Cristian Dávila
Enrique González

Deliverable

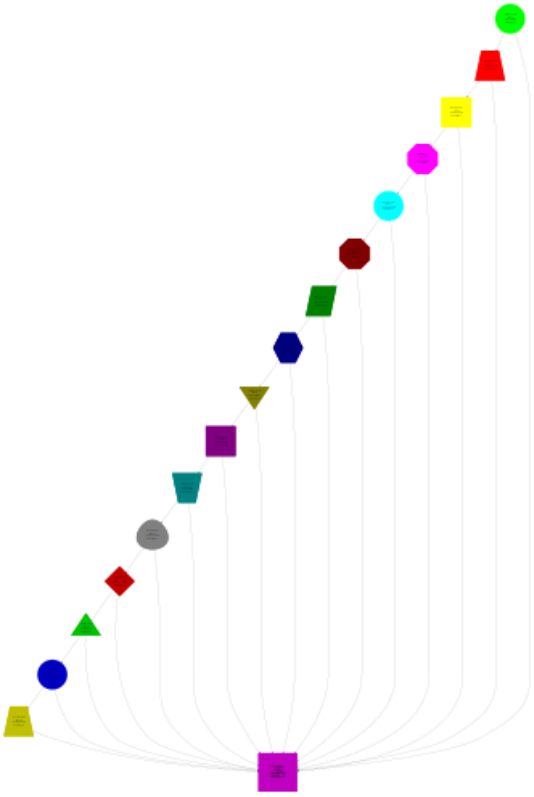
Deliver a compressed tar file (GZ or ZIP) with the PDF that contains the answers to the questions below and the C source codes with the Tareador instrumentation and with the OpenMP parallelization. In the PDF file clearly state the name of all components of the group. Only one file has to be submitted per group through the Raco website.

4.1 Analysis of dependences for the heat equation

Write a text, similar to the one provided below, with one or more paragraphs explaining the dependences that appear in the task graph for each one of the solvers and how do you plan to enforce them in the OpenMP parallel version. We strongly suggest that you include tables and figures to summarize and/or graphically support the information presented.

The analysis of the dependencies caused by the proposed task decomposition for the three solvers of the heat equation has been done using Tareador. Dependences appear when a task reads a memory position previously written by another task. Some dependences may impose task ordering constraints while other may impose data access constraints. For solving the heat equation three different solvers have been used, each with different dependence patterns. For the Jacobi solver:

Using Tarareador on Jacobi, we obtain this graph:



We can see that the dependences enforces a sequential code. cause the dependences of the code, we have disabled the variable ‘sum’, because this variable it’s the cause of the impossibility of make a parallel version.

Jacobi Serial

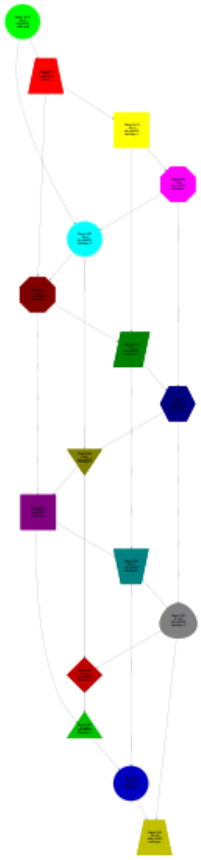
In the parallelization with OpenMP, we plan to guarantee these dependences.

Ignoring this variable, we obtain the second graph:

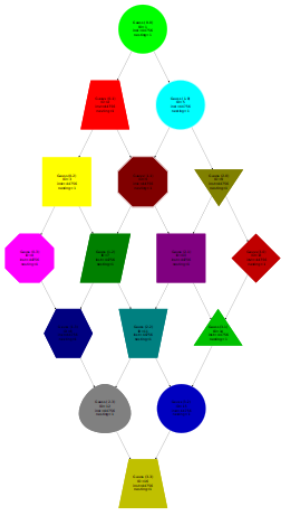
Jacobi Parallel (variable sum disabled)

On the second graph, we can see that we must ensure the synchronization of the variable sum.

Now, we will analyze the dependences of Gauss-Seidel solver.



Analyzing the graph that we obtain using tarareador on Gauss-Sacobi, we can see that the variable that is causing the dependences is the same than on Jacobi: the variable ‘sum’.



