An Introduction to HPC — Exercises

Stuart Rankin
sjr20@cam.ac.uk

Research Computing Services (http://www.hpc.cam.ac.uk/)
University Information Services (http://www.uis.cam.ac.uk/)

31st July 2019 / UIS Training

Exercise 1: Login

- Log into your RCS training account.

  Hints: Create a terminal window and use **ssh** to login to your cluster training account.

  The remote host is **login.hpc.cam.ac.uk**. The user name is the same name as your MCS Desktop training account (i.e. **z4XY**).

  N.B. If in doubt about the name of your training account, check the number of your station (see the label on the top of the box), then station 1**XY** should correspond to account z4**XY**.

Exercise 2: Simple command line operations

(a) List your current directory (folder) using **ls -al**. Use **df -h** to see the various cluster filesystems, their sizes and their current total usages. You will be on a random login node – use **hostname** to confirm which one, and **w** to find out who else is using it. Use **lstopo** to find out more about the internal structure of the login node.

(b) Examine your personal filesystem quotas with the command **quota**.

You should see a 40GB quota on /home, a 1TB block and 1024k file quota on /rds-d2 (which corresponds to~/rds/hpc-work).

(c) Ask the scheduler what compute resources are available to you with **mybalance**. This command may take a little while to return (the units are CPU/GPU/KNL hours).

Exercise 3: Laptop setup

Before attempting exercise 3:

- `https://www.hpc.cam.ac.uk/training-courses`
- Download the file **exercises.tgz** to your desktop.
- You will need an sftp client on your laptop to then transfer this file to the cluster.
- Mac users - open a terminal, you can use sftp from the command line
- Windows users - download winsftp

Exercise 3: File transfer

- Use SFTP to transfer the file **exercises.tgz** to your Research Computing Service training account directory ~/rds/hpc-work.

  Hints: The command is **sftp**. Use the same remote host, username and password as in the previous exercise.

  Use **cd rds/hpc-work** to change the target directory, then **put exercises.tgz** to transfer the file from your desktop to the target directory on the Research Computing Service cluster. Use **quit** to close the connection.

  Optionally, copy the file over again using **rsync**.

Exercise 3: File transfer (ctd)

- Switch back to the SSH session you created in the previous exercise. Verify that the file is now present by using **ls**.

  Hints: Do **ls -al ~/rds/hpc-work/**. Note that you can often reduce typing by pressing **TAB**.

- Unpack the tar archive to create an exercise subdirectory.

  Hints: Do **cd ~/rds/hpc-work/** then **tar -zxvf exercises.tgz**.

Exercise 5: Modules and Compilers

- Go to the **exercises** directory of your cluster account.

  Hints: Firstly you may need to review Exercise 1 in order to reconnect to your cluster account. At the remote command prompt, change to the exercises directory (**cd ~/rds/hpc-work/exercises**).

- Try to compile the **hello.c** program using the default **gcc** compiler (it will fail because there is a deliberate bug).

  Hints: **gcc hello.c -o hello**

- To fix the problem, open the **hello.c** file in an editor (e.g. **gedit**, **nano**, **emacs**).

  Hints: Launch gedit in the background by doing **gedit&**. A gedit window should appear. Remove the word **BUG**, save the file and recompile. Do **./hello** to run the program.

Exercise 5: Modules and Compilers (ctd)

- The default version of gcc on the RCS HPC clusters is 4.8.5. Compile hello.c again with **gcc 5.4.0**.

  Hints: module av, module load, then **gcc hello.c -o hello2**

- Launch the Matlab GUI. Note this should work from either the SSH command-line or remote desktop sessions.

  Hints: **module load matlab** then run: **matlab&**

- Quit Matlab and launch it again without the graphical desktop interface. This is the way to launch it inside a batch job.

  Hints: **matlab -nodisplay -nojvm -nosplash**

Exercise 6: Submitting Jobs (Matlab)

- Submit a job which will run **matlab** on the **file.m** command file (which contains just the Matlab **ver** command).

Hints:

1. Load the matlab module at the place indicated in the file **job_script** in your exercises directory.
2. Set the value of application to **¨matlab -nodesktop -nosplash -nojvm¨**
3. Set the value of options to **¨-r file¨**
4. Submit the job with **sbatch job_script**. The jobid is then printed.
5. Watch the job in the queue with **squeue**.
6. After it has disappeared, open the output file **slurm-jobid.out** in your editor. It should contain a list of licensed Matlab features from the ver command.
7. For more demanding work you can increase the available memory by increasing the number of cpus.

Exercise 7: Submitting Jobs (serial or threaded application)

- Submit a job which will run a copy of your hello program on 1 cpu.

  Hints:

  1. Edit the script **job_script** in your exercises directory. Set:
     #SBATCH –nodes=1
     #SBATCH –ntasks=1
     application="./hello"
  2. Submit the job with **sbatch job_script**. The jobid is then printed.
  3. Watch the job in the queue with **squeue**.
  4. After it has disappeared, open the output file **slurm-jobid.out** in your editor. There should be exactly one "Hello, World!" message.

  Experiment with varying the number of nodes and tasks (you are limited to 4 nodes). Note you will need to launch the application with **srun** to actually use more than 1 cpu.

Exercise 8: Submitting Jobs (R)

- R jobs may be serial, threaded, or even MPI parallel depending on the packages used. Submit a job which will run the trivial script **hello.r** program on 1 cpu.

  Hints:

  1. Edit the script **job_script** in your exercises directory. Set:
     #SBATCH –nodes=1
     #SBATCH –ntasks=1
     application="Rscript"
     options="hello.r"
  2. Submit the job with **sbatch job_script**. The jobid is then printed.

- Repeat this using a different version of R.

Exercise 9: Array Jobs

- Submit your last job in the form of an array with indices 1-64. Use -H with sbatch to mark the array as held (so that it won't run immediately).

  Hints:
  1. Use **sbatch -H --array=1-64 job_script**
  2. Use **squeue -u userid** to see your array job. Note that **-r** reports each array element individually.

- Release array element 1 and allow it to run. Then release the others.

  Hints:
  1. Use **scontrol release ${SLURM_ARRAY_JOB_ID}_1**
  2. Use **squeue -u userid** again to watch what happens.
  3. Release the others with
     scontrol release ${SLURM_ARRAY_JOB_ID}
     i.e. use the array id to release the entire array.
  4. When all the jobs complete you should have 64 slurm-${SLURM_ARRAY_JOB_ID}_N.out files saying hello from various cpus on possibly