue: debug

16 cores and 29 GB memory 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.}
Typical SLURM script: my_job.sh

```bash
#!/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@server.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.}
#!/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

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

#SBATCH -o outputs/name.out  # Output logs
#SBATCH -e outputs/name.err   # Job name
#SBATCH -J some_name
#SBATCH -N 1                  # Minimum number of nodes reserved
#SBATCH -n 1                  # Cores reserved per node
#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

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

#SBATCH -o outputs/name.out  # Output logs
#SBATCH -e outputs/name.err  # Job name
#SBATCH -J some_name
#SBATCH -N 1  # Minimum number of nodes reserved
#SBATCH -n 1  # Cores reserved per node
#SBATCH -p batch  # Partition: batch, debug
#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.}
```
#!/usr/bin/env bash

#SBATCH -o outputs/name.out  # Output logs
#SBATCH -e outputs/name.err
#SBATCH -J some_name
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -A batch
#SBATCH -p batch
#SBATCH -t 00:15:00
#SBATCH --mail-user user@server.something
#SBATCH --mail-type ALL
#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  Output logs
#SBATCH -e outputs/name.err  Job name
#SBATCH -J some_name
#SBATCH -N 1  Minimum number of nodes reserved
#SBATCH -n 1  Cores reserved per node
#SBATCH -A batch  Partition: batch, debug
#SBATCH -p batch
#SBATCH —mail-user user@server.something  Summary mail
#SBATCH —mail-type ALL
#SBATCH -t 00:15:00  Time after which job is killed
#SBATCH —mem-per-cpu 1000

{Write here your bash code as you’d normally do.}
#!/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.}
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
- I made a script to help you: `/home/pescator/setup/submit.py`
  
  ```python
  python submit.py -d myjob -n subjob my_command
  ```

It’s actually very similar to “srun” but:

- On the cluster 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

The script you’d normally launch locally

sbatch script
Long queues and memories

- **Point 1**: do not bypass slurm, you are just making queues longer for others

- **Point 2**: set memory and cores responsibly
  - In other words *do not do this*!
    
    ```bash
    #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?
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
Optimising memory

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

```python
@profile
def my_func():
    # Do here all your fancy physics stuff
    if __name__ == '__main__':
        my_func()
```

- Profile!  
  ```
  python -m memory_profiler myscript.py
  ```

- There’s valgrind for C++: `valgrind --tool=memcheck myprogram`
- Gaudi has valgrind built in:
  ```
  gaudirn.py --profilerName=valgrindmemcheck myOptions
  ```
• A framework is now available to make comments to papers and notes in an interactive way.

• Buckle up for the live tutorial and type this link in your browser:

   https://pluca.web.cern.ch/pluca/EasyComment/

• Nobody is forced to use it but if you do it will make things easier.
• Feel free to report bugs or ask for improvements.
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 do 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.