Ignoring this variable, and obtaining the corresponding graph, we can see that Gauss-Seidel has a wave-front dependency. That’s why the block (i,j) have dependencies with block (i-1, j) and (i, j-1).

Gauss-Seidel Parallel (variable sum disabled)

Gauss-Seidel Serial

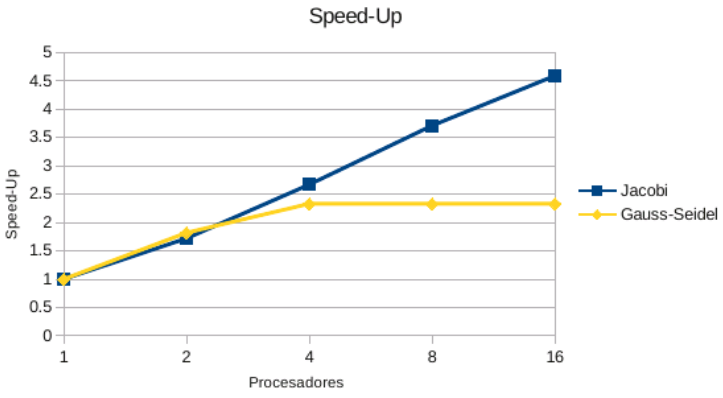
The following table summarizes the predicted speed–up for the parallel execution for the two solvers, with 2, 4, 8 and 16 processors with respect to the simulation with just 1 processor.

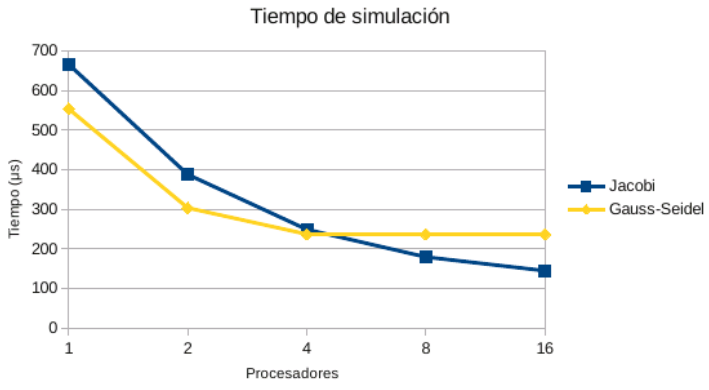
Jacobi

# CPUs	Estimated time	Speed Up
1	666,546 µs	1
2	388,370 µs	1.72
4	249,282 µs	2.67
8	179,738 µs	3.71
16	144,966 µs	4.59

Gauss-Seidel

# CPUs	Estimated time	Speed Up
1	553,863 µs	1
2	303,724 µs	1.82
4	237,252 µs	2.33
8	237,252 µs	2.33
16	237,252 µs	2.33





We can see on the obtained data on the simulations that the best algorithm is Jacobi-solver, although for a small number of cores, Gauss-Seidel is better than Jacobi.
We see that with 4 cores, on the algorithm Gauss-Seidel we can not appreciate any improvement, because the maxim number of blocks that we can process at the same time are 4, cause the dependence on the calculation (nby = 4). However, Jacobi's algorithm continues the improvement with the increase of CPUs.

Generic competence "Tercera llengua". As you know, this course contributes to the generic competence "Tercera llengua", in particular G3.2 "To study using resources written in English. To write a report or a technical document in English. To participate in a technical meeting in English." If you want this competence to be evaluated, write THIS section in english.

4.2 Execution analysis

1. Plot and comment the speedup achieved by the OpenMP parallel version, with respect to sequential execution time, for the different solvers (Jacobi and Gauss-Seidel).

