|epcc|

Crib Sheet: Bridges2 MPI & OpenMP Exercises

David Henty

1 Logging on

Use your username and password to access Bridges2:

```
[user@latop ~]$ ssh -p 2222 xsede-username@bridges2.psc.edu
```

Although Bridges2 has high performance filesystems which should be used for large production jobs, here we will just use the home filesystem for simplicity.

2 Obtaining source code

All the course material, including source code, is available on github:

```
[user@bridges2 ~]$ git clone https://github.com/davidhenty/ihpcss2022.git
```

3 Compiling code

For these initial tests, `cd` into `hello` and then the appropriate subdirectory for your chosen language / parallel model.

Whenever you log in you have to load a module to access MPI:

```
[user@bridges2 ~]$ module load openmpi
```

The GNU compilers (`gcc`, `g++` and `gfortran`) are used for compiling MPI programs via the standard wrappers `mpicc`, `mpicxx` and `mpif90`.

To compile an MPI program in C:

```
[user@bridges2 ~]$ mpicc -o hello hello.c
```

or in Fortran:

```
[user@bridges2 ~]$ mpif90 -o hello hello.f90
```

As an interpreted language, there is no explicit compilation stage for Python programs. However, you need to do some additional setup:

```
[user@bridges2 ~]$ module load anaconda3
(base) [user@bridges2 ~]$ pip install mpi4py
(base) [user@bridges2 ~]$ conda deactivate
```

Setting up `mpi4py`, the Python wrappers for the standard MPI library functions, may take some time when you first install it but should be much faster afterwards.

For OpenMP, you pass a special option to the GNU compilers, e.g. for C:

```
[user@bridges2 ~]$ gcc -fopenmp -o hello hello.c
```

For Fortran:

```
[user@bridges2 ~]$ gfortran -fopenmp -o hello hello.f90
```

Due to the way that the Python interpreter is implemented, threaded programming is problematic and there is no Python equivalent of OpenMP.

4 Running on Bridges2

4.1 MPI

You cannot run MPI jobs on the login nodes. The simplest approach for development work on small numbers of processes is to use the `interact` command to access the compute nodes.

For example, if you want to run on up to 8 processes:

```
[user@bridges2 ~]$ interact -n 8
```

After a small wait you should be allocated resources and see a different prompt. You can then run C or Fortran executables:

```
[user@r001 ~] mpirun -n 8 hello
```

or for Python:

```
[user@r001 ~] mpirun -n 8 python3 hello.py
```

4.2 OpenMP

You can run parallel OpenMP jobs on the login nodes, but you should only do this for small test jobs. For example, to run on 8 threads:

```
[user@bridges2 ~]$ export OMP_NUM_THREADS=8
[user@bridges2 ~]$ ./hello
```

For compute intensive jobs, use the `interact` command to get access to dedicated resources on the compute nodes as described in the MPI section above.

4.2.1 Python

As mentioned above, there is no equivalent of OpenMP for Python.

5 Running jobs during the Summer School

For the actual summer school we will have reservations in place to give you guaranteed access to resources. Instructions on how to access the reservations will be given at the relevant hands-on sessions.