Crib Sheet: Bridges MPI & OpenMP Exercises

David Henty

1 Logging on

Use your username and password to access Bridges:

```
[user@@latop ~]$ ssh -Y myusername@bridges.psc.edu
```

Although Bridges has both scratch and persistent high performance filesystems (`/pylon1/` and `/pylon2/`), which should be used for large production jobs, here we will just use the home filesystem for simplicity.

2 Obtaining source code

You can obtain the example code packages `IHPCSS-pi.tar` and `IHPCSS-traffic.tar` from the Moodle pages. However, if you download these from a browser then they will end up on your laptop which isn't particularly useful.

It is probably easier if you copy them directly from my account on Bridges, e.g.:

```
[user@bridges ~]$ cp /home/dsh/ihpcss17/IHPCSS-pi.tar .
```

Unpack using tar:

```
[user@bridges ~]$ tar -xvf IHPCSS-pi.tar
```

which will create a number of directories containing code.

3 Compiling code

You will see that there are MPI versions provided in both C and Fortran, and in some cases OpenMP and serial versions as well.

Compile using the supplied Makefiles, e.g. for the pi MPI example in C:

```
[user@bridges ~] cd piexample/C-MPI
[user@bridges ~] make
[user@bridges ~] mpicc  -c piparallel.c
[user@bridges ~] mpicc  -o piparallel piparallel.o -lm
```

The GNU compilers (`gcc` and `gfortran`) are used for the MPI programs (via the standard wrappers `mpicc` and `mpif90`), whereas for OpenMP we use the Intel compilers (`icc` and `ifort`).

4 Running on the Bridges compute nodes

To make things simple, I have provided template batch job scripts for SLURM (`mpibatch.job` and `ompbatch.job`) that you can use to run your own programs.

For example, to run a parallel MPI executable called `piparallel` you just have to make a copy of the template with a matching name and then submit to SLURM:

```
[user@bridges ~] cp mpibatch.job piparallel.job
[user@bridges ~] sbatch piparallel.job
```

You can monitor how your jobs are progressing using `squeue -u myusername`. When the job has finished, the output will appear in a file called `slurm-XXXXXX.out` where `XXXXXX` is the job number assigned at submission.

By default, the job scripts run your program on 4 processes (MPI) or 4 threads (OpenMP). This is easy to alter – see the instructions in the batch jobs themselves.

Things to note:

- each of the standard nodes on Bridges has 28 cores;
- this means that the maximum number of threads for OpenMP programs is 28;
- to run MPI programs on more than 28 processes, you must request more than one node via the `-N` option in the MPI template.

5 Running on the login nodes

Although you must run jobs on the compute nodes to get meaningful performance results, it can be useful to run locally on the login nodes during development and debugging.

To run an MPI program on, for example, 4 processes:

```
[user@bridges ~] mpirun -n 4 ./mympicode
```

For OpenMP on 4 threads:

```
[user@bridges ~] export OMP_NUM_THREADS=4
[user@bridges ~] ./myopenmpcode
```

For the exercises during the Summer School we have access to dedicated resources via special reservations. The MPI jobs are set up automatically to use Monday's reservation `ihpcssm`, and the OpenMP ones to use Tuesday's `ihpcsst`.

If you need to change these just edit this line in the batch job:

```
#SBATCH --res=ihpcssm
```

6 Interactive access to the compute nodes

It is possible to gain interactive access to the Bridges compute nodes using SLURM. For example, to reserve 2 nodes (i.e. 56 cores) for your own exclusive use:

```
[user@bridges ~] interact -n 56
```

You can then run interactively as described in Section 5. The difference is that you have exclusive access to the nodes you have reserved, as opposed to the login nodes which are shared between users, so performance results should be reliable and reproducible.