MPI: Practical session 1

Note: These practicals are based on the "Further Work" section at the end of Dr Rutter's slides. Extra (or at least different) detail may be found there (https://www.mjr19.org.uk/courses/MPI/MPI.pdf).

Fortran users: Programs `hello_world.f08` and `get_version.f08` are provided on the course's Moodle page. To compile, replace `mpicxx` with `mpifort`.

1 My First MPI Program

Open your favourite text editor and type in the following:

```
#include <iostream>
#include <mpi.h>
int main(int argc, char *argv[])
{
        MPI_Init(&argc, &argv);

        int rank, size;
        MPI_Comm_size(MPI_COMM_WORLD, &size);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);

        std::cout << "Hello parallel world, I am process "\
        << rank << " out of " << size << "\n";

        MPI_Finalize();
        return 0;
}
```

Save the file as `hello_world.cpp` and then, at the command-line, type the following:

```
$ mpicxx hello_world.cpp -o hello_world
$ ./hello_world
```

Now type the following:

```
$ mpiexec ./hello_world
```

In order to set the number of processors, use the `-n` option, e.g.:

```
$ mpiexec -n 8 ./hello_world
```

Try different numbers of processors, up to and even beyond the number of physical cores on your chosen computer.

2 Using different MPI implementations

There are multiple implementations of the MPI standard. One or more may be installed on whatever computer you are using and they must be used consistently. For example, `apollo` has two implementations installed. The default version is OpenMPI, but MPICH is also available:

```
$ mpicxx.mpich hello_World.cpp -o hello_world
$ ./hello_world
```

Now type the following:

```
$ mpiexec.mpich ./hello_world
```

In order to run on different numbers of processors, use the `-n` option, e.g.:

```
$ mpiexec.mpich -n 8 ./hello_world
```

Do you notice any differences between MPICH and OpenMPI? The differences you can see are purely cosmetic. There are more subtle differences in how the two implementations are written, and hence their respective performance for particular cases. There may also be (usually) non-obvious bugs that exist in one but not the other.

3 Mixing different MPI implementations

Since MPI implementations are apparently interchangable, try the following:

```
$ mpicxx.openmpi hello_world.cpp -o hello_world
$ mpiexec.mpich -n 8 ./hello_world
```

What does this tell you? Try the same thing with `openmpi` and `mpich` reversed. When you come to use computers with many more implementations of MPI installed, you will need to be very careful that your compilation commands and execution commands are consistent. Even different versions of OpenMPI may not have compatible `mpiexec`s.

4 MPI Wrappers

As you will have realised, MPI is essentially a header file and some pre-compiled library functions (rather than a compiler feature). If you type:

```
$ mpicxx -show hello_world.cpp -o hello_world
```

then you will see that `mpicxx` actually just calls a compiler with particular options which point it to the header and library files.

5 MPI Version

Check the version of MPI provided by all implementations of MPI that exist on your computer, using:

```
#include <iostream>
#include <mpi.h>
int main(int argc, char *argv[])
{
        int major, minor;

        MPI_Get_version(&major, &minor);
        std::cout << "Run-time MPI version is " << major << "." << minor << std::endl;
        return 0;
}
```

6 Parametric studies

OpenMPI sets environment variables when it runs, which can be used outside MPI programs. For example, try:

```
$ mpiexec -n 4 /bin/bash -c 'echo I am rank $OMPI_COMM_WORLD_RANK of $OMPI_COMM_WORLD_SIZE'
```

Note the single-quotes, and the fact that we have to spawn a new shell.
Consider how this might be used to run an embarrassingly parallel parametric study, where the individual simulations are serial. The environment variables are implementation specific; in order to find those for MPICH, try:

```
$ mpiexec.mpich -n 1 env
```