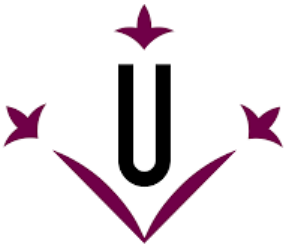

Universitat de Lleida

Mandatory Activity: Implementation with OpenMP of the HPC project

Made by
Joaquim Pico

Delivery
30th of April, 2022

Universitat de Lleida

Escola Politècnica Superior

Màster en Enginyeria Informàtica

Computacio d'altes prestacions

Professorate:

Francesc Gine

Contents

List of Figures

List of Tables

1 Code Paralelized

Firstable, as it was accorded through mail, the part of coding of this work has been done together with the pair of students composed by Oriol Alas Cercos and Sergi Simon Balcells. Never the less, we worked independently on the analysis of the results obtained. The code is pretty explanatory on which thing can be parallelized. As always, the first look is on the loops. Moreover, one thing that we should avoid when paralyzing is when reading/writing from a file. With this two liine in our head, we have a lot more reduced the possible targets in wich we can apply a loop. The last one discarded is the loop regarding the initialization of the grid matrix (reservation of space with all 0). Therefore the remaining zones with loops that don't involve in any of the previously mentioned tasks will be able to be parallelized. Those are the loop zones that are located in the initialization of the grid, the update of the grid and the loops regarding setting the border lines of the board.

2 Strategy used

As we are working with matrices, we can go in all cases with a shared grid between the different workers. The reason is because they will distribute the task in a way that they will never collide at any position. The only variables that are required to be private for all the different threads are going to be the ones that impact on the access of the matrices. That also will be the ones that we will keep incrementing inside the loops, making them more secure to race conditions if we do them private.

3 Other Tested Strategies

We were discussing this with Sergi and Oriol. We tested static dynamic and guided and the one that performed better was the guided scheduler. This may be because it does the balancing between the threads better that the dynamic strategy would probably do. As guided scheduler, changes the chunk size while de execution is runs and it adjusts this size if the workload is balanced.

4 Analysis

For the analysis, a change was done to the serial implementation due that was a think impacting on the analysis. There was a log that was doing I/O which is a costly operation too much times, worsening the quality of the data for the analysis. So the log it was cleaned with a boolean constant that can be set to true when it is wanted to do some debug.

Once said this, lets take a look at the executions that I did and the results that I ended with. First lets discuss about the executions that I did. I have executed the program with 1-2-3-4 cores for the examples of size 5000x5000 and 10000x10000. As those examples are the bigger ones, are the ones that are going to let me see the real scalability and speed up of the solution. The smaller once, where finishing faster which leads to more difficulty to insights on them was more difficult.

