|epcc|

ARCHER2 Performance Optimisation on EPYC Practical Notes

Getting started

Logging on to ARCHER2

You should already have an account on ARCHER2, and should be able to log on to it using

```
ssh -X accountname@login.archer2.ac.uk
```

(replacing accountname with the name of your account on the system) or with the SSH client of your choice (`-X` ensures that graphics are routed back to your desktop). Once you have successfully logged in you will be presented with an interactive command prompt.

For more detailed instructions on connecting to ARCHER2, or on how to run commands, please see the Appendix.

Download and extract the exercise files

Firstly, change directory to make sure you are on the `/work` filesystem on ARCHER2.

```
cd /work/ta039/ta039/accountname/
```

Where ta039 is the name of the project being used for this course. `/work` is a high performance parallel file system that can be accessed by both the frontend and compute nodes. **All jobs on ARCHER2 should be run from the `/work` filesystem.** ARCHER2 compute nodes cannot access the `/home` filesystem at all: any jobs attempting to use `/home` will fail with an error.

Use the following commands (on ARCHER2) to get the exercise files archive from the web and unpack it:

```
cp /work/z19/shared/SNO.tar .
tar xvf SNO.tar
```

Exercise 1: Placement

The code for this exercise is in `Archer2Opt/Placement`. The aim of this exercise is to investigate the performance variation across the cores on an ARCHER2 node. Currently the code is setup to run the STREAMs benchmark with 16 cores, all on a single chiplet on the ARCHER2 system. The batch script will run `xthi`, a program that will print out the placement and binding information for the current configuration, and then the STREAMs benchmark. You can build the applications using the command:

```
make
```

You can submit the application using the command:

```
sbatch --reservation=XXX run_streams.sh
```

XXX should be replaced by the reservation name given to you at the course.

Run the batch script as is, then edit it to run across NUMA regions rather than withing a single NUMA region. You can do this by changing srun -c 1 to srun -c 8. Rerun the application, what is the performance difference?

Now alter the batch script to run 32 processes rather than 16 processes. You will also need to alter the -c 8 flag to -c 4 to change the process binding. What performance difference do you see? Try for other ranges of process counts up to 128.

Exercise 2: CrayPat

The code for this exercise is in `Archer2Opt/VH1`. Please follow the separate instructions in `CrayPAT-intro.pdf`.

Exercise 3: Compiler Optimisation

The code for this exercise is in `Archer2Opt/CompilerOpt`. Please follow the separate instruction in `compilerexercise.pdf`.

Exercise 4: Optimisation

The `Archer2Opt/MD/*` directory, where `*` is either `C` or `Fortran`, contains a sequential implementation of a molecular dynamics simulation which has been deliberately written to have poor performance. Use profiling and compiler listings to look for the performance problems. Once you have exhausted the compiler's ability to optimise, try some code modifications.

Exercise 5: Bandwidth and NUMA

The example code can be found in `Archer2Opt/Stream/*`. This is the well-known STREAM benchmark for measuring memory bandwidth. Use the Makefile to compile the code, and run it using different numbers of threads using the supplied batch script.

Does the bandwidth scale linearly with processors? Now try removing the OpenMP loop directive from the initialisation of the arrays. How does the performance change? You can also try using the "wrong" schedule for the loop, or selecting different sets of cores to run on.

Extra exercise

Try reducing the array size `N` by a factor of 100 or 1000 (and increase the repetition count `NTIMES` by the same amount).

Exercise 6: Cache blocking

The `Archer2Opt/Matmul/*` directory contains a simple matrix multiplication code. Try implementing cache blocking by hand, and see the effect on the performance (stick to `-O0` to stop the compiler doing its own optimisation!).

Exercise 7: OpenMP

The `Archer2Opt/MolDyn/*` directory contains a (not very efficient) OpenMP parallel version of a simple molecular dynamics code (not the same as in Exercise 2!). Run the code using the script supplied with the script supplied to measure the performance on 1, 2, 4, 8 and 16 threads.

Try to identify the performance bottlenecks and fix them! You can try using `pat_build -g omp` to instrument the OpenMP runtime library.

