Albert Bernal (2205) i Sergi Bertran (2206) 08-10-2020

Parallelism - Lab 1

1.- Node architecture and memory

Number of sockets per node: 1
Number of cores per socket: 6
Number of threads per core: 2
Maximum core frequency: 2395 MHz
L1-I cache size (per-core): 32K
L1-D cache size (per-core): 32K
L2 cache size (per-core): 256K
Last-level cache size (per-socket): 12288K
Main memory size (per socket): 12GB
Main memory size (per node): 24GB

2.- Strong vs. weak scalability

In strong scalability the number of threads is changed with a fixed problem size. In this case parallelism is used to reduce the execution time of the program. Strong scalability means that increasing the number of threads or processors (for example) doubling the cores should, ideally, reduce by half the time elapsed in its execution.
In weak scalability the problem size is proportional to the number of threads. In this case parallelism is used to increase the problem size for which the program is executed. In other words, weak scalability means that the improvement is not in reducing time, but the ability to process more operations. So, if we double the processing power, the problem size for which the program is executed should be doubled (or close to it) as well.

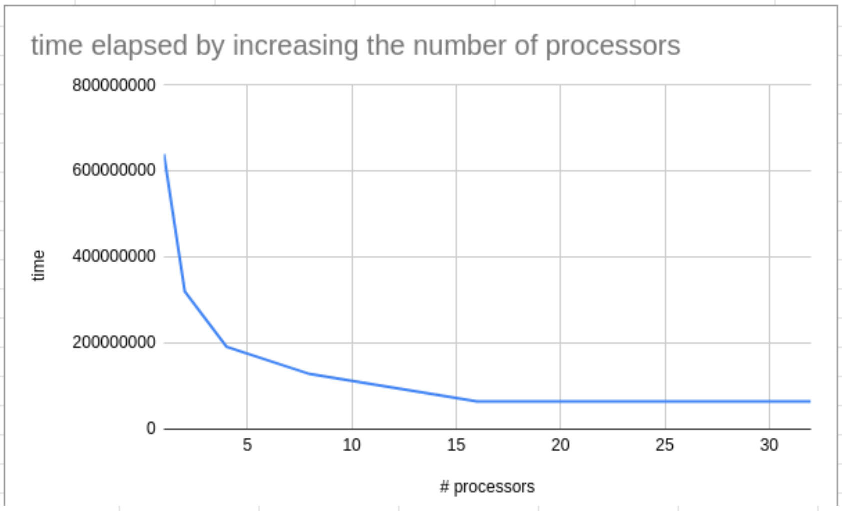
3.- Analysis of task decompositions for 3DFFT

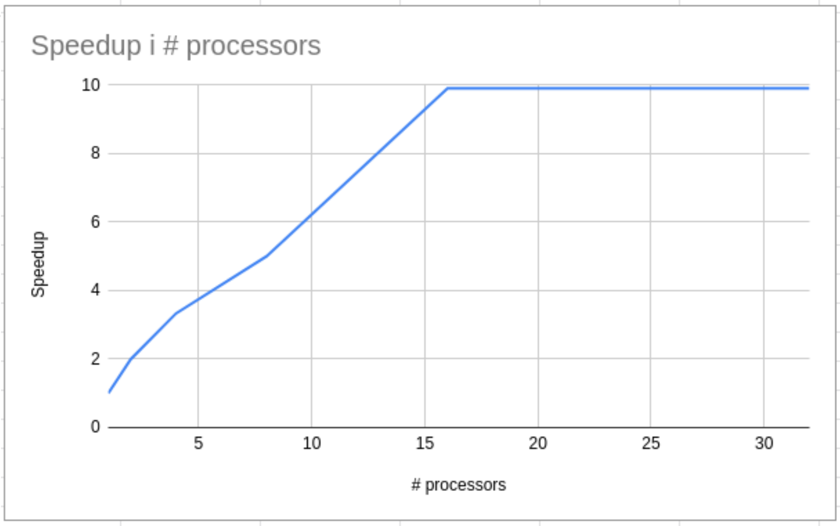
Version	t1 (ns)	t_∞ (ns)	Parallelism
seq	639.780.001	639.707.001	1,00
v1	639.780.001	639.707.001	1,00
v2	639.780.001	361.439.001	1,77
v3	639.780.001	154.939.001	4,13
v4	639.780.001	64.614.001	9,90
v5	639.780.001	39.104.001	16,36