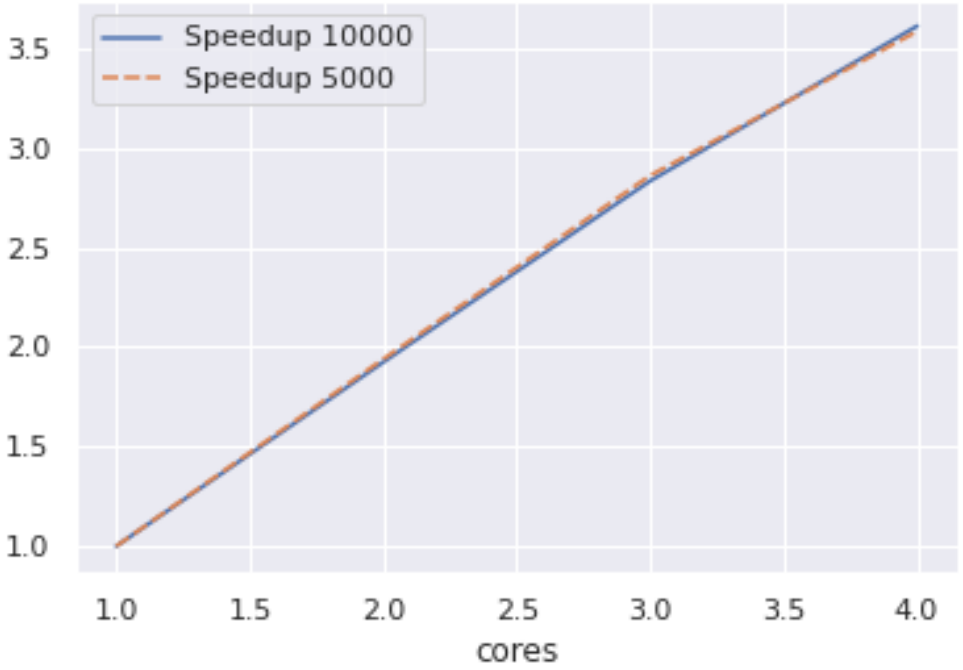

Figure 1: Speedup for 5000x5000 and 10000x10000

Let's start talking about speedup, as we can see on the graphic 1 as we increase the number of cores for our execution, the more fast the execution of the program becomes. So we can conclude that doing a parallelization of the problem improves the performance. As we can see, for both examples the improvement is pretty similar. Never the less, the improvement it's mostly linear.

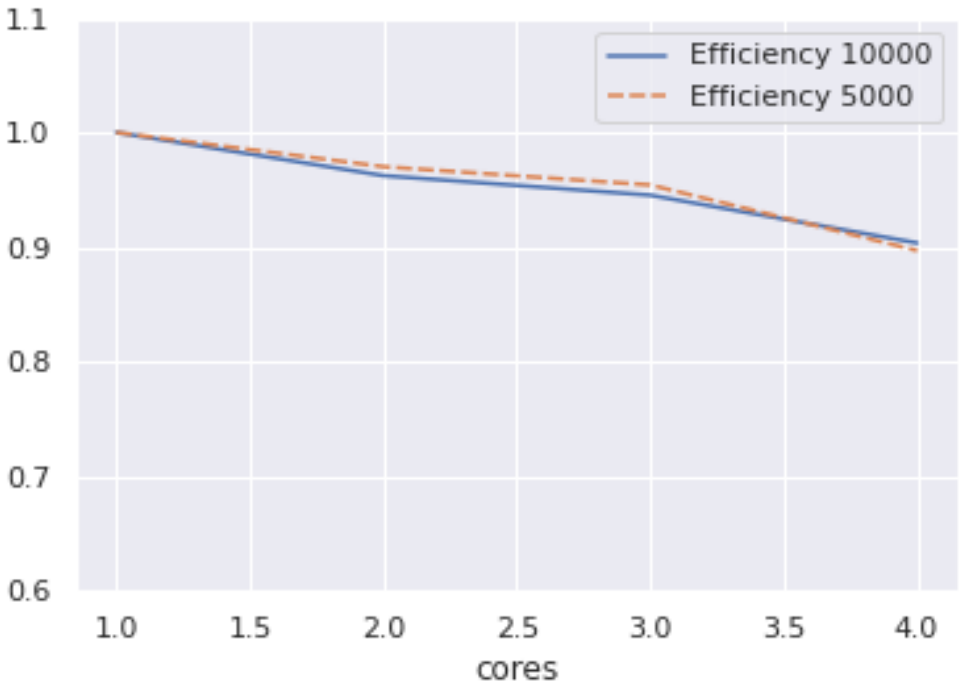

Figure 2: Efficiency for 5000x5000 and 10000x10000

In terms of efficiency, we can see that it has the tendency to go down. Even though it does not in a radical way. it keeps going down in a linear way with a low slope.

