ARCHER2 for Data Scientists

Practical 1: Getting to know ARCHER2

Adrian Jackson

1 Introduction

In this exercise we are going to log on to the system and run some simple applications to test placement of processes and memory bandwidth on ARCHER2.

2 Using the system

For this course we are using ARCHER2. You should already have a personal ARCHER2 account associated with this course, probably within the project that has been setup for this course (`ta105`). If you already have an ARCHER2 account in a different project you can use that as well.

To access ARCHER2 you should do the following (replacing `username` with your actual account username on ARCHER2):

```
ssh username@login.archer2.ac.uk
```

From here you can compile and submit jobs. We use the modules environment for controlling software such as compilers and libraries. You can see what software has been installed on the system using the following command

```
module avail
```

The system is configured with a login node separate from the compute nodes in the system. When we initially log on the Cray compilers are loaded. You can choose other compilers like this:

```
module load PrgEnv-gnu
```

We use the Slurm batch system to access and enquire about the compute nodes. You can discover how many compute nodes there are, and what memory they have installed, using the following command:

```
sinfo
```

Below is an example of a Slurm batch script we can use to run a job:

```
#!/bin/bash
#SBATCH --job-name=xthi
#SBATCH --nodes=2
#SBATCH --tasks-per-node=128
#SBATCH --partition=standard
#SBATCH --qos=short
#SBATCH --account=ta105
#SBATCH --time=0:10:0
```

```
module load xthi
srun -n 256 xthi
```

To run a job on the system we use the `sbatch` command, i.e. (assuming the script above is called `runtestjob.sh`)

```
sbatch runtestjob.sh
```

You can `squeue` to see running jobs (`squeue -u $USER` will show only your jobs) and `scancel` to cancel a job.

The `srun` command in the script above is the parallel job launcher which runs the executable on the selected number of nodes.

On ARCHER2 we have multiple filesystems. The first filesystem you encounter when you log in is the /home filesystem. For anything running on the compute nodes we need to use the /work Lustre filesystem. Filesystems are organised as follows

```
/home/project/project/$USER
/work/project/project/$USER
```

For instance, if you are in the ta105 project and your username is adrianj, you will have the following directories available to you:

```
/home/ta105/ta105/adrianj
/work/ta105/ta105/adrianj
```

As only /work is available on the compute nodes, for this course we recommend always using your /work directories.

You can download the exercise code and handouts we will use for this course by doing the following:

```
cd /work/project/project/$USER
git clone https://github.com/EPCCed/archer2-data-science
```

3 Process placement

In the first exercise we will just use an existing program on ARCHER2, xthi, which lets us investigate process placement and thread binding on the system. In the `exercises/binding` directory of the repository for this course there is a batch script called run_xthi.sh. It contains the following:

```
#!/bin/bash
#SBATCH --job-name=xthi
#SBATCH --nodes=1
#SBATCH --tasks-per-node=128
#SBATCH --cpus-per-task=1
#SBATCH --partition=standard
#SBATCH --qos=short
#SBATCH --time=0:10:0
module load xthi
```

```
export OMP_NUM_THREADS=1
srun xthi
```

As specified this will run a 128 process MPI program, each with 1 OpenMP thread. You should get output that states which process is running on which core, and how many threads they have.

Run the program using:

```
sbatch run_xthi.sh
```

Your task for this exercise is try and vary the number of MPI tasks and OpenMP threads, whilst still ensuring there is a single thread per core (i.e. no workers are overlapping on compute cores). Try and run the following configurations:

Processes	Threads
128	1
64	1
64	2
32	4

You can change the number of OpenMP threads used by `xthi` by editing the batch script and change the `export OMP_NUM_THREADS=…` line.

4 Memory performance

In the `exercises/binding` directory there is a file called streams.tar.gz. You can unpack this using the following command:

```
tar xf streams.tar.gz
```

Then you can go into the streams directory and type:

```
make
```

Then you can run it using:

```
sbatch run_streams.sh
```

Streams is a memory bandwidth benchmark, so we can use it to explore what memory bandwidth we can achieve when running different process/thread configurations on a compute node. Run the following configurations and check what memory bandwidth streams achieves:

Processes	Threads
128	1
64	1
64	2
32	4
16	8
8	16
4	32

To change the number of processes you'll need to alter the batch script to change the `cpus-per-task` and `tasks-per-node`. To change the number of threads you'll need to alter the batch script to change the `for THREADS in 1` line to use different numbers.

In which configuration do you see the best process and node memory bandwidth numbers? Does removing `--distribution=block:block` from the `srun` line make any difference to the bandwidth achieved for any of the configurations?