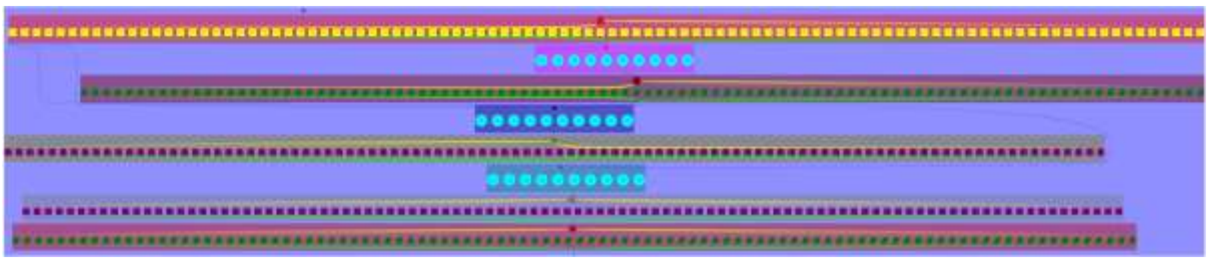
In v4 we can see that it would need 10 processors in order to reach t_∞
Instead of parallelising loop k, we have parallelising loop j. This way, there are more tasks to be parallelized. As a result, it would need 101 processors in order to reach t_∞.

The times obtained by running the simulations on v4 are the following:
1 CPU: 639.780.001
2 CPUs: 320.257.001
4 CPUs: 191.882.001
8 CPUs: 127.992.001
16 CPUs: 64.614.001
32 CPUs: 64.614.001

Following are two plots that describe through execution time and speedup how the improvement by increasing the number of processors diminishes as we increase the processor quantity.

