OpenMP Programming Assignment:

Finite Differences Report

Sofia Almirante Castillo (1527718)

Pau Reig Llunell (1527816)

December 2022

Study of the steps of parallelization

Our aim is to study the impact in execution time of the program after the parallelization of different parts of the code provided. First, we compared the execution time of the program provided with the one observed after the parallelization of the 3 functions of the program, being the Initial Conditions, the Body and the Error (using the *for* and *reduction* method). The code implemented is shown below, and the numerical results are shown in Table 1 and represented in Fig. 1.

```
// Initial Conditions
  #pragma omp parallel for private(x)
  for (x = 1; x < X; x++)
  {
    U[x][0] = sin( x * M_PI / (double) X );
    U[x][1] = U[x][0] * cos( M_PI / (double) T );
  }
```

```
// Body
    for (t = 1; t <=T; t++)
  {
        #pragma omp parallel for private(x) shared(U,L)
    for (x = 1; x < X; x++)
    {
      U[x][t+1] = (2.0 * (1.0 - L) * U[x][t] +
                  L * (U[x-1][t]+U[x+1][t]) -
                  U[x][t-1]);
    }
  }
```

```
//Checksum
    #pragma omp parallel for private(x) shared(U) reduction(+:S)
  for (x = 1; x < X; x++)
  {
```

```
    S += U[x][T+1];
  }
```

Table 1: Execution time in seconds for different parts of the code parallelized and different number of threads. The size of the matrix nxn is n = 10000 and the maximum number of iterations is T = 10000.

Method	**Threads**	**Execution time (s)**
-	-	1.00
CI	-	1.10
	4	1.11
	8	1.12
Body	-	0.42
	4	0.43
	8	0.42
	16	1.41
Reduction	-	1.12

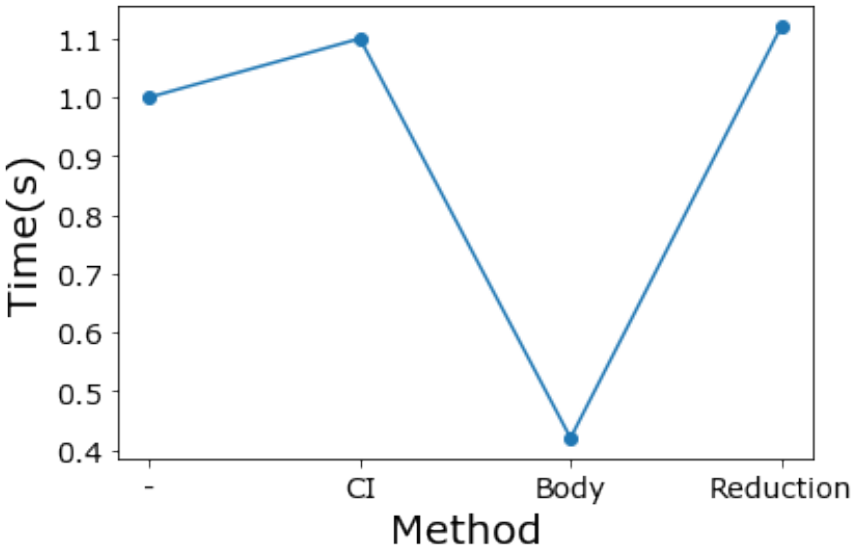

Figure 1: Execution time in seconds in function of the part of the code parallelized. The size of the matrix nxn is n = 10000 and the maximum number of iterations is T = 10000.

We observe that the Initial Condition *for* parallelization increases the execution time by 10%, while the error reduction does it by 12%. In the other hand, the parallelization of the main part of the program (the body part, which represents the majority of the execution time) presents an improvement of 58% in the execution time. For this reason, we have decided to keep only the parallelization in the body part of the program.

Study of the size of the matrix

With the decision of only using the body parallelization strategy for the rest of the tests, we continued our study of the impact in execution time in function of the size of the matrix. The results are shown in Table 2 and represented in Fig. 2. We observe a linear tendency.

Table 2: Execution time for different matrix sizes after parallelizing the body of the program. The number of threads is by default (8) and the maximum number of iterations is T = 10000.

Constants	**Size**	**Execution time (s)**
T = 10000	2500	0.12
	5000	0.22
	10000	0.40
	20000	0.96

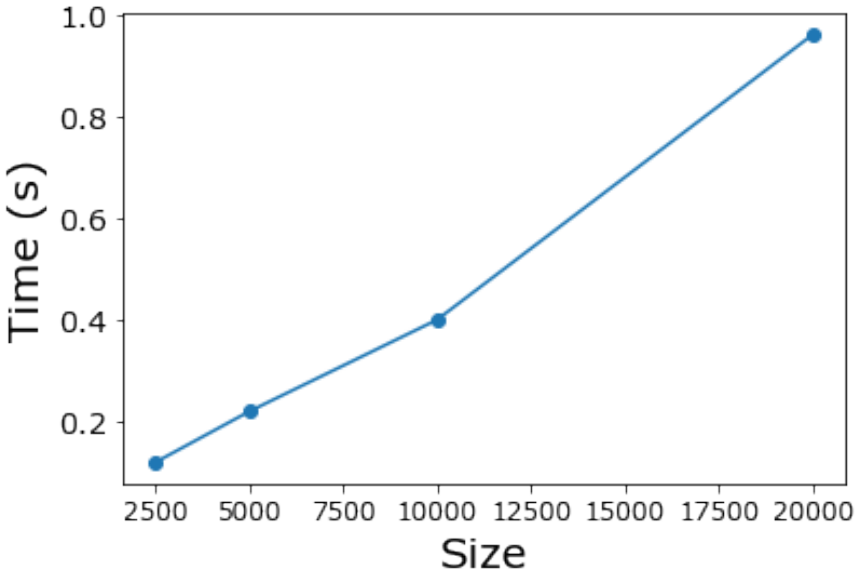

Figure 2: Execution time in seconds in function of the matrix sizes after parallelizing the body of the program. The number of threads is by default (8) and the maximum number of iterations is T = 10000.

