Introduction to the CSF

Practical session 5: Using Job Arrays to run multiple instances of an application.

Overview

We are going to use job arrays to run the programs several times to process different inputs.

There are two parts to this exercise – a simple job array example and a more advanced example.

With the simple job array, you will

1. Modify the job array jobscript to run the `prime_factor.exe` program with a different parameter each time (to find the prime factors of the numbers 1 to 50.)

In the more advanced job array, you will

2. Write a job array jobscript from scratch, using your knowledge of Slurm jobscripts. You'll run a python script to count objects in an image. The job array will run this on several different images. You should be able to determine which image has the most objects.

More information on Slurm Job Arrays on the CSF can be found at https://ri.itservices.manchester.ac.uk/csf3/batch-slurm/job-arrays-slurm/

Instructions – Part 1 – A simple job-array

1. Connect to the CSF using `ssh` if you don't already have a shell on the login node (see practical 1 for how to do this if you can't remember).

2. (This step won't do much if you did it in exercise 1, but won't harm if repeated)
   Install the necessary files by running the following on the CSF login node:

   ```
   module load training/RCSF
   ```

   (enter your University IT `password` – same as used for CSF login - when asked)

3. Ensure you are in the directory that contains the files for today's training

   ```
   cd ~/training/RCSF/examples
   ```

   Linux is case-sensitive so ensure upper and lowercase letters are correct. The ~ character is shorthand for "your home directory".

   Tip: You can press the [Tab] key while typing folder names to see if they'll auto-complete – this can save you a lot of typing!

   We are now going to edit a simple job-array jobscript, then submit it to the queue. It will run an application named *`prime_factor.exe`* to find the prime factors of a number you supply to the program on the command-line.
   We'll calculate the prime factors of the numbers 1 to 50. The prime_factor.exe program only uses a single CPU core (it's a "serial" application.)

   This is a parameter sweep. We are supplying different parameters to the (same) program, and then we'll look at the output from each run to answer a question.

   As always, we describe the job – the resources it needs and the commands we want it to run – in a "jobscript", which is a small text file.

4. Let's examine the text file `pf_jobscript`. You can do this using:

   ```
   gedit pf_jobscript
   ```
   (this will open the text-editor from exercise 1)

   This is the *jobscript* and it tells Slurm (the batch system):
   1. The jobscript is written using the BASH script language.
   2. To run the job on the compute nodes (hardware) dedicated to 1-core "serial" jobs.
   3. To allow a maximum of 2 minutes for the program to complete, once it starts (we know this program doesn't take long to do its computation.)
   4. This is a job array (it has the **`-a`** flag.) **BUT, THE LINE IS INCOMPLETE – SEE BELOW.**
   5. For today, we have some reserved compute nodes, so we use those.

6. The command that we want the job to run. In this case it's a program named `./prime_factor.exe` (the `./` at the start means the program is in the folder where we submit the job from.) **BUT, THE LINE IS INCOMPLETE – SEE BELOW.**

5. **Modify the jobscript - change the "`#SBATCH -a CHANGEME`" line so that the job-array runs 50 copies of the job numbered 1 to 50.**

   Also modify the final line so that the `prime_factor.exe` program is given the current *Slurm array task ID* number. This ensures that:
   Array task 1 will calculate the prime factors of the number 1.
   Array task 2 will calculate the prime factors of the number 2.
   Array task 3 will calculate the prime factors of the number 3.
   Array task 4 will calculate the prime factors of the number 4.
   ...
   Array task 50 will calculate the prime factors of the number 50.
6. Submit the job to the batch system (i.e., submit it to the "queue")

   ```
   sbatch pf_jobscript
   ```

   it will return a unique *JOBID* number to you. **Make a note of it**.

7. Check the status of your array job in the batch queue by running:

   ```
   squeue
   ```

   to see just *your* jobs. If you see nothing, your job has finished!

8. When the job has finished, examine the output files. Remember that the job array will output a file for each task in the array job.

   ```
   ls -ltr
   ```

   ```
   cat slurm-1234567_1.out
   cat slurm-1234567_2.out
   ```

   If you want to display all of the output files with one command, use a `*` to mean "any characters". This is known as a wildcard. For example:

   ```
   cat slurm-1234567_*.out
   ```

   Q1: Which numbers between 1 and 50 have the most unique prime factors?
   IF YOU CANNOT ANSWER THIS QUESTION, PLEASE ASK FOR HELP NOW.

9. Optional extra – can you make a simple change to the jobscript to make it calculate the prime factors of the numbers 1000, 1100, 1200, ... 2000.

Instructions – Part 2 – A more complex job-array

1. Go to the exercise directory – it has its own folder:

   ```
   cd ~/training/RCSF/examples/hudf_images
   ```

2. List the files:

   ```
   ls
   ```

   Q: How many image files are there? You will need this number when you write your jobscript.

   The images are from Hubble Ultra Deep Field https://esahubble.org/images/heic0611b/
   (Credit: NASA, ESA, and S. Beckwith (STScI) and the HUDF Team.)

   Have a look at one of the images using:

   ```
   eog hudf_1.png
   ```

3. Start `gedit` to write a new jobscript

   ```
   gedit myarrayjob.txt
   ```

   Write a serial jobscript to process an image using:

   ```
   module purge
   module load apps/binapps/anaconda3/2021.11
   python process.py filename.png
   ```

   Allow **5 minutes** for each image to be processed.
   Make it an array-job with the correct start and end task range.
   Use the unique task ID number the image *filename* so that each task processes a different image.

4. Submit your job-array jobscript to the batch system.

   ```
   sbatch myarrayjob.txt
   ```

5. Once the job has run, notice that new image files have been written. You can have a look at one of these using:

   ```
   eog hudf_1_out.png
   ```

   Examine all of the **`slurm-JOBID_TASKID.out`** files – the python code will have reported how many objects it detected in each image.
   Q: Which image file contains the most detected objects?