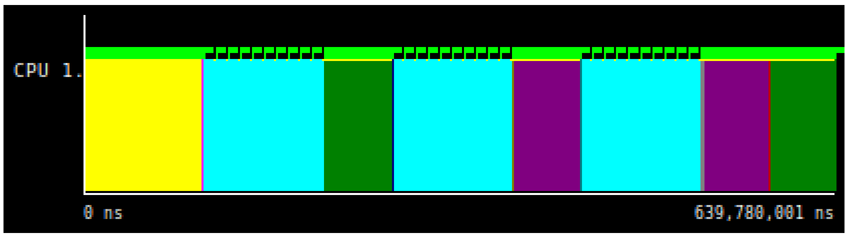


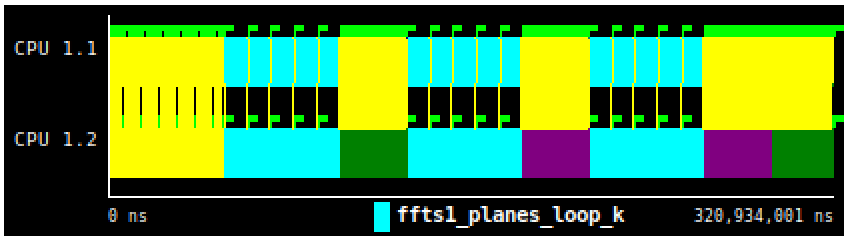


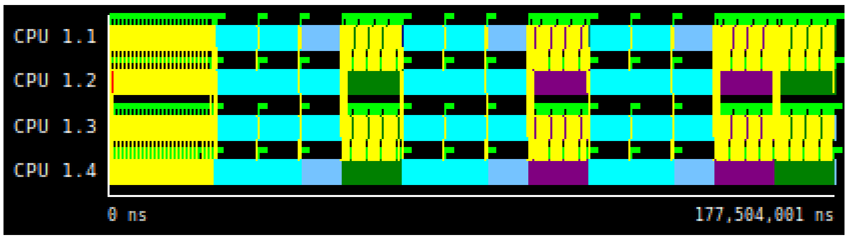

Task dependence graph v5

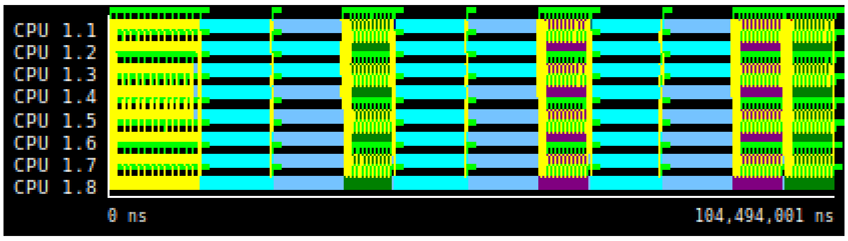
Following are the simulations given by Tareador regarding v5 and changing the number of processors. In the images of the 16 and 32 processor runs, the time elapsed isn't quite clear, here is every value recorded in nanoseconds:

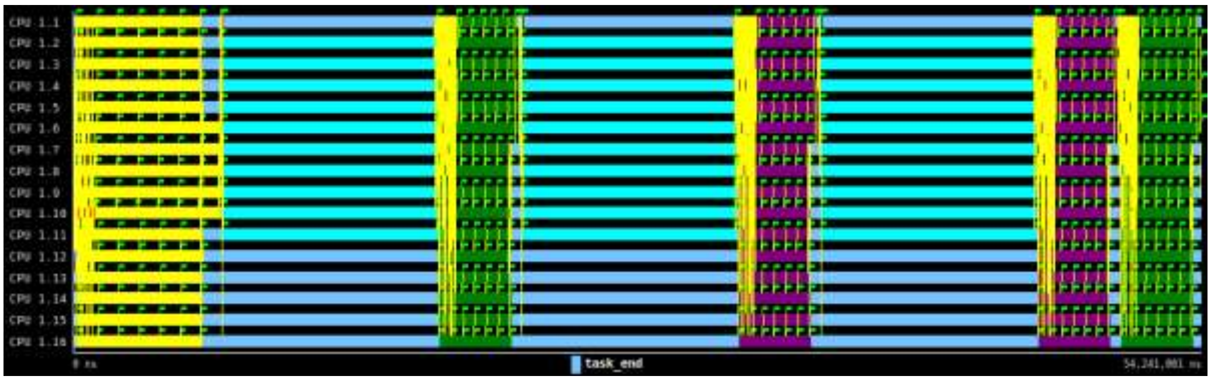
1 CPU: 639.780.001
2 CPUs: 320.934.001
4 CPUs: 177.504.001
8 CPUs: 104.494.001
16 CPUs: 54.241.001
32 CPUs: 44.476.001

