UNIVERSITY OF CAMBRIDGE

# 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/)

19th October 2017 / UIS Training

## Exercise 1: Login

- Using MobaXterm, login to your HPCS training account.

  *Hints:* Start MobaXterm by double clicking on the file MobaXterm_Personal_7.3.exe in the folder U:\\MobaXterm. Press Session (top left) and SSH in the settings panel which appears.

  The remote host is login-gfx2.hpc.cam.ac.uk. Specify the username — this 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**.

  Enter your password as supplied on the sheet.

  Alternatively Linux or Mac users may prefer to start a MobaXterm command line and use the ssh command as in the course notes instead.

## Exercise 2: File transfer

- Using MobaXterm, SFTP the file exercises.tgz to your HPCS training account.

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

- Unpack the tar archive to create an exercise subdirectory.

## Exercise 2: File transfer

- Using MobaXterm, SFTP the file exercises.tgz to your HPCS training account.

  *Hints:* Start a SFTP session, using the same remote host, username and password as in the previous exercise.

  Drag the exercises.tgz file from the U drive to your home directory on the HPCS (this is the initial directory viewed after the connection is made).

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

- Unpack the tar archive to create an exercise subdirectory.

## Exercise 2: File transfer

- Using MobaXterm, SFTP the file exercises.tgz to your HPCS training account.

  *Hints:* Start a SFTP session, using the same remote host, username and password as in the previous exercise.

  Drag the exercises.tgz file from the U drive to your home directory on the HPCS (this is the initial directory viewed after the connection is made).

- 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 exercise*

- Unpack the tar archive to create an exercise subdirectory.

## Exercise 2: File transfer

- Using MobaXterm, SFTP the file exercises.tgz to your HPCS training account.

  *Hints:* Start a SFTP session, using the same remote host, username and password as in the previous exercise.

  Drag the exercises.tgz file from the U drive to your home directory on the HPCS (this is the initial directory viewed after the connection is made).

- 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 exercise*`

- Unpack the tar archive to create an exercise subdirectory.

  *Hints:* Do `tar -zxvf exercises.tgz`

## Exercise 3: Remote desktop [Optional]

- Using MobaXterm, connect to the remote desktop running on login-gfx2.hpc.cam.ac.uk on display 66. The VNC password is *“trAin99”*.

## Exercise 3: Remote desktop [Optional]

- Using MobaXterm, connect to the remote desktop running on login-gfx2.hpc.cam.ac.uk on display 66. The VNC password is *"trAin99"*.

*Hints:* Start a VNC session. Because the HPCS only allows SSH connections, to use VNC we need to tunnel via SSH.

Use localhost as the remote hostname, and set the Port to $5900 + 66 = 5966$.

Now go to Advanced VNC settings, tick Connect through SSH gateway and enter login-gfx2.hpc.cam.ac.uk as the gateway server, with your training account ID as the user. Click OK.

You should be prompted first for your training account password, then for the VNC password which is *"trAin99"*. Note that this is a view-only password.

## Exercise 4: Modules and Compilers

- Go to the exercises directory of your HPCS training account.

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

- To fix the problem, open the hello.c file in the gedit editor.

## Exercise 4: Modules and Compilers

- Go to the exercises directory of your HPCS training account.

  *Hints:* Firstly you may need to review Exercise 1 in order to reconnect to your HPCS training account. (Note that your earlier SSH session may in fact be saved on the left side.) At the HPCS command prompt, change to the exercises directory (cd ~/exercises).
- Try to compile the hello.c program using the default icc compiler (it will fail because there is a deliberate bug).
- To fix the problem, open the hello.c file in the gedit editor.

## Exercise 4: Modules and Compilers

- Go to the exercises directory of your HPCS training account.

  *Hints:* Firstly you may need to review Exercise 1 in order to reconnect to your HPCS training account. (Note that your earlier SSH session may in fact be saved on the left side.) At the HPCS command prompt, change to the exercises directory (`cd ~/exercises`).

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

  *Hints:* `icc hello.c -o hello`

- To fix the problem, open the hello.c file in the gedit editor.

## Exercise 4: Modules and Compilers

- Go to the exercises directory of your HPCS training account.

  *Hints:* Firstly you may need to review Exercise 1 in order to reconnect to your HPCS training account. (Note that your earlier SSH session may in fact be saved on the left side.) At the HPCS command prompt, change to the exercises directory (cd ~/exercises).

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

  *Hints:* icc hello.c -o hello

- To fix the problem, open the hello.c file in the gedit editor.

  *Hints:* Launch gedit in the background by doing gedit&. A gedit window should appear. Remove the word BUG and save the file.

## Exercise 4: Modules and Compilers (ctd)

- Try again to compile the hello.c program using the default icc compiler, and run it. You should see "*node* says: Hello, World!".

- Which version of icc did you use? Find this out by listing the current modules loaded.

- Compile and run the hello.c program in the exercises directory using a non-default C compiler.

## Exercise 4: Modules and Compilers (ctd)

- Try again to compile the hello.c program using the default icc compiler, and run it. You should see "*node* says: Hello, World!".

  *Hints:* `icc hello.c -o hello` then run: `./hello`.
- Which version of icc did you use? Find this out by listing the current modules loaded.
- Compile and run the hello.c program in the exercises directory using a non-default C compiler.

## Exercise 4: Modules and Compilers (ctd)

- Try again to compile the hello.c program using the default icc compiler, and run it. You should see "*node* says: Hello, World!".

  *Hints:* icc hello.c -o hello then run: ./hello.

- Which version of icc did you use? Find this out by listing the current modules loaded.

  *Hints:* module list — the Intel compiler modules are named intel/cce/*version*. You can also work out the version directly by finding the location of the binary, e.g. doing which icc which should return the path: /usr/local/Cluster-Apps/intel/cce/*version*/bin/icc.

- Compile and run the hello.c program in the exercises directory using a non-default C compiler.

## Exercise 4: Modules and Compilers (ctd)

- Try again to compile the hello.c program using the default icc compiler, and run it. You should see "*node* says: Hello, World!".

  *Hints:* icc hello.c -o hello then run: ./hello.

- Which version of icc did you use? Find this out by listing the current modules loaded.

  *Hints:* module list — the Intel compiler modules are named intel/cce/*version*. You can also work out the version directly by finding the location of the binary, e.g. doing which icc which should return the path: /usr/local/Cluster-Apps/intel/cce/*version*/bin/icc.

- Compile and run the hello.c program in the exercises directory using a non-default C compiler.

  *Hints:* E.g. load the latest PGI C compiler (pgcc) with module load pgi. module av will show all possible choices (not all of which are compilers).

## Exercise 5: Submitting Jobs

- Submit a job which will run a copy of your hello program on all cores of 4 of the 24-core nodes which are available for training.

## Exercise 5: Submitting Jobs

- Submit a job which will run a copy of your hello program on all cores of 4 of the 24-core nodes which are available for training.

  *Hints:*
  1. Edit the script job_script in your exercises directory. Set:
     #SBATCH –nodes=4
     #SBATCH –ntasks=96
     application="./hello"
     In the module section, make sure that the module you used to compile hello is also loaded (last).
  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 24 "Hello, World!" messages from 4 different nodes.

## Exercise 6: Submitting Jobs (ctd)

- Experiment with changing the number of nodes and tasks by changing and submitting job_script (you are limited to 8 nodes in total).