Computação Paralela

November-2020

Introduction to MPI

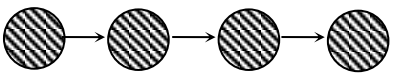
These exercises aim to introduce the basic concepts of MPI programming.

Consider the following MPI program:

```
#include <mpi.h>
#include <stdio.h>
int main( int argc, char *argv[]) {
    int rank, msg;
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    /* Process 0 sends and Process 1 receives */
    if (rank == 0) {
        msg = 123456;
        MPI_Send( &msg, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
    }
    else if (rank == 1) {
        MPI_Recv( &msg, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &status );
        printf( "Received %d\n", msg);
    }
    MPI_Finalize();
    return 0;
}
```

Change the program to implement a pipeline of processes (using the SPMD model):

a) 4 processes each process receiving 1 message that is successively processed by each process in the pipeline



b) Same as a) but the number of processes is provided in the `mpirun -np xx` command line (note: use the `MPI_Comm_size` call to get the number of processes spawned by the `mpirun` command). Test the program with several `np`

c) Same as b) but each process should receive and/or send 10 messages

Notes:

1) reserve 4 nodes on the search with: `qsub … -nodes=4 ...`
2) load the mpi environement: `module load gnu/openmpi_eth/1.8.4`
3) compile and run with: `mpicc prog.c`
4) run with `mpirun -np 2 -mca btl self,sm,tcp a.out`

Other notes:

- on more recent MPI implementations it might also require the `-oversubscribe` flag on `mpirun`
- on Ubunto 18.2 - install mpich