Instituto Superior Técnico

Electrical and Computer Engineering

Parallel and Distributed Computing

2nd Semester 2018/2019

Particle Simulation

Serial + MPI

84037 – Eduardo Costa
84038 – Eduardo Melo
84087 – João Sebastião

Group: 25
Shift: Wednesday, 12:30
Professor: Nuno Roma

May 17th 2019

1 Program Structure and Parallelization Strategy

The program structure is presented in figure 1.

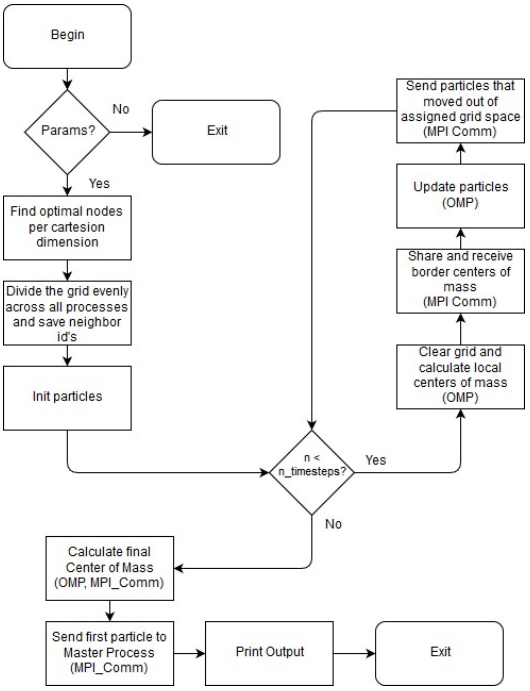

Figure 1: Fluxogram of the program structure.

In order to get the most performance out of the processes used, the parallelization strategy used was the division of the grid evenly between all the processors. To implement this, we made use of the function MPI_DIMS_CREATE to get the optimal number of processes per cartesian dimension (in our case 2 dimensions). Since we are dealing with large sets of probabilistic data, we can expect that the distribution of the particles on the grid will be homogeneous. Thus, we can guarantee that, for large sets of data, the initial number of particles for each process will be, on average, the same and so the processing load is well balanced.

The sections of the program that require MPI communication between nodes are identified by MPI Comm in the fluxogram above. Some of the program sections, mainly the sections that compute values and require no communication, have been parallelized using OpenMP, identified by OMP. An effort was also made to minimize the number of sends, in order to decrease the initial overhead of the communication.

2 Performance Results

Regarding the performance results, the first target of analysis was the influence of the increase in the number of processes in the speedup. Several tests were performed, with different grid sizes, time steps and number of particles, and the results obtained are represented in figure 2.

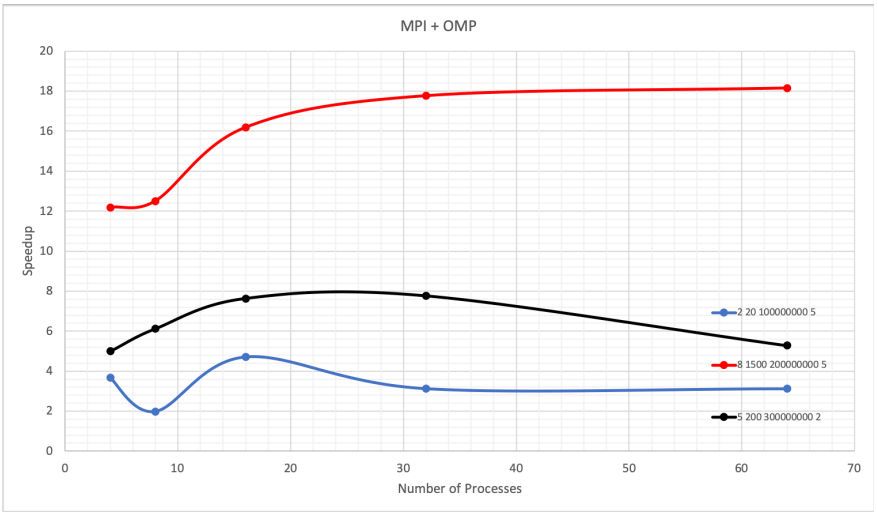

Figure 2: Speedup obtained in 3 different tests with different number of processes.

