Introduction to the CSF

Practical session 2: Using Slurm to submit a serial job

Overview

We are going to use a simple executable *simple.exe* as an application to run on the CSF:

1. Examine the *simple_jobscript* file which will run *simple.exe* as an Slurm (batch) job
2. Submit the jobscript to run on a compute node using Slurm's **sbatch** command
3. Use Slurm commands **squeue**, **scancel, seff** and **sacct**

More information on Slurm on the CSF can be found at https://ri.itservices.manchester.ac.uk/csf/batch-slurm/

Instructions

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 run an application named *`simple.exe`* as a job on the CSF. This is a program that does a calculation.

   More realistically, the program would do a large simulation or process a large data file that might take several days to complete. It could be a well-known application like `matlab` or `abaqus` or `gaussian`, or even a program you have written yourself (e.g., some python code).

   But the principle here is that we can run whatever program(s) we need for our research by submitting jobs to the batch system. Those jobs will run our programs on the powerful compute nodes in the CSF.

   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 `simple_jobscript`. You can do this using:

   `gedit simple_jobscript` (this will open the text-editor from exercise 1)
   or
   `cat simple_jobscript` (this will just print the file to screen)

   This is the *jobscript* and it tells Slurm (the batch system):

   1. The jobscript is written using the BASH script language. There are several scripting languages available on Linux. BASH is a popular one and is the same as the login node "shell" that you type commands into.

   2. To run the job on the compute nodes (hardware) dedicated to 1-core “serial” jobs. Our simple.exe program will only ever use one CPU core.
   3. To allow a maximum of 5 minutes for the job to complete, once it starts (we know this program doesn’t take long to do its computation.)
   4. For today, we have some reserved compute nodes, so use those.
   5. The command(s) that we want the job to run. In this case it’s a program named `./simple.exe` (the `./` at the start means the program is in the folder where we submit the job from.)

5. Submit the job to the batch system (i.e., submit it to the “queue”)

   ```
   sbatch simple_jobscript
   ```

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

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

   ```
   squeue
   ```

   to see just *your* jobs. The *ST* column (short for STATUS) should be either `PD` (if your job is pending - i.e., waiting) or `R` (when your job starts running). If you see nothing, your job has finished!

7. When the job has finished, examine the output file. Remember that the job has run on a backend “compute node” and so you don’t see output that is normally written to the screen, like when you run commands on the login node.

   Instead, any output that the application would normally print to screen, is captured into a file called **`slurm-`*`JOBID`*`.out`**

   To list your files and then to display the contents of one of the output files (change the number at the end appropriately):

   ```
   ls -ltr
   cat slurm-123456.out
   ```

   Notice that the most-recently written files appear at the bottom of the list when you run `ls -ltr`. This makes it easier to see which files were last updated by your jobs.

   Q1: What is the answer generated by the simple.exe calculation?
   IF YOU CANNOT ANSWER THIS QUESTION, PLEASE ASK FOR HELP NOW.

8. Once the job has finished you can find out how much resource it used (overall runtime, peak memory, number of cores etc) using:

   ```
   seff JOBID
   ```

(replace ***JOBID*** with your job ID generated by the `sbatch` command) and observe the output. This is useful for finding information such as run times for completed jobs.

You can also use the `sacct` command to get a LOT of stats about the job:

```
sacct -j JOBID
```

For example, `MaxRSS` is the peak memory usage. Use `man sacct` (later) to find out more about the various columns.

9. The `slurm-`*`123456`*`.out` filename does not tell you much about which jobscript was run to generate that output. We can ask the batch system to use a different name.

   Edit the jobscript using:

   ```
   gedit simple_jobscript
   ```

   Add the following line to the list of `#SBATCH` lines:

   ```
   #SBATCH -o %x-%j.out
   ```

   `%x` will be replaced by the jobscript name, `%j` will be replaced by the job ID number. Submit the jobscript to the queue:

   ```
   sbatch simple_jobscript
   ```

   When the job has finished, check the contents of this new output file.

   Q2: What output file did the batch system generate when the job ran?
   IF YOU CANNOT ANSWER THIS QUESTION, PLEASE ASK FOR HELP NOW.

10. Submit 2-3 more jobs (use the `simple_jobscript`), find the job ID numbers and enter

    ```
    scancel JOBID
    ```

    to kill these jobs while they are pending or running.

    This is useful if you submit a job and then decide you don't need it (for example because you've given it the wrong input data, or you've examined the output while the job is running and realise it isn't doing what you thought it would.)

    Check that the jobs really have been deleted from your list of pending / running jobs.

    ```
    squeue
    ```

This should show that the jobs have been removed. If you see no output at all from this command, it simply means you have no jobs in the queue!

11. Summary – you have now submitted at least one job to the CSF. You used a batch script (a simple text file) to say what resources the job should use and for how long. The jobscript also contains the actual commands that you wanted the job to run – in this case it ran the simple.exe program. The output from that program was captured into a text file named slurm-JOBID.out. It was then possible to query Slurm for some job statistics. The scancel command can be used to remove jobs from our queue that we no longer want.