Introduction to the CSF

Practical session 2: Using SGE 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 SGE (batch) job
2. Submit the jobscript to run on a compute node using SGE's **qsub** command
3. Use SGE commands **qstat**, **qdel** and **qacct**

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

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 **may** not be necessary if you did it in practical 1)

   Install the necessary files by running the following on the CSF login node:

   ```
   module load training/RCSF
   ```

   (enter central University IT `password` when asked)

3. Ensure you are in the directory to that containing 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".

4. We are going to run an application named *`simple.exe`*. This is just a simple program that counts to ten using a loop and does a dummy calculation (if you want to see how, edit the *`simple.exe`* file but often you can't edit applications in this way).

   Obviously counting to 10 is not a very interesting task. More realistically the application would do a large simulation, or could actually be a well-known application like `matlab`, an application to process genome data or a program you have written yourself etc. But the principle here is that the *`simple.exe`* just does your required task, whether it is something trivial or some long running complicated processing you need for your research.

5. Examine the text file `simple_jobscript`. This is the *jobscript* and it tells SGE (the batch system):

   1. We will be writing the jobscript using the BASH script language. There are several scripting languages available on Linux. BASH is a popular one.
   2. To run the job in the current directory
   3. To set job name to be ***`myjob`***
   4. When the job eventually runs on a compute node, run the *executable* `./simple.exe` there (our *application* in this example).

   Note: by default (unless you say otherwise) jobs on the CSF run on a single core

6. Edit `simple_jobscript` and change ***`myjob`*** to something unique to you.

7. Submit the job to run on a compute node on the CSF by entering

   ```
   qsub simple_jobscript
   ```

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

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

   ```
   qstat
   ```

   to see just *your* jobs. The *state* column should be either `qw` (if your job is waiting) or `r` (when your job starts running). If you see nothing your job has finished.

   If jobs are completed faster than you can find them, edit `simple.exe` and increase simple time (use `gedit`).

   Use the `qstat` command to view the status of the queues / other jobs. For example, enter:

   ```
   qstat -u "*"
   ```

   to see jobs from all users (it will be a long list, showing how busy the system is and how much work is run on the CSF!) To get even more verbose output, run:

   ```
   qstat -f -u "*"
   ```

9. When the job completes examine the output files. Remember that the job has run on a backend compute node and so you can't see output that is normally written to the screen when you run any software. Instead *standard output* is sent to a file called ***`job_name.ojobID`***, and *error messages* are sent to ***`job_name.ejobID`*** (although some software will write all messages to the `.o` file only). For example, to list your files and to read one of the output files (change the number at the end appropriately):

   ```
   ls -ltr
   cat myjob.o123456
   ```

   The names of the files will depend on what you changes ***myjob*** to in step 6. Notice that the most-recently written files appear at the bottom of the list. This makes it easier to see which files were last updated by your jobs.

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

    `qacct -j` ***`jobID`***

    (replace ***`jobID`*** with your job ID generated by the `qsub` command) and observe the output. This is useful for finding information such as run times for completed jobs. Use `man accounting` to find out more about the various outputs.

11. Submit 2-3 jobs, find the job ids and enter

    `qdel` ***`jobID`***

    to kill these jobs while they are pending or running.