Ahora comprobaremos si las predicciones que realizamos a través de las simulaciones con Dimemas coinciden con la ejecución de estos algoritmos ejecutados en boada:

Jacobi

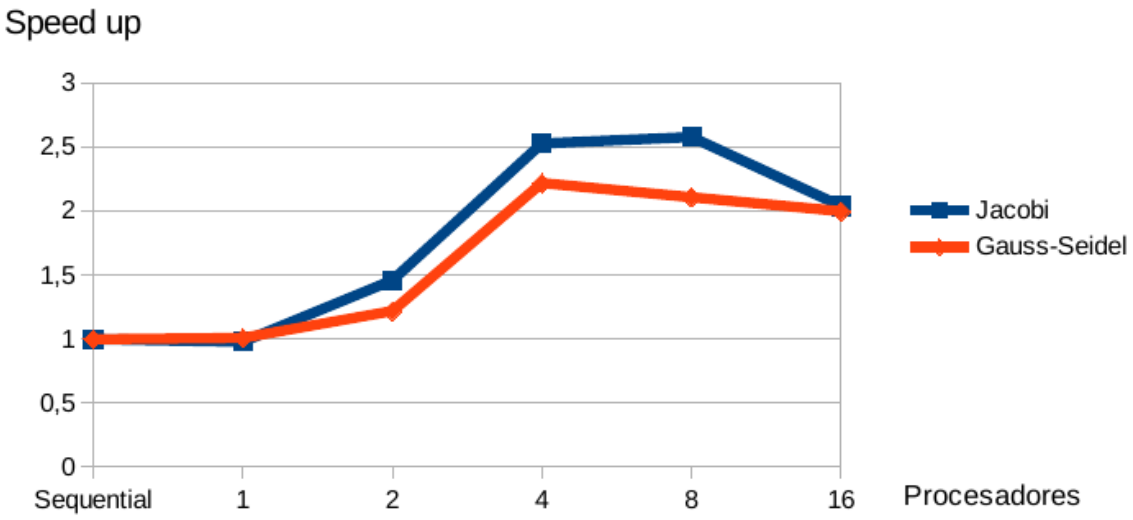
# Treads	Seq. version time	Paral. version time	Speed Up
Sequential	5.264 s	--	1
1	--	5.346 s	0.98
2	--	3.581 s	1.46
4	--	2.082 s	2.53
8	--	2.041 s	2.58
16	--	2.585 s	2.04

Gauss-Seidel

# Treads	Seq. version time	Paral. version time	Speed Up
Sequential	6.369 s	--	1
1	--	6.336 s	1.01
2	--	5.221 s	1.22
4	--	2.867 s	2.22
8	--	3.023 s	2.11
16	--	3.190 s	2.00

Podemos observar que la predicción que hicimos de Gauss se acerca bastante a la realidad ya que el Speed Up máximo que conseguimos en esta ejecución con 4 threads coincide con la simulación que hicimos con Dimemas para 4 o más CPUs.

En cambio, Jacobi no llega a alcanzar el Speed Up que predijimos anteriormente. Pese a que al principio, en las ejecuciones con 2 y 4 threads, sí cumple con las expectativas, finalmente el Speed Up obtenido en las simulaciones para más de 4 CPUs dista mucho de la realidad.



2. In the parallelization of Gauss-Seidel, analyze the impact on the parallel execution time of the block size ”by” (or equivalently the number of blocks ”nby”)1. For example try with nby=2*nbx, nby=4*nbx, ... Is there any performance impact? Is there an optimal point? Justify your answer.

nby = nbx = #Threads

#Threads (nbx x nby)	**Jacobi**	**Gauss-Seidel**
2 (2x2)	3.525 s	4.885 s
4 (4x4)	2.082 s	2.867 s
8 (8x8)	1.703 s	1.798 s
16 (16x16)	1.967 s	1.368 s

nby = 2·nbx = 16

#Threads	**Jacobi**	**Gauss-Seidel**
2	4.530 s	3.440 s
4	2.466 s	2.014 s
8	1.732 s	1.399 s
16	1.949 s	1.543 s

nby=4·nbx = 16

#Threads (nbx x nby)	Jacobi	Gauss-Seidel
2	4.444 s	3.243 s
4	2.383 s	1.939 s
8	2.431 s	2.020 s
16	3.005 s	2.344 s

En el apartado anterior veíamos como el método de Jacobi era más rápido que el de Gauss, pero nos limitabamos a un sólo número de bloques. Ahora que hemos hecho un estudio con diferentes tamaños de bloque, incrementando nby, hemos podido ver que Jacobi obtiene un peor tiempo de ejecución. Sin embargo, el método de Gauss obtiene un mejor tiempo de ejecución, ya que el número de sincronizaciones forzadas (flush) disminuye, lo que hace que obtengamos un mejor resultado con este método.