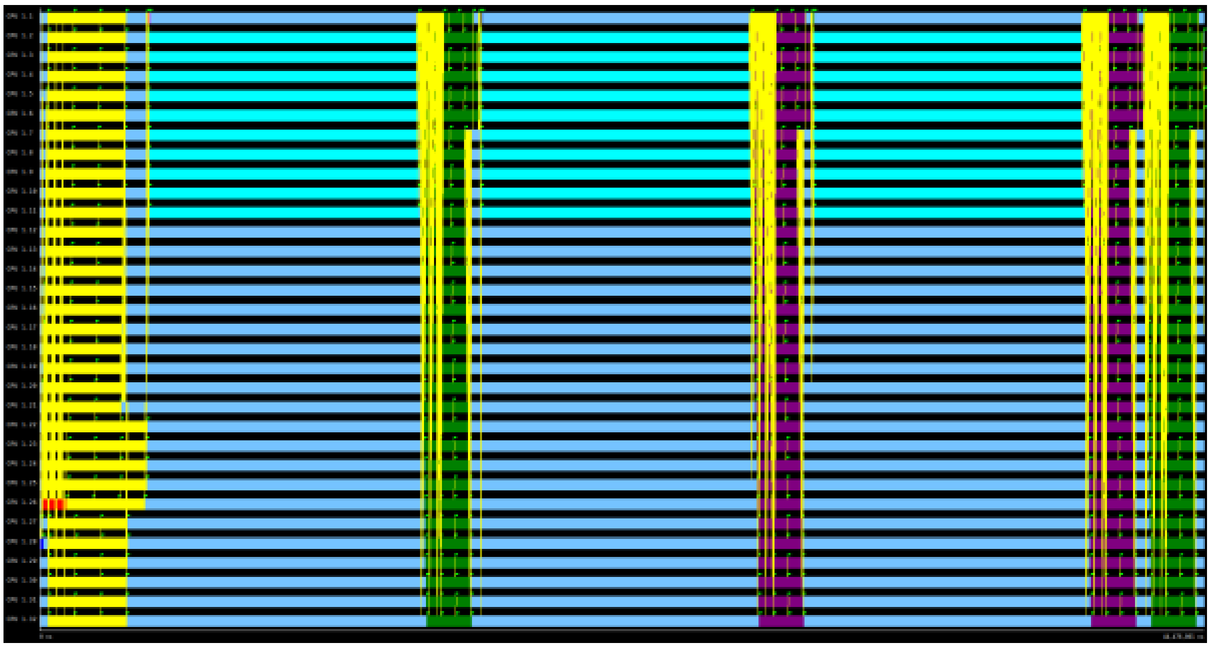




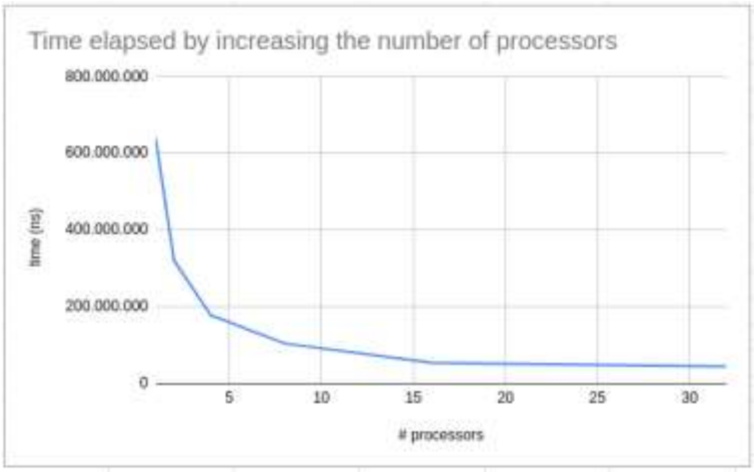


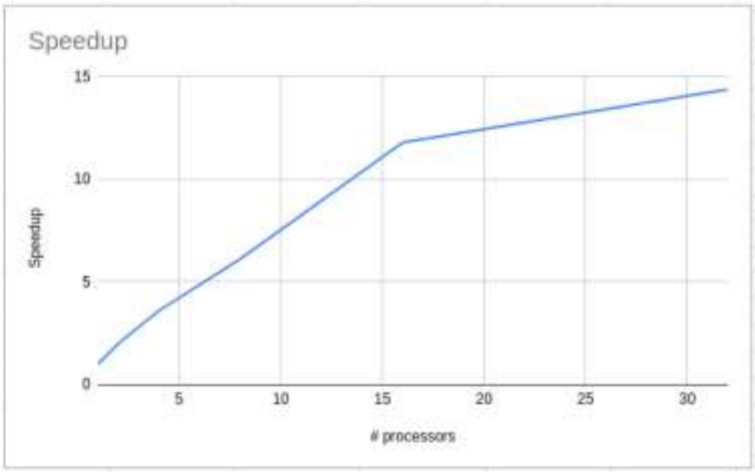






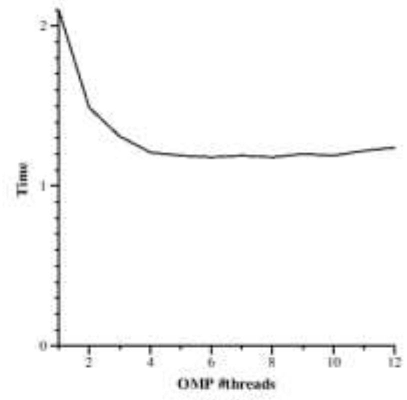
And the plots:



