MPI Assignment Report

High Performance Computing Course – Riccardo Caprile (43707074)

INFN Cluster Specifics

- **CPU :** Intel(R) Xeon Phi(TM) CPU 7210 @ 1.30 GHz
- **Architecture :** x86_64
- **Cores :** 64
- **Threads per core :** 4 (256 threads total)

How to run the program using MPI

- mpiicc pi_homerwork_parallel .c -o pi_homework_parallel
- mpiexec -hostfile machinefile.txt -perhost 32 -np 256 ./pi_homework_parallel , otherwise if some host are not available : mpiexec -np 256 ./pi_homework_parallel.

-np specify the number of processes we want to run.

Code Analysis

At the beginning of the program , there is the definition of a constant value of intervals (100.000.000.000) , this number determines the quantity of intervals used to compute the approximation of the pi.

In the serial version of the program , the loop (which is the hotspot) is executed sequentially by a single process , and clearly this will slow down the execution time because of the high number of intervals. So we have to speed it up using MPI , by splitting the work among multiple processes. In this way each process computes the sum of a subset of the intervals , and at the end of the loop all the subsets are summed together.

For parallelizing the program with MPI , we have to use some MPI functions. The first is MPI_Init(&argc , &argv) which initializes the message passing routines. Then we need to know how many processes are associated with a communicator and to know the IDs of each process using MPI_Comm_size(MPI_COMM_WORLD,&rank) and MPI_Comm_rank(MPI_COMM_WORLD,&rank). In this way all the next lines of code are executed in parallel.

Next we have to compute the start and the end indices of the for loop based on the rank and size of the process working because we want that each process computes a partial part of the pi. After the foor loop , we have to sum together every partial part we need to use the instruction MPI_Reduce(&aux_pi,&pi,1,MPI_DOUBLE,MPI_SUM,root,MPI_COMM_WORLD), in this way all the partial parts contained in aux_pi are summed together and the final value saved in pi. The instruction MPI_Barrier(MPI_COMM_WORLD) is used at the beginning and at the end of the parallel section for starting all the processes together and waiting for each other at the end. The last instruction to be used is MPI_Finalize() which finalizes the MPI environment

Conclusions

The goal of this assignment was to improve the performances of the program provided using MPI environment and the goal has been clearly achieved.

The serial version of the program took 1319.64 seconds for completing the task. Too much time. The parallel version of the program with MPI took 12.46 seconds , it is a considerable improvement. The real value of PI is 3.141592653589793238462643 , while the computed value of PI is 3.141592653589780237410878. The two values are very similar , this tell us that the parallelization was correct.