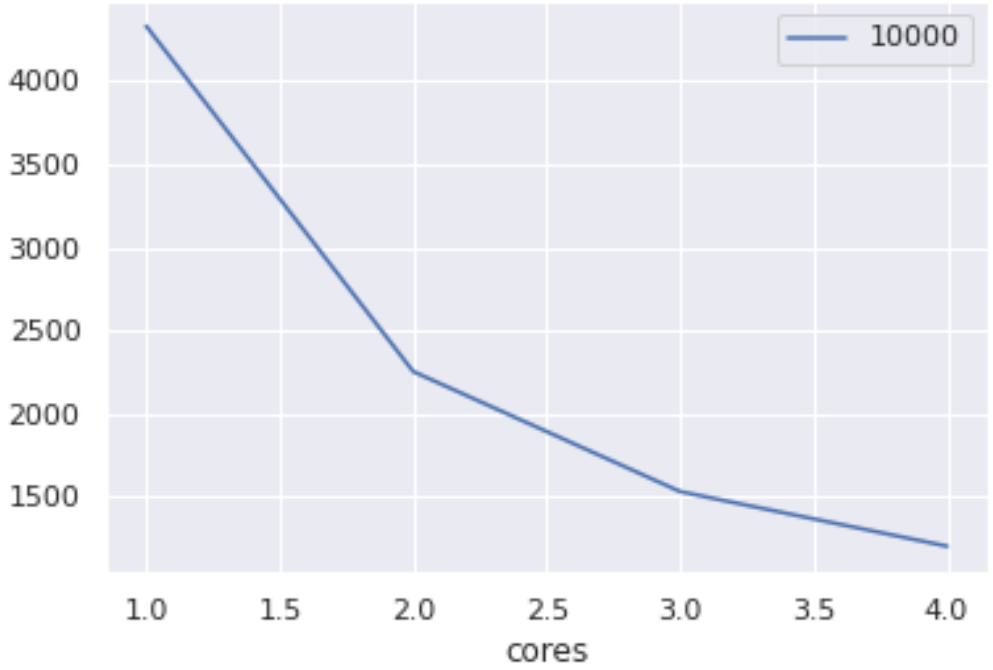

Figure 3: Time for 10000x10000

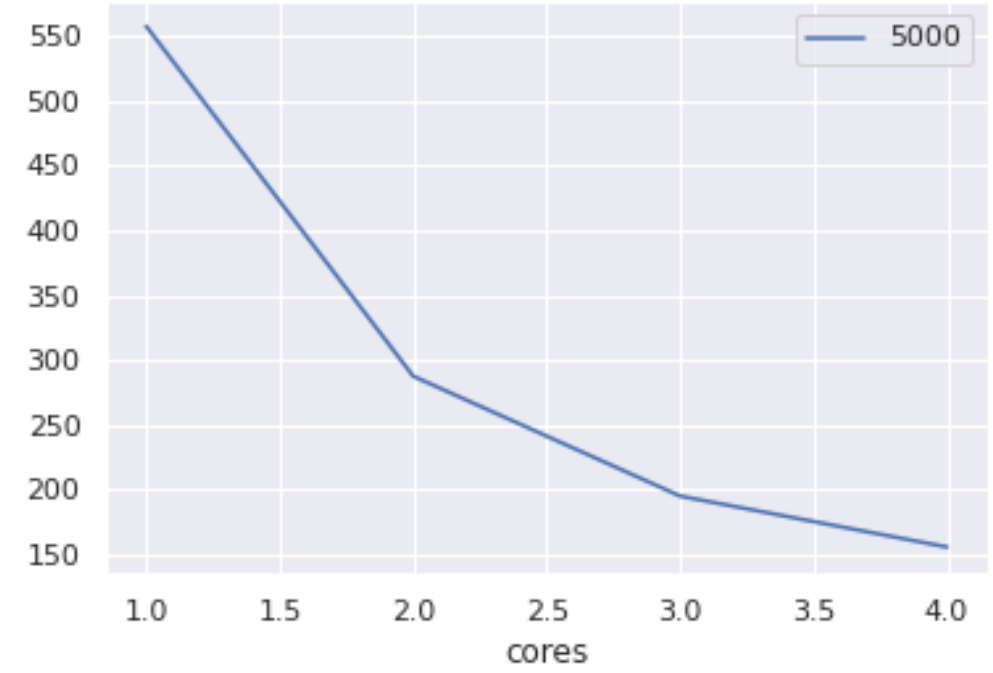

Figure 4: Time for 5000x5000

In terms of time, as we can see on the figures 3 and 4, they have the same curve of time.

Nevertheless if we focus on the time of execution in the y axis we can notice how the time is a lot more for the execution of 10000x10000.

5 Conclusions

Seeing the similarity that these solutions have in terms of the shapes of the plots. We can say that the program will improve in a linear way, and with a negative efficiency always.