SLURM assignment

Xoel Mato Blanco

Slide 19

Exercise 1

Write a submit script from scratch. The script should use the following parameters:
- Uses 1 node from the research.q queue
- Creates a file (called text.txt with content: "I have written a submit script"
- Sleeps for 30 seconds
- Lists the contents of the folder
Submit your script and check the output

The slurm script was written with the necessary options included in the first lines and the commands to run worked as expected. To run the script, the following command was used:

```
sbatch exercise_1.slurm
```

Script files: exercise_1.slurm
Output files: exercise_1-output, text.txt

Exercise 2

Submit the same job from the command line (i.e. all sbatch options should be added to the command line together with a script file)

The sbatch options were removed from the script and they were added to the command used:

```
sbatch --job-name=exercise_2 --output=exercise_2-output--nodes=1 \
--partition=research.q exercise_2.slurm
```

Script files: exercise_2.slurm
Output files: exercise_2-output, text.txt

Exercise 3

Do the same without using the script file (i.e. adding a –wrap option)

The following command is equivalent to the previous scripts.

```
sbatch --job-name=exercise_3 --output=exercise_3-output --nodes=1 \
--partition=research.q --wrap="echo 'I have written a submit file' > text.txt;sleep 30;ls"
```

Output files: exercise_3-output, text.txt

Slide 28

Exercise 4

We want to sort several text files (names_0.txt. . . names_4.txt). Write a solution that uses SLURM job arrays.
(Hint: use the sort command from Linux to build your solution)

I solved this exercise using a redirection of the echo and sort functions to the file itself (shown below). I also added some options for the slurm array job submission. I made use of the variable *SLURM_ARRAY_TASK_ID* to sort all the files:

```
echo $(sort names_$SLURM_ARRAY_TASK_ID.txt) > names_$SLURM_ARRAY_TASK_ID.txt
```

Input files: names_[0-4].txt
Script files: exercise_4.slurm
Output files: exercise_4-output, names_[0-4].txt

Slide 32

Exercise 5

Modify Job4 and turn it into an array job. When does Job5 start now?

To make the job 4 an array job, I added the following option to the script. Besides, I slightly changed the output name so I could answer the question:

```
#SBATCH --array=1-5
#SBATCH --output=basic-job4-output%a
```

Job 5 only starts after all of the jobs in the array have been completed as I could check by the ending times of job4 outputs and the starting time of the job 5.

Script files: dependencies.slurm, job[1, 2, 3, 4_array, 5].slurm
Output files: basic-job-output, basic-job4-output[1-5]

Exercise 6

Modify individual job scripts so that each job writes its output in a different file.

To do this, I just changed the SBATCH output option in the scripts as follows (eg. job2):

```
#SBATCH --output=basic-job2-output
```

Script files: dependencies.slurm, job[1-5].slurm
Output files: basic-job[1-5]-output

Exercise 7

Write a Python script that does the same as the previous bash script. Which approach (bash or Python script) seems easier for you?

Even though I found python consistently easier to understand, learn and use than bash, bash is more convenient to interact with command line-only tools. Since we are dealing SBATCH and we need to capture its outputs, the python script is full of calls to the *subprocess* module and string manipulations.
Apart from all of that, I run across several problems to properly use the *subprocess* module due to the use different python versions.
For a task like this one, I would rather use a bash script.

Script files: dependencies.py, job[1-5].slurm
Output files: basic-job[1-5]-output