DELIVERABLE LAB1

Paral·lelisme i concurrència

Toni Cifre i Eva Hernández
Grupo: paco0013
Fecha: 7/10/2019
3r año, 1r cuatrimestre

Índice

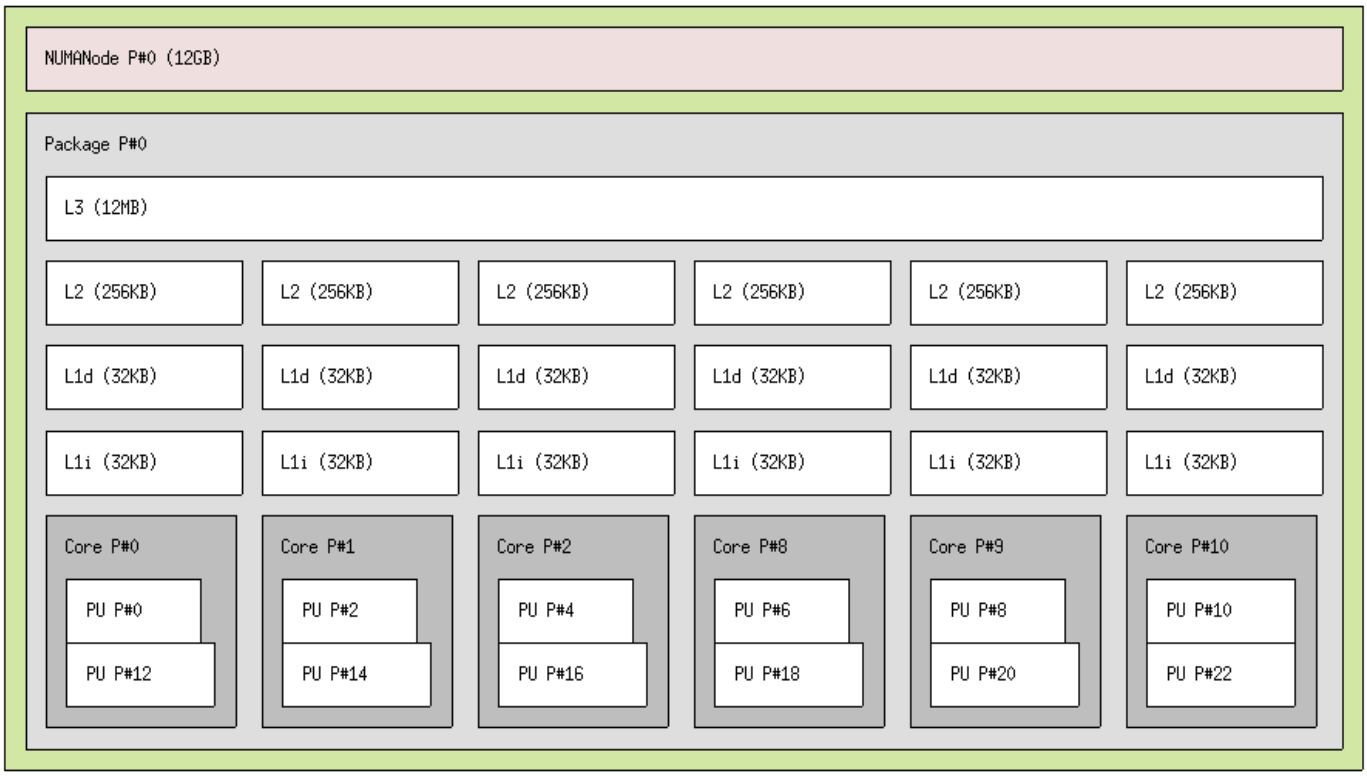
Node architecture and memory

Describe the architecture of the boada server. To accompany your description, you should refer to the following table summarising the relevant architectural characteristics of the different node types available:

	boada-1 to boada-4	boada-5	boada-6 to boada-8
Number of sockets per node	2	2	2
Number of cores per socket	6	6	8
Number of threads per core	2	2	1
Maximum core frequency	2395.0000	2600.0000	1700.0000
L1-I cache size (per-core)	32K	32K	32K
L1-D cache size (per-core)	32K	32K	32K
L2 cache size (per-core)	256K	256K	256K
Last-level cache size (per-socket)	12288K	15360K	20480K
Main memory size (per socket)	12GB + 12MB	31GB + 15MB	16GB + 20MB
Main memory size (per node)	320KB	320KB	320KB

