Persistent Memory Exercises

Adrian Jackson

1 Introduction

In this exercise we are going to log on to the system and run a simple program to ensure you can access the system and you know what the system setup is like. We provide a pre-compiled application for you to run.

2 Using the system

For this tutorial we will be using the NEXTGenIO prototype system. You should have an account on this system already. To access the system you need to ssh to a gateway node (replacing `XX` in the `ngguestXX` string below with the number of the actual account you have been given):

```
ssh ngguestXX@hydra-vpn.epcc.ed.ac.uk
```

Once on this gateway node you need to ssh in to the NEXTGenIO system itself (if you want to open any windows, i.e. emacs, you'll need to add a –Y or –X flag to the command below):

```
ssh nextgenio-login2
```

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. 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
```

Or:

```
sinfo -N -l
```

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

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=01:00:00
#SBATCH --job-name=test_job
#SBATCH --nvram-option=1LM:1000
#SBATCH --cpus-per-task=1

mpirun -n 48 -ppn 48 test_job
```

The line `#SBATCH --nvram-option=1LM:1000` specifies the setup of the B-APM required by the job. For this system this option is required otherwise your job will not run. The first part of the text after `--nvram-option=` specifies the platform mode required for the job, `1LM` specifies AppDirect mode and `2LM` specifies MemoryMode. The number after the platform mode (i.e. `:1000`) specifies how much B-APM memory is required for the specified platform mode (in this case the script is specifying AppDirect with 1TB of B-APM memory).

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 `mpirun` command in the script above is the MPI job launcher which runs the executable on the selected number of nodes. The default MPI library being used is the Intel MPI library (although you can use the OpenMPI library as well by swapping out the requisite modules). For the Intel MPI library there are a number of arguments we are passing, i.e.:

```
mpirun -n 48 -ppn 48 test_job
```

Here `-n 48` specifies the number of processes (or copies of you executable) to run. `–ppn 48` specifies how many processes to run on each node requested. The exercises we are doing in this practical will have submission scripts with `mpirun` already specified in them

We are using the Intel compiler setup on this machine. This means the C compiler is called `icc` or `mpicc` and the Fortran compiler is called `ifort` or `mpif90`.

On the NEXTGenIO prototype system we have a Lustre filesystem where your home directory is installed, at:

```
/home/nx04/nx04/$USER
```

On the compute nodes there are two further filesystems:

```
/mnt/pmem_fsdax0
/mnt/pmem_fsdax1
```

These correspond to the B-APM installed with the first processor in the node (pmem_fsdax0) and the second processor in the node (pmem_fsdax1). Accessing the B-APM attached to the processor your application is running on is faster than accessing the B-APM attached to the other processor in the node.

3 Running a basic program

To get started on the system copy the following software into your home directory:

```
/home/nx01/shared/pmtutorial/exercises/streams.tar.gz
/home/nx01/shared/pmtutorial/exercises/IOR.tar.gz
```

You should be able unpack both with the commands:

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

`tar xf IOR.tar.gz`.

For IOR you need to go into the IOR directory and type:

```
make mpiio
```

Then you can run a Lustre IOR benchmark using:

```
sbatch lustre_ior.sh
```

And you can run a fsdax IOR benchmark using:

```
sbatch fsdax_ior.sh
```

For the streams benchmark you need to go into the streams directory and type:

```
make
```

Then you can run it using:

```
sbatch run_streams.sh
```

The purpose of this practical is to run both IOR and STREAMs and ensure you can get jobs running and get output from jobs. Once you have applications running the task is to compare the IOR benchmark results from the Lustre and fsdax filesystem, which is comparing Lustre to the on-node B-APM hardware so you can see the raw difference in filesystem performance. You can also compare the filesystem performance to the memory performance in the system by comparing to the STREAMs benchmark output. Using these three values should give you a rough estimate of the overall performance of the B-APM hardware in this system compared to the standard DRAM that is installed.

Lustre Performance: _______________

Fsdax Performance: ________________

STREAMs Performance: ___________