UNIVERSITY OF CAMBRIDGE

An Introduction to High Performance Computing: Exercises

Paul Sumption
support@hpc.cam.ac.uk

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

22nd May 2018 / CSD3 User Training

Exercise 1: Login

Using a Linux terminal you will login to the cluster with your HPC training account.

- Start the terminal by double clicking on the terminal icon
- In your terminal enter:
- ssh -Y **abc123**@login-cpu.hpc.cam.ac.uk
  Replace abc123 with your training account username
- Enter your password as supplied on the sheet
- Leave this terminal open, you will need it for exercise 3!

Exercise 2: Transfer some files

You will need to transfer the exercise files to the cluster.

- Open a second Linux terminal on your training computer.
- Enter this command: cd `~\Course_material`
- Check the file 'exercises.tar is in your directory listing
- Hint: ls

Exercise 3: Transfer some files

Transfer the exercises.tar to your HPC home folder.

- In the local terminal on your training computer enter the command:
- sftp abc123@login.hpc.cam.ac.uk
  Change abc123 to your training account username
- The command: put exercises.tar will transfer the file from your local computer to the remote one
- Type 'exit' to close the local terminal

Exercise 3: Learn more about a command

(a) View the man page for the cp command by doing man cp. Use SPACE to page down and b to page up. Press q to exit the manual page command.

(b) Copy exercises.tar to the ~/hpc-work directory. Note that ~ is just a convenient shorthand for your home directory. Omitting the ~/ will look for a hpc-work in the current directory.

Hints: Do cp exercises.tar ~/hpc-work/. Note that you can often reduce typing by pressing TAB.

Exercise 4: Unzip the excercises.tar file

(a) Use the cd command to enter the ~/hpc-work directory and then list the contents — you should see the copy of exercises.tgz.

Hints: Do cd ~/hpc-work/ then ls -al. Note that cd .. will take you back up one step to the home directory.

(b) Unpack the tar archive to create an exercise subdirectory.

Hints: Do tar -zxvf exercises.tar

Exercise 5: File listings

(a) In a terminal logged into the cluster list the contents of your current directory ls. This won't show everything — use ls -al for a long listing showing all files. Initially you will start in your home directory — use pwd to print the name of your current working directory. If you get lost, you can always do cd without arguments to return to your home directory.

(b) Focus your long listing on all files with names beginning "exercise".

Hints: Do ls -al exercise*

(c) Print a long listing of the subdirectory hpc-work.

Hints: Do ls -al hpc-work/. Note that omitting the / reveals that the item hpc-work is actually a shortcut (technically a symbolic link) to /hpc-work/username.

Excercise 6: Environment Modules

- Connect to the cluster using your training account: See excercise 1 if you have closed your terminal.
- Get a list of modules that are currently loaded

Hints: module list

- Get a list of available R modules

Hints: module av R

Excercise 7: Run an Rscript

- Connect to the cluster using your training account: See excercise 1 if you have closed your terminal.
- In the exercises folder you transferred earlier there is a file called test.r
- Run this script using: Rscript hello.r
- Load the module for: R/3.3.2

Hints: module load R/3.3.2

- Run the script again: Rscript hello.r
- What happens? what changes?

Exercise 8: Install the R library locally

As a user you can create a local R library directory for packages that you want to install.

- Load an R module: module load r-3.4.3-gcc-5.4.0-rbvhnga
- Create a folder in your home for your own R package installs: mkdir /my-R-libs
- Make R aware of the new library location:

  `echo "R_LIBS_USER=~/my-R-libs" \textgreater .Renviron`
- Start R: R
- Display your library paths: .libPaths()
- Try loading a library: require(pander)
- Its not insalled, lets install it: install.packages("pander")
- Try loading a library: require(pander)
- Library is now installed, lets quit R: quit()

Excercise 8: Explained

- Our R modules: module load r/(version). We have two sets of R modules, those with an upper case R where compiled for an older version of Linux and should be ignored (Darwin legacy)
- echo " " outputs the text between the quotes, >redirects the text into the .Renviron file.
- When we start R the .Renviron file is read and R will now be aware of our local library directory.
- .libPaths() is how to check your library locations

Exercise 9: Modules and Compilers

- Connect to the cluster using your training account: See excercise 1 if you have closed your terminal.
- Go to the exercises directory that you unzipped in hpcwork.
  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 the gedit editor.

  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.
- If you get this error:

  `WARNING **: cannot open display:`

  then you have missed the '-Y' in your SSH command

Exercise 9: Modules and Compilers

- The default version of gcc is 4.8.5. Compile hello.c again with gcc 5.4.0.

  Hints: module av, module load gcc-5.4.0-gcc-4.8.5-fis24gg, then gcc hello.c -o hello2

Exercise 10: Submitting a Matlab job

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

  Hints:
  1. Load the matlab module using the 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.
  7. For more demanding work you can increase the available memory by increasing the number of cpus.

Exercise 11: Submitting compiled code

- 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.

Exercise 11: Array Jobs

- Submit your last job in the form of an array with indices 1-32. Use -H with sbatch to mark the array as held (so that it won't run immediately).

  Hints:
  1. Use sbatch -H --array=1-32 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 multiple nodes.