Boada está compuesta por 8 nodos (boada-1 al boada-8) y cada uno de ellos tiene un procesador de diferente generación. La diferencia que tiene boada-1 al boada-4 con el restos de nodos de boada es que este es el único que puede ejecutar interactivamente y los demás deben hacerlo con cola. También podemos observar que el número de cores por socket y el número de threads por core es el mismo en boada.1 - boada.4 y boada.5 pero en boada 6-8 varia.

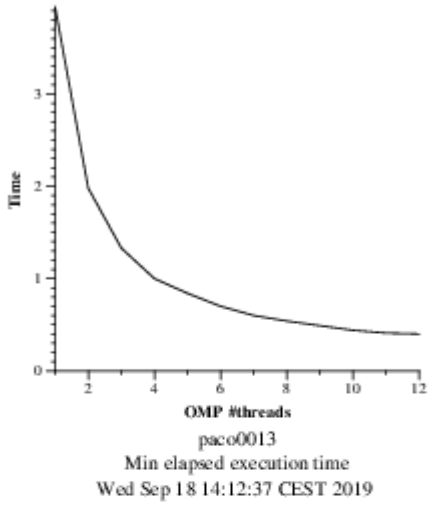
Diagrama boada 1 - boada 4

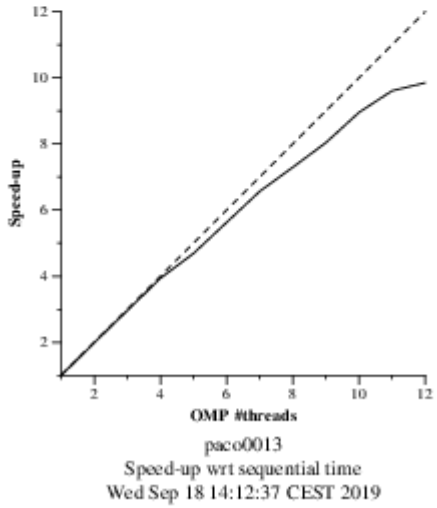


Strong vs. weak scalability

Briefly explain what strong and weak scalability refer to. Exemplify your explanation using the execution time and speed–up plots that you obtained for pi omp.c. Reason about the results obtained.

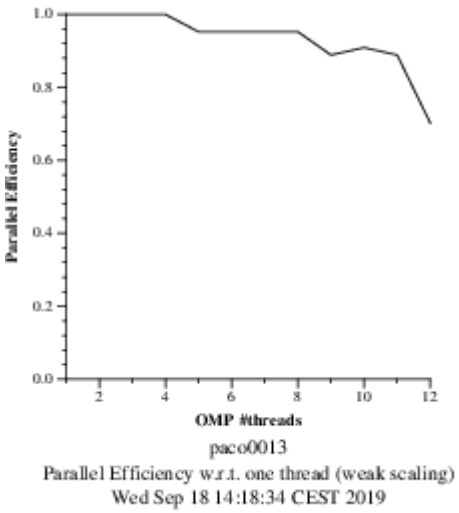
Strong scalability





Es el tiempo de ejecución del programa dependiendo de la variación de los threads que se utilizan. Como más threads se utilizan , mejor es el tiempo de ejecución.
Pero tal y como podemos observar en la segunda gráfica, el número de threads i la velocidad no es proporcional debido a la carga de asignación de tareas de cada thread, es decir, que si tuviéramos un número muy elevado de threads no mejoraríamos el resultado.

