Instituto Superior Técnico

Mestrado Integrado Em Engenharia Eletrotécnica e de Computadores

CPD

CPD MPI Report

Trabalho realizado por:	*Número:*
Duarte Correia	81225
José Nobre	84107
Rafael Forte	84174

Grupo: 10

**

17/05/2019

1 Parallelization

In this project, we worked with OMP and MPI in order to maximize the efficency of our program. To accomplish this we've created a MPI version for our program and then joined the OMP code that was used in the last delivery, So we have several machines in parallel running a parallel program, which makes our program much more efficient.

2 Decomposition

Unlike openMP, MPI works with distributed memory systems. Working with these systems means the program will be running on different processes (CPUs) and each will have its own local memory. In order for a process to access data outside his local memory it has to communicate via a interconnection network, this communication takes a long time so it is something that should be kept to a minimum if possible.

In order to minimize the number of communications between processes in this project, the particles were distributed along the processes. So each process would get approximately n/p particles, where **n** is the number of particles and **p** the number of processors. If after this division there are some particles left (the rest of the operation), they are distributed one by one by the first processes, for example, if there are ten processes and the remainder of the division is three, the first three processes will have one more particle than the rest.

3 Approach and Load Balancing

With this approach each processor have their portion of the particles set, and its own version of the cells. The only communication needed will be a reduction of the cells of the different processors. Since the number of cells is usually much smaller than the number of particles, this approach is better than the alternative option of dividing the cells between the processors and sending to them only the particles corresponding to their cells (this could also lead to a big unbalance of work between processors, since there is no way to guarantee that the cells have the same amount of particles).

This way all the processors will have the same amount of work to do.

4 Concerns

There are two major concerns in the approach taken. The first is making sure the particles created in the different processors would be the equal to the particles if they were created all in one process. In order to guarantee this, we use the same seed in every processor, and execute the number of rand() calls necessary, so that the rand() call that produces the the particles gives the same value.

The other concern is making sure that before starting computing the new velocity and position, all processes have the updated cells (after the reduction), otherwise the processor only have the values of the cells they calculated based on their particles.

5 Performance Results

In order to test the program performance 3 types of tests were made (Figure 1):

1. Changing the number of nodes used and the number of cores per node
2. Changing the number of cells on the side of the grid
3. Changing the number of particles to be simulated

For test 1. the execution times of the program demonstrate an improving tendency, as expected. On the hand it should be noted that this was not achieved for every test, the reason shall be placed on how the cluster manages resources asked since the tests that show the more disparities were made with two cores per process and the divisions made by the cluster depend on the resources that are available. Since communications between two processes on the same machine can be faster when contrasted to inter machine communications, the resource division can create this differences. In order to mitigate these variations results of 10 tests with the same parameters were averaged.

With test 2. it's possible to show that the execution time grows exponentially with the number of cells per side of the grid, this was also expected since the number of cells grow with $\mathcal{O}(N^2)$ time complexity and for a large most of the time is used for communications, it is noteworthy that the time did not improve with more nodes, since the time gained in the particles calculations is lost with more communications (this is true for larger number of cells).

At last for test 3. it was verified that the execution time is really small (and probably mostly communications) and increasing linearly with the number of particles, the biggest test is for 40GB worth of particles information and since each node only has 2GB available it's near the limits of RAM when using 40 nodes (the particles are scattered uniformly over every process), there is still great efficiency when using more nodes since doubling from 40 to 80 decreases the processing time to half.

All in all the program behaves great with very high number of particles since it splits the computational power needed equally among all nodes but it doesn't deal well with increasing number of cells since those are "shared" between all processesand the reducing operations can take a lot of time for bigger grids.

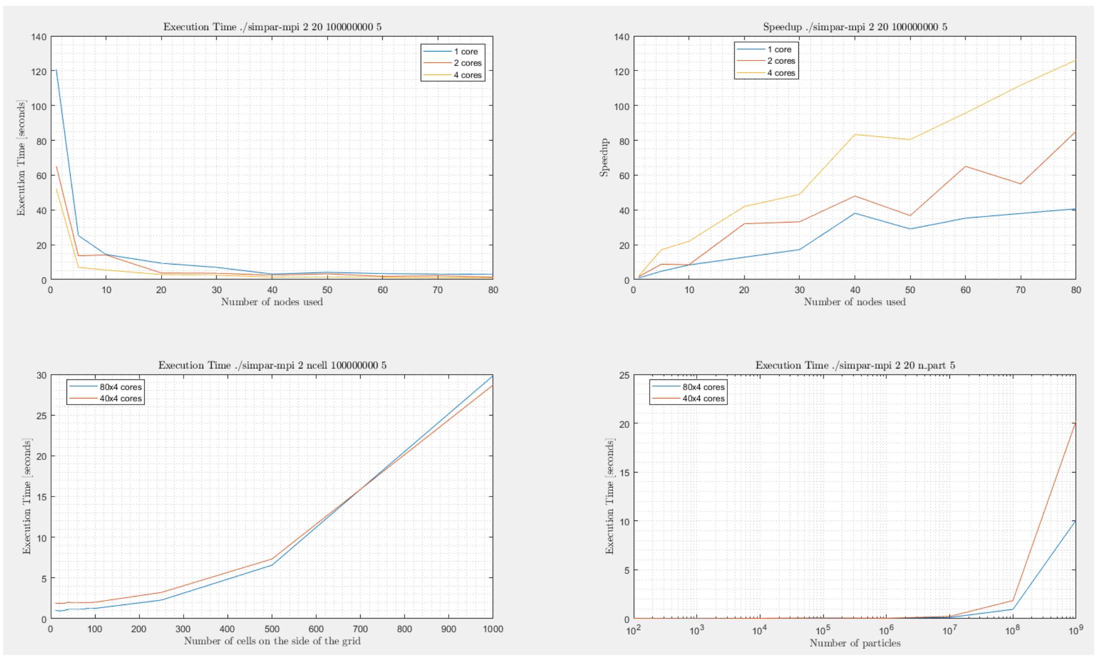

Figure 1: *Performance of the program for multiple nodes and different parameters*