University of Wolverhampton
School of Mathematics and Computer Science

5CS022 Distribute and Cloud Systems Programming Week 1 Workshop

Overview

The aim of this workshop is to familiarise you with building, compiling and running MPI programs. You can carry out this workshop either the university servers: thinlinc.wlv.ac.uk or your own Linux system (if you prefer to use Putty to log in instead of the Thinlinc client, connect to the server tl-01.wlv.ac.uk instead).

Tasks

1. Download the sample MPI programs from Canvas into your Linux system. Compile and run the program mpi01.c. To compile it, run the following command in the terminal:

   ```
   mpicc mpi01.c -o mpi01
   ```

   Now run it with the following:

   ```
   mpiexec ./mpi01
   ```

   This will (probably) only run only one process, which is not very interesting. Run it again with the following command::

   ```
   mpiexec -n 4 -oversubscribe ./mpi01
   ```

   Note the output this time. It should indicate that 4 processes have run and they all have different process IDs.

   Experiment with higher and higher numbers of processes until it stops running. Then have a look at the error message and try and work out why it stop working.

2. Compile and run the program mpi02.c. Try running it with 2, 3 and 4 processes. Eg.:

   ```
   mpiexec -n 2 -oversubscribe ./mpi02
   mpiexec -n 3 -oversubscribe ./mpi02
   mpiexec -n 4 -oversubscribe ./mpi02
   ```

   Note what happens. It doesn't let you run the program with anything other than 3 processes.

3. Now change the code so that you remove the check for only 3 processes. Now run it with 2, then 3 , then 4 and then more processes.

4. When you try to run it with 4 or more processes, it probably runs and appear to work, but never ends. You will have to end with "Ctrl-C". Why do you think it doesn't end when you run it with more than 3 processes? Change it so that it will work with any number of processes.

5. Build and run the program mpi03.c. In this program Process 0 will wait for messages from Process 1 and Process 2. However, Process 1 end sup blocking Process 2 because it sleeps for 5 seconds. How would you change the code so that Process 1 does not block Process 2, even if it does sleep for 5 seconds?

6. The following is a simple program that looks for prime numbers between 1 to 10000:

```
#include <stdio.h>

int main(int argc, char **argv)
{
  int i, c;
  int nstart=1, nfinish=10000;
  printf("%s : Prime numbers between %d and %d are :\n",
         nstart, nfinish);
  for(i=nstart; i<=nfinish; i++)
  {
    for(c=2; c<=i-1; c++)
    {
      if ( i%c==0 )
        break;
    }
    if ( c==i )
      printf("%s : %d\n",argv[0], i);
  }
  return 0;
}
```

Convert it to MPI so that it can run with different numbers of processes including just one process.