Weak scalability



La weak scalability relaciona el número de threads utilizados con la eficiencia de la ejecución en paralelo. En este caso, cuantos más threads se utilizan más disminuye la eficiencia de la ejecución en paralelo debido a la distribución de trabajo entre los threads.

Analysis of task decompositions for 3DFFT

In this part of the report you should summarise the main conclusions from the analysis of task decompositions for the 3DFFT program. Backup your conclusions with the following table properly filled in with the information obtained in the laboratory session for the initial and different versions generated for 3dfft tar.c, briefly commenting the evolution of the metrics.

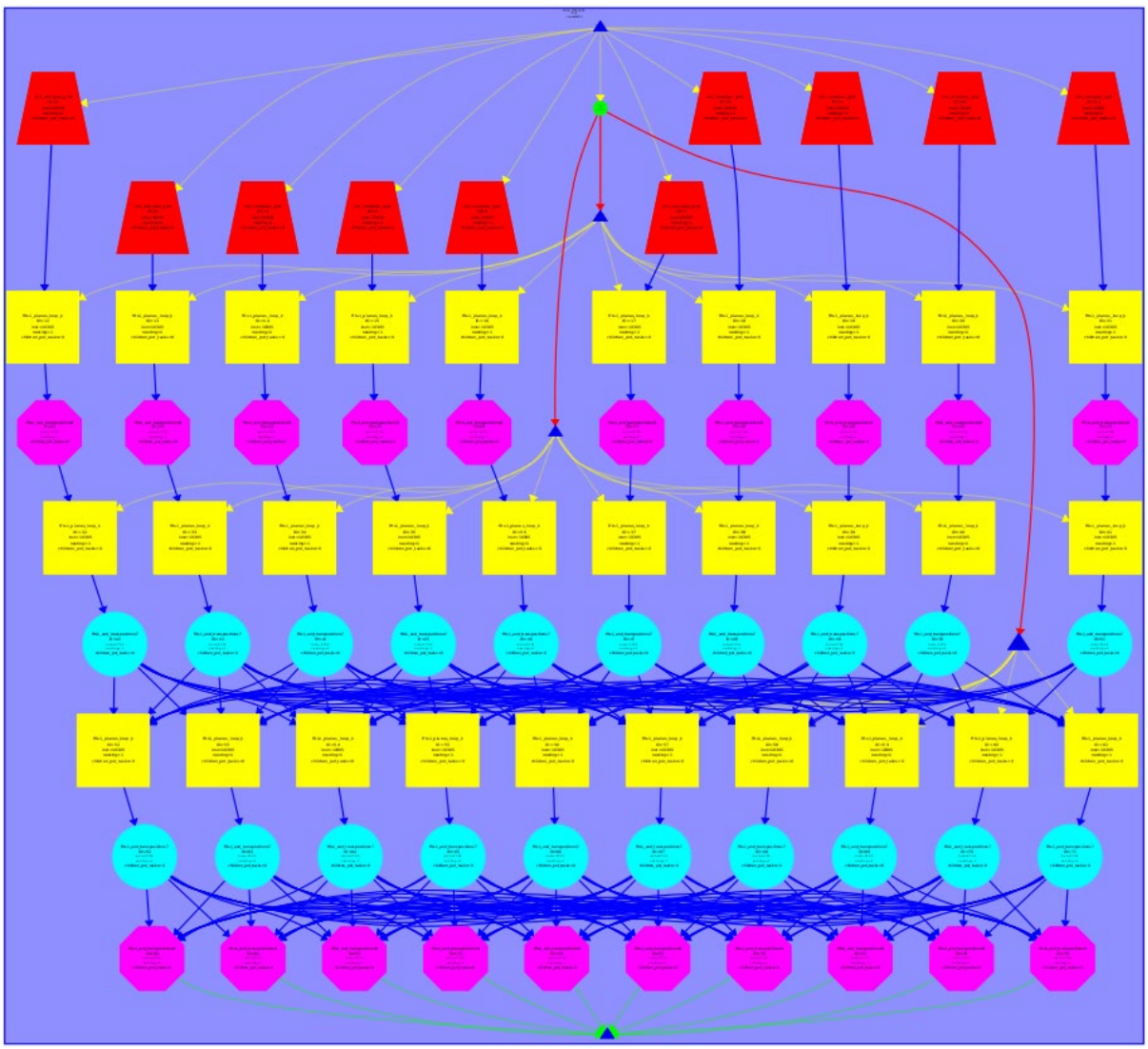
Version	T1	T∞	Parallelism
Seq	639.780,001 ns	639.707,001 ns	T1/Tinf= 1.000114115
v1	639.780,001 ns	639.707,001 ns	T1/Tinf= 1.000114114
v2	639.780,001 ns	361.190,001 ns	T1/Tinf= 1.771311496
v3	639.780,001 ns	154.354,001 ns	T1/Tinf= 4.144887705

