Parallel and Distributed Computing Report

Diogo Paulo Dias 93980
Gonçalo Pires Teixeira 94064
Tiago Bartolomeu Luiz 93976

May 2019

1 Introduction

For this stage, we have implemented three versions in order to test which one grants a better scalability and performance. First we adapted the Open MP version to use MPI directives, that is, each process is responsible for a piece of the matrix and after the center of mass of each cell is computed, we perform a reduce operation. The other two versions use a completely different approach, where instead of iterating the particles, we iterate the cells, having each cell the array of particles it owns. Both versions use a row-wise approach, however one assumes the existence of two ghost-rows and that all the particles of the process' adjacent rows are communicated, whether the other version assumes the existence of only one ghost-row and that the adjacent processes are responsible to communicate the particles that will transferred. In this report we shall focus on the row-wise versions, since the one that uses reduce was already explained on the previous report, being the only novelty the fact that we perform an *AllReduce* instead of a common reduce, since every process needs the complete matrix to compute the particles' movements.

On both row-wise versions, each process needs to have at least the number of ghost-rows, so if we use two ghost-rows, for a matrix whose side is 5, we can have, a maximum of 2 processes, if we have a side of 10, we can have a maximum of 5.

2 Implementation Details

The algorithm can be divided into three different phases, (1) the computation of the center mass of each cell, (2) the computation of the movement of the cells and (3) the computation of the global center mass.

In the first phase, after computing the center of mass of the cells assigned to each process (primitive task), then each process communicates with the ones that are adjacent to it (process two with the number one and three, and there on). In the version with two ghost-rows, we communicate two rows of each adjacent process, and also the particles that are in the adjacent row, for example, process 1 has the rows 1 to 3, process 2 rows 4 to 6 and process 3 7 to 9, then process 2 will receive the cells from rows 2, 3, 7 and 8, however it will only receive the particles that are in rows 3 and 7. In the version with only one ghost rows, the methodology is similar, however we only transmit a row and do not need to transmit any particles in this phase.

In the second phase, the version with two ghost-rows starts computing the movements at the second top ghost-row (the one with the particles), so that a process can see which particles will be moved from the ghost-row (which represents an adjacent process) to a cell that belongs to it. The need for two ghost-rows is due to the fact that, to compute the movement of the particles on the ghost-row, we need to have the neighbours (from whom we only need the position and mass of the center mass) that are above (on the top ghost-row) and below (on the bottom ghost-row). For its turn, the other row-wise version, only computes the movements of the particles that belongs to it, after that, the particles that were moved to a ghost-row, are communicated to the process which that ghost-row represents. This is the major difference of both versions, since one communicates all the particles independently if they move out of the process or not, which is more inefficient, however, for this delivery, due to time constrains we choose to deliver the more inefficient version since it is more stable even with a high number of timestamps.

Finally, in the third phase, all versions do the same thing, compute the global center mass of their particles, and then reduce the result which is then completed and printed by process zero.

As said above, the version that was delivered is the row-wise that uses two ghost-rows. In order to make up for the fact that is a more inefficient version, we also use Open MP where it is possible.

3 Isoefficiency Analysis

On Table 1 we only analyze two versions, the one that uses the reduce and the one that uses two ghost-rows (row wise), the version with one ghost-row was not evaluated since it will be very similar to the one with two and because it was the only one that didn't have consistent results. On this analysis we refer to the particles as Pa, to all cells as Ce, to the cells of a given row as RoCe and to X as the particles that are on a row that will be sent to a given process.

The reduce version is a little above a linear function (borderline), therefore, not scalable. Regarding the row-wise version, given that we depend of variables that depend on the initialization of the particles and much more, is not easy to do a 100% correct analysis, however, assuming that (1) 2X is much smaller than Pa, (2) 4RoCe is much smaller than Ce, than this version is more scalable than the other, however still not ideal. A more scalable version would use checkerboard partitioning, whose impact would be felt at the communication level (and consequently at the scalability function).

	Reduce	**Row-Wise Cells**
Sequential Complexity	2 (Pa + Ce)	2 (Pa + Ce)
Parallel Complexity	(2 (Pa + Ce))/p	(2 (Pa + Ce))/p
Communication Complexity	Ce * Log(p)	4RoCe * 2X * Log(p)
Parallel Overhead	Ce * p * Log(p)	4RoCe * 2X * p *Log(p)
Isoeficiency Relation	C(Ce * p * Log(p))	C(4RoCe * 2X * p * Log(p))
Scalability Function	C(Ce * Log(p))	C(4RoCe * 2X * Log(p))

Table 1: Isoefficiency Analysis of the two approaches used