UNIVERSITY OF CAMBRIDGE

An Introduction to High Performance Computing on the Minerva Cluster — 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/)

Exercise 1: Login

- Using MobaXterm, login to your Minerva account.

  Hints: Start MobaXterm. Press Session (top left) and SSH in the settings panel which appears.

  The remote host is minerva-login1.npl.co.uk. Specify the username — this will be your AD identifier and of the form npl\abXY.

  The password will be your usual password in the NPL Active Directory (AD).

Exercise 2: Simple command line operations (i)

(a) List your current directory (folder) using 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".

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

Exercise 2: Simple command line operations (i)

(a) List your current directory (folder) using 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.

Exercise 2: Simple command line operations (i)

(a) List your current directory (folder) using 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.

Exercise 2: Simple command line operations (ii)

(d) 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.

(d) Copy exercises.tgz 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.

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

- Unpack the tar archive to create an exercise subdirectory.

Exercise 2: Simple command line operations (ii)

(d) 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.

(d) Copy exercises.tgz 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.tgz ~/hpc-work/. Note that you can often reduce typing by pressing TAB.

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

- Unpack the tar archive to create an exercise subdirectory.

Exercise 2: Simple command line operations (ii)

(d) 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.

(d) Copy exercises.tgz 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.tgz ˜/hpc-work/. Note that you can often reduce typing by pressing TAB.

(e) 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.

- Unpack the tar archive to create an exercise subdirectory.

Exercise 2: Simple command line operations (ii)

(d) 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.

(d) Copy exercises.tgz 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.tgz ˜/hpc-work/. Note that you can often reduce typing by pressing TAB.

(e) 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.

- Unpack the tar archive to create an exercise subdirectory.

Hints: Do tar -zxvf exercises.tgz

UNIVERSITY OF CAMBRIDGE

Exercise 3: File transfer

- Using MobaXterm, SFTP the file exercises.tgz from Minerva back to your local filespace.

Exercise 3: File transfer

- Using MobaXterm, SFTP the file exercises.tgz from Minerva back to your local filespace.

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

  Drag the exercises.tgz file from the remote Minerva folder (this will be your home directory on Minerva) to the local PC.

Exercise 4: Remote desktop

- Connect to Minerva and launch a remote desktop. You will need to set a (different!) password the first time. Note your unique display number.
- Using MobaXterm, connect to the remote desktop running on minerva-login1.npl.co.uk on the correct display number.

Exercise 4: Remote desktop

- Connect to Minerva and launch a remote desktop. You will need to set a (different!) password the first time. Note your unique display number.
- Using MobaXterm, connect to the remote desktop running on minerva-login1.npl.co.uk on the correct display number.

  Hints: Because the cluster only allows SSH connections from outside, to use VNC we need to tunnel via SSH.

  Use localhost as the remote hostname, and set the Port to $5900 + \textit{displaynumber}$ (the reason for this is ancient history).

  Now go to Advanced VNC settings, tick Connect through SSH gateway and enter minerva-login1.npl.co.uk as the gateway server, with your AD identifier npl\abc12) as the user. Click OK.

  You should be prompted first for your AD password, then for the VNC password.

Exercise 5: Modules and Compilers

- Go to the exercises directory of your Minerva account.

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

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

Exercise 5: Modules and Compilers

- Go to the exercises directory of your Minerva account.

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

Exercise 5: Modules and Compilers

- Go to the exercises directory of your Minerva account.

  Hints: Firstly you may need to review Exercise 1 in order to reconnect to your Minerva account. (Note that your earlier SSH session may in fact be saved on the left side of the MobaXterm GUI.) Alternatively, use your VNC desktop session. At the Minerva command prompt, change to the exercises directory (`cd ~/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 the gedit editor.

UNIVERSITY OF CAMBRIDGE

Exercise 5: Modules and Compilers

- Go to the exercises directory of your Minerva account.

  Hints: Firstly you may need to review Exercise 1 in order to reconnect to your Minerva account. (Note that your earlier SSH session may in fact be saved on the left side of the MobaXterm GUI.) Alternatively, use your VNC desktop session. At the Minerva command prompt, change to the exercises directory (`cd ~/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 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.

Exercise 5: Modules and Compilers (ctd)

- The default version of gcc is 4.8.5. Compile hello.c again with gcc 5.4.0.
- Launch the Matlab GUI. Note this should work from either the SSH command-line or remote desktop sessions.
- Quit Matlab and launch it again without the graphical desktop interface. This is the way to launch it inside a batch job.
- Launch the COMSOL GUI.

Exercise 5: Modules and Compilers (ctd)

- The default version of gcc 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.

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

- Launch the COMSOL GUI.

Exercise 5: Modules and Compilers (ctd)

- The default version of gcc 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.

- Launch the COMSOL GUI.

Exercise 5: Modules and Compilers (ctd)

- The default version of gcc 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`

- Launch the COMSOL GUI.

Exercise 5: Modules and Compilers (ctd)

- The default version of gcc 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`
- Launch the COMSOL GUI.

  Hints: Search for and load the module, then run `comsol`.

Exercise 6: Submitting Jobs

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

Exercise 6: Submitting Jobs

- 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 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.
  7. For more demanding work you can increase the available memory by increasing the number of cpus.

Exercise 6: Submitting Jobs (ctd)

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

Exercise 6: Submitting Jobs (ctd)

- 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.

UNIVERSITY OF CAMBRIDGE

Exercise 7: 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).

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

Exercise 7: 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.

Exercise 7: 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 multiple nodes.

UNIVERSITY OF CAMBRIDGE