For versions v4 and v5 of 3dfft tar.c perform an analysis of the potential strong scalability that is expected. For that include a plot with the execution time and/or speedup when using 1, 2, 4, 8, 16 and 32 processors, as reported by the simulation module inside Tareador . You should also include the relevant(s) part(s) of the code that help the reader to understand why v5 is able to scale to a higher number of processors compared to v4, capturing the task dependence graphs that are obtained with Tareador.

Versión 4:

Version=v4	T∞	Parallelism
T1=639.780.001 ns	64.018.001 ns	T1/Tinf= 9.993751617
T2=320.310.001 ns		
T4=165.389.001 ns		
T8=91.496.001 ns		
T16=64.018.001 ns		
T32=64.018.001 ns		

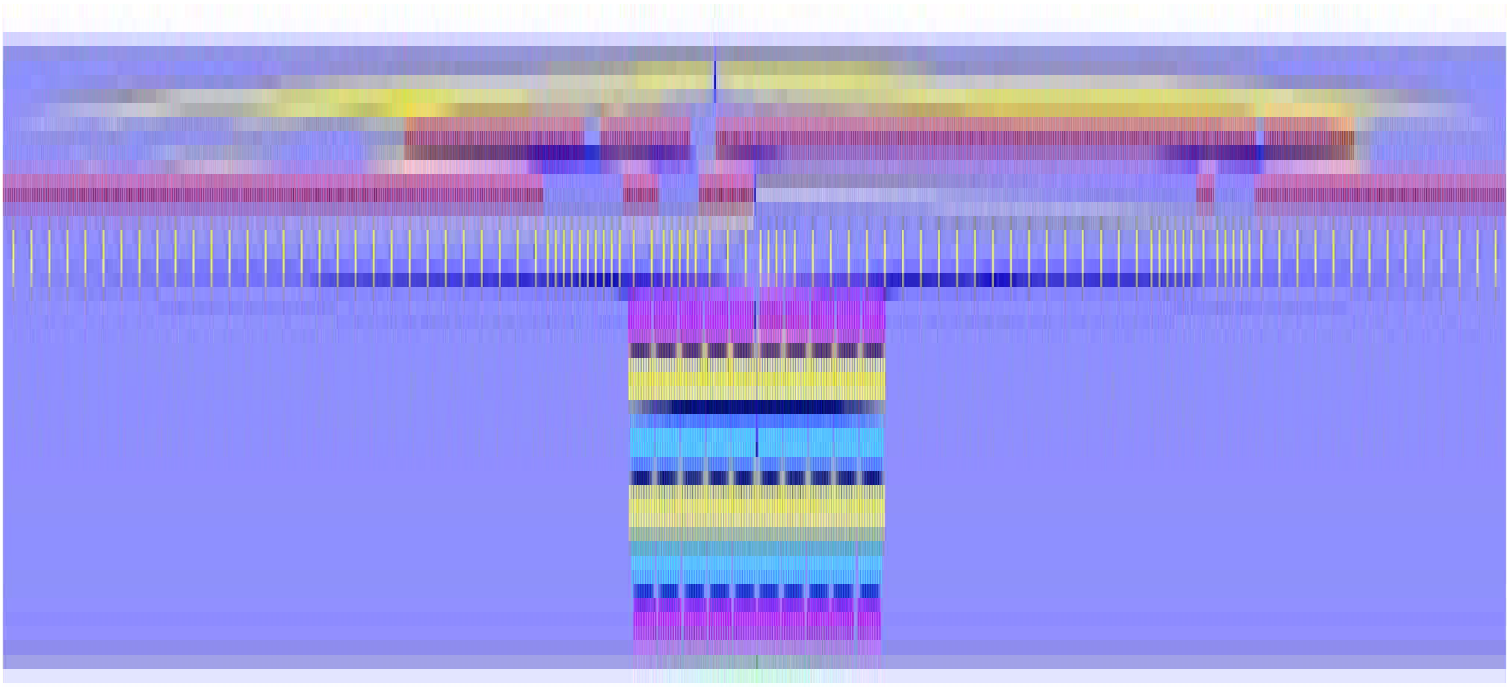
Diagrama



Versión 5:

Version=v5	T∞	Parallelism
T1=639.780,001 ns	16.655.001 ns	T1/Tinf= 38.41368734
T2=319.952.001 ns		
T4=160.048.001 ns		
T8=80.230.001 ns		
T16=43.618.001 ns		
T32=26.875.001 ns		
T128= 16.655.001 ns		

Diagrama



Conclusiones:

Como podemos observar de la versión 2 a la 4 mejoramos respecto a la versión 1 inicial porque cada vez vamos sustituyendo la partes no paralelizables por paralelizables. Como podemos ver en la versión 4 a partir de los 16 