In figure 2, the graph shows that in the 3 different tests, the speedup increases until reaching its maximum, when it will start decreasing, due to the Amdahl's effect. Also, the speedup grows with the size of the grid and the number of particles, because the serial implementation grows terribly with an increase in any of them, while the MPI implementation keeps handling it well.

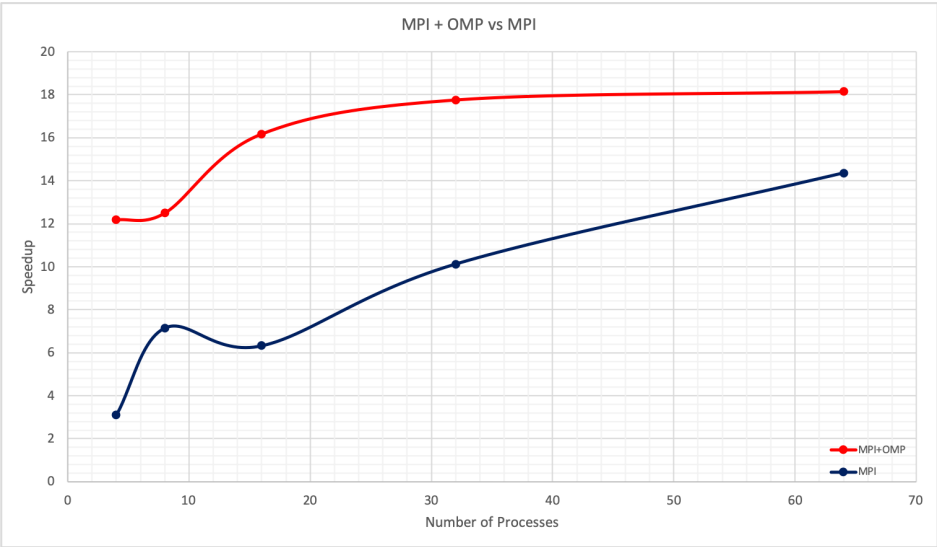

Figure 3: Speedup comparison between MPI + OMP and MPI with different number of processes.

Since our implementation of the program uses a combination of MPI and OMP, it is important to analyse how great the impact of OMP is. For this reason, we conducted the same test in an MPI+OMP run as well as using just MPI. The results obtained are represented in figure 3.

It is noticeable that the MPI + OMP implementation reaches its peak before the MPI implementation, and has a higher speedup for all the number of processes tested.

After that, the third target of analysis was the influence of the increase in the number of particles in the speedup. We analised 2 tests, with different grid sizes and time steps, and the results obtained are represented in figure 4.

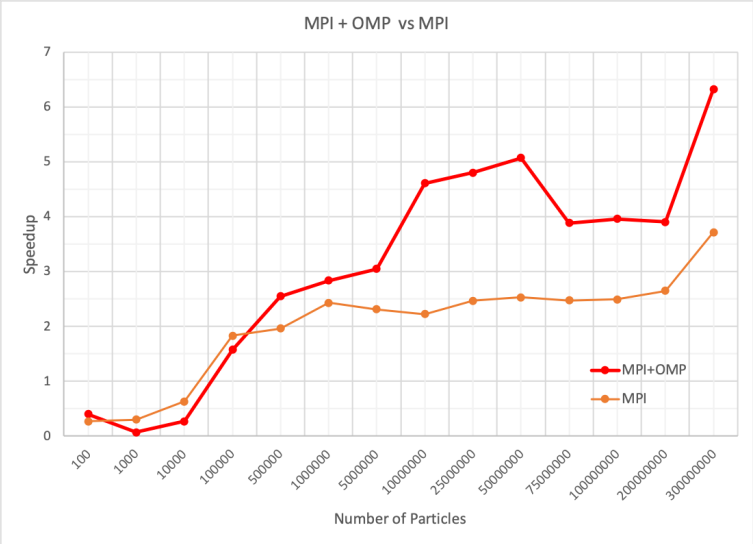

Figure 4: Speedup comparison between MPI + OMP and MPI with different number of particles.

Overall, The MPI + OMP implementation has a higher speedup than the MPI implementation, as expected. The MPI + OMP implementation has a noticeable drop in performance when reaching the 75M particles. This may happen due to the test conditions, namely this being ran in the cluster, where some of the CPU's used may also being used locally.

As demonstrated in figures 3 and 4, the MPI + OMP implementation performs better than the MPI implementation standalone in both cases. This is expected, since the OMP implementation will increase the performance on the computations that do not require communication between different processes.