Study of the iterations

The results of the study for the maximum number of iterations are shown in Table 3 and represented in Fig. 3.

Table 3: Execution time for different maximum number of iterations T after parallelizing the body of the program. The number of threads is by default (8) and the size of the matrix nxn is n=10000.

Constants	**Iterations T**	**Execution time (s)**
size n = 10000	2500	0.109
	5000	0.21
	10000	0.40
	20000	1.04
	40000	1.59

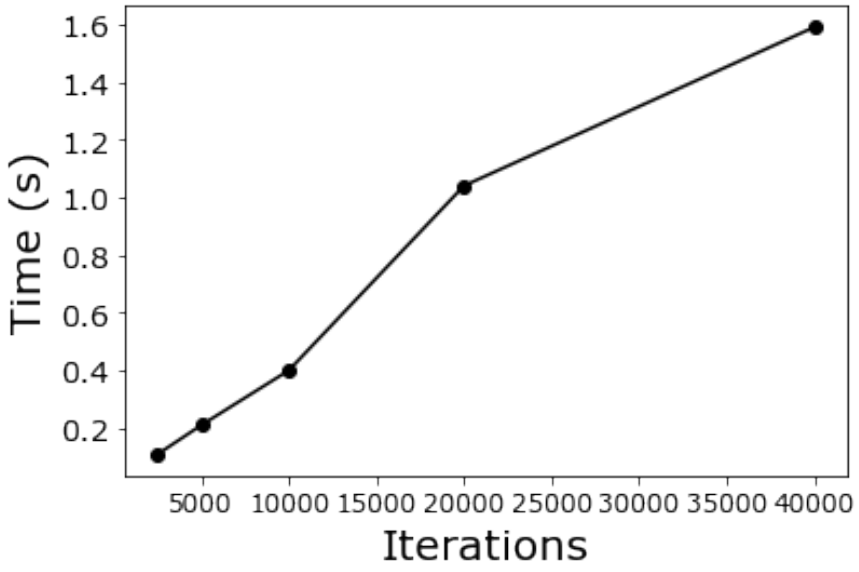

Figure 3: Execution time in function of the maximum number of iterations T after parallelizing the body of the program. The number of threads is by default (8) and the size of the matrix nxn is n=10000.

Since the program spends the most time in computing the body part of the code, increasing the maximum number of iterations of this loop directly increases the execution time, behavior that explains the linear tendency showed. We have to take into account that at some point the program might finish before the maximum number of iterations is reached, which could break the linear tendency.

Study of the iterations

Lastly, the results for the impact of the number of threads are shown in Table 1 and represented in Fig. 4. As we expected, the execution time of the program reaches its minimum when the number of threads passed coincides with the number of units in the computer (which is 8). A better performance is observed when the number of threads is 4 than when it is 16, that can be interpreted that in this case the machine spends less time in reading bigger tasks than distributing more small tasks to a bigger amount of units.

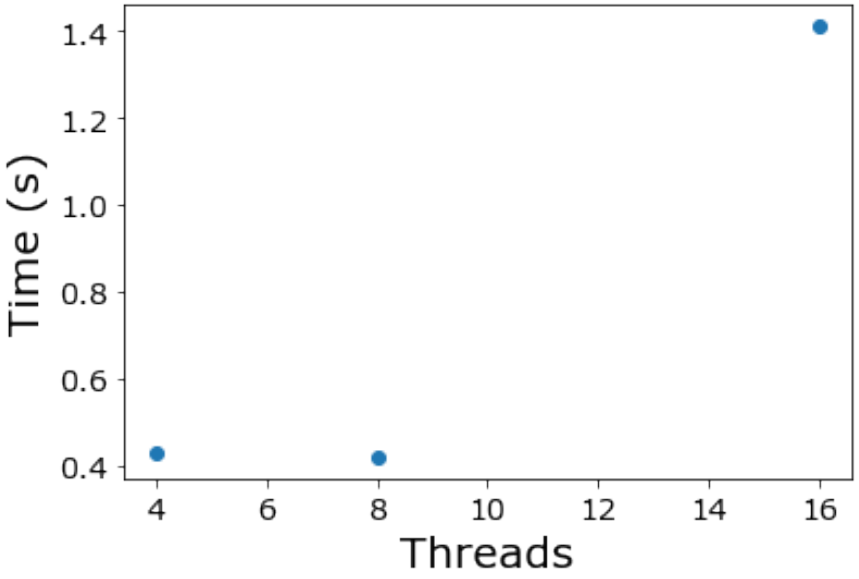

Figure 4: Execution time in function of the number of threads after parallelizing the body of the program. The maximum number of iterations is T = 10000 and the size of the matrix nxn is n=10000.