Appendix

Detailed Login Instructions

Procedure for Mac and Linux users

Open a command line *Terminal* and enter the following command:

```
local$ ssh -X username@login.archer2.ac.uk
Password:
```

you should be prompted to enter your password.

Procedure for Windows users

Windows does not generally have SSH installed by default so some extra work is required. You need to download and install a SSH client application - PuTTY is a good choice:

`http://www.chiark.greenend.org.uk/~sgtatham/putty/`

When you start PuTTY you should be able to enter the ARCHER2 login address:

`login.archer2.ac.uk`

When you connect you will be prompted for your user ID and password.

Running commands

You can list the directories and files available by using the *ls* (LiSt) command:

```
username@archer2:~> ls
bin  work
```

You can modify the behaviour of commands by adding options. Options are usually letters or words preceded by '-' or '–'. For example, to see more details of the files and directories available you can add the '-l' (l for long) option to *ls*:

```
username@archer2:~> ls -l
total 8
drwxr-sr-x 2 user z01 4096 Nov 13 14:47 bin
drwxr-sr-x 2 user z01 4096 Nov 13 14:47 work
```

If you want a description of a particular command and the options available you can access this using the *man* (MANual) command. For example, to show more information on *ls*:

```
username@archer2:~> man ls
Man: find all matching manual pages
 * ls (1)
   ls (1p)
Man: What manual page do you want?
Man:
```

In the manual, use the spacebar to move down a page, 'u' to move up, and 'q' to quit and exit back to the command line.

Using the Emacs text editor

As you do not have access to a windowing environment when using ARCHER2, Emacs will be used in *in-terminal* mode. In this mode you can edit the file as usual but you must use keyboard shortcuts

to run operations such as "save file" (remember, there are no menus that can be accessed using a mouse).

Start Emacs with the *emacs* command and the name of the file you wish to create. For example:

```
username@archer2:~> emacs sharpen_batch.pbs
```

The terminal will change to show that you are now inside the Emacs text editor.

Typing will insert text as you would expect and backspace will delete text. You use special key sequences (involving the Ctrl and Alt buttons) to save files, exit Emacs and so on.

Files can be saved using the sequence "Ctrl-x Ctrl-s" (usually abbreviated in Emacs documentation to "C-x C-s"). You should see the following briefly appear in the line at the bottom of the window (the minibuffer in Emacs-speak):

```
Wrote ./sharpen_batch.pbs
```

To exit Emacs and return to the command line use the sequence "C-x C-c". If you have changes in the file that have not yet been saved Emacs will prompt you (in the minibuffer) to ask if you want to save the changes or not.

Although you could edit files on your local machine using whichever windowed text editor you prefer it is useful to know enough to use an in-terminal editor as there will be times where you want to perform a quick edit that does not justify the hassle of editing and re-uploading.

Useful commands for examining files

There are a couple of commands that are useful for displaying the contents of plain text files on the command line that you can use to examine the contents of a file without having to open in in Emacs (if you want to edit a file then you will need to use Emacs). The commands are *cat* and *less*. *cat* simply prints the contents of the file to the terminal window and returns to the command line. For example:

```
username@archer2:~> cat sharpen_batch.pbs
aprun -n 4 ./sharpen
```

This is fine for small files where the text fits in a single terminal window. For longer files you can use the *less* command. *less* gives you the ability to scroll up and down in the specified file. For example:

```
username@archer2:~> less sharpen.c
```

Once in *less* you can use the spacebar to scroll down and 'u' to scroll up. When you have finished examining the file you can use 'q' to exit *less* and return to the command line.

Hardware

Each node of ARCHER2 consist of two sockets, each containing a 64-core AMD Epyc Rome processor.

Compiling

The default compilers are the Cray compilers for Fortran 90 and C. To use the AMD or GNU compilers:

```
username@archer2:~> module restore PrgEnv-aocc
```

or

```
username@archer2:~> module restore PrgEnv-gnu
```

The compiler is always invoked with `ftn` or `cc`, but you will need to modify the flags for the different compilers.

Job Submission

To run codes, you should submit a batch job as follows:

```
sbatch --reservation <resnum> scriptfile.sh
```

where `resnum` is the reservation name for the session.

You can monitor your jobs status with the `squeue` command, and jobs can be deleted with `scancel`.