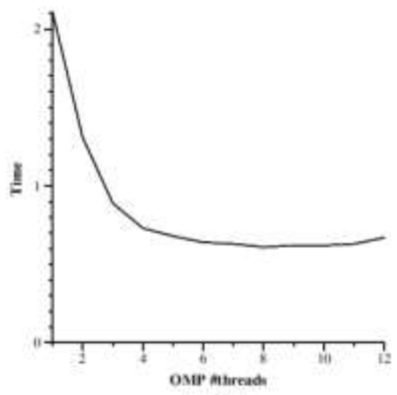


4.- Understanding the parallel execution of 3DFFT

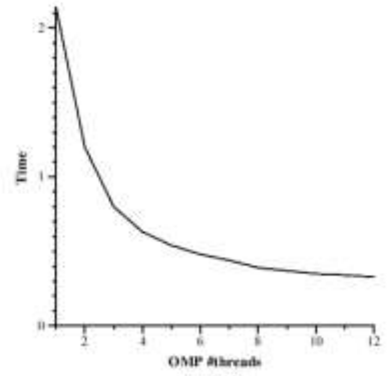
The strong scalability plots that were obtained by executing the submit-strong-omp.sh script with the different modifications to 3dfft_omp (no changes, improved φ 3dfft_omp version and the version that reduces parallelisation overheads) are the following:



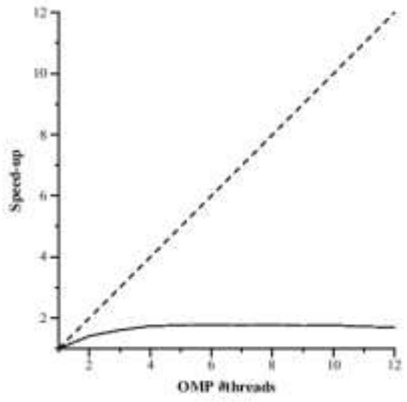
par2205
Min elapsed execution time
Wed Oct 7 21:57:14 CEST 2020



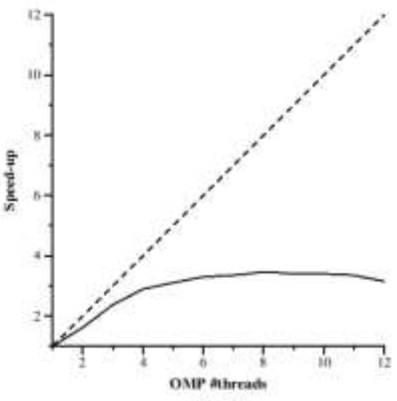
par2205
Min elapsed execution time
Wed Oct 7 21:39:06 CEST 2020



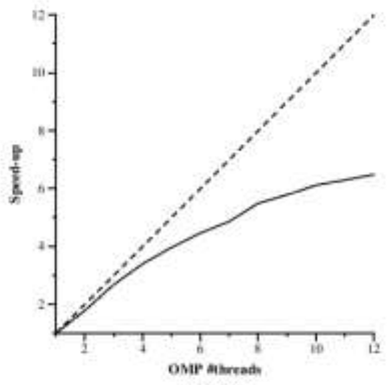
par2205
Min elapsed execution time
Wed Oct 7 21:50:51 CEST 2020



par2205
Speed-up wrt sequential time
Wed Oct 7 21:57:14 CEST 2020



par2205
Speed-up wrt sequential time
Wed Oct 7 21:39:06 CEST 2020



par2205
Speed-up wrt sequential time
Wed Oct 7 21:50:51 CEST 2020

As shown in these plots, the code optimization applied resulted in great improvements regarding the scalability of the program. In the first version, parallelisation is barely taken advantage of, showing that, no matter the amount of cores a computer has, these are useless if we don't use them correctly.
When φ is improved, and therefore the percentage of parallelizable code in our program increases, a better result is obtained, but the speedup seems to reach its highest value from 4 to 12 threads, not seeing the strong scalability wanted.
In the third execution is where the slope is much better, showing that increasing the number of threads does make a difference and the program is strongly scalable.

Version	**φ**	**S_∞**	**T1 (ns)**	**T8 (ns)**	**S8**
initial version	0,664	2,97	2.441.509.472	1.442.448.356	1,69
improved φ	0,905	10,57	2.392.321.907	902.009.720	2,65
reduced parallelisation overheads	0,906	10,63	2.293.938.687	597.999.871	3,84