threads ya no reducimos el tiempo debido al coste del overhead, pero en la versión 5 disminuimos la granularidad para eliminar el coste del overhead al máximo y conseguir mejor tiempo. Por lo tanto podemos apreciar que el overhead es muy importante a la hora de distribuir las tareas.

Understanding the parallel execution of 3DFFT

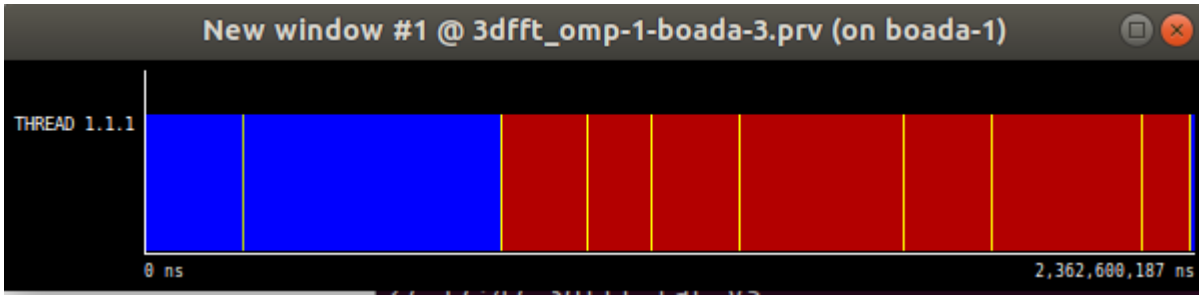
In this final section of your report you should comment about how did you observed with Paraver the parallel performance evolution for the OpenMP parallel versions of 3DFFT. Support your explanations with the results reported in the following table which you obtained during the laboratory session. It is very important that you include the relevant Paraver captures (timelines and profiles of the % of time spent in the different OpenMP states) to support your explanations too.

Versión	φ	S∞	T1	T8	S8
initial version in 3dfft omp.c	Tpar=1.537380 Tpar/T1=0.6939	1/(1-φ)= 3.2669	2.215363s	1.511621s	1.4655545s
new version with improved φ	Tpar=2.082559 Tpar/T1=0.99977	1/(1-φ)= 4347.826	2.083035s	0.729025s	2.8572888s
final version with reduced parallelisation overheads	Tpar=2.230241 Tpar/T1=0.99977	1/(1-φ)= 4347.826	2.230741s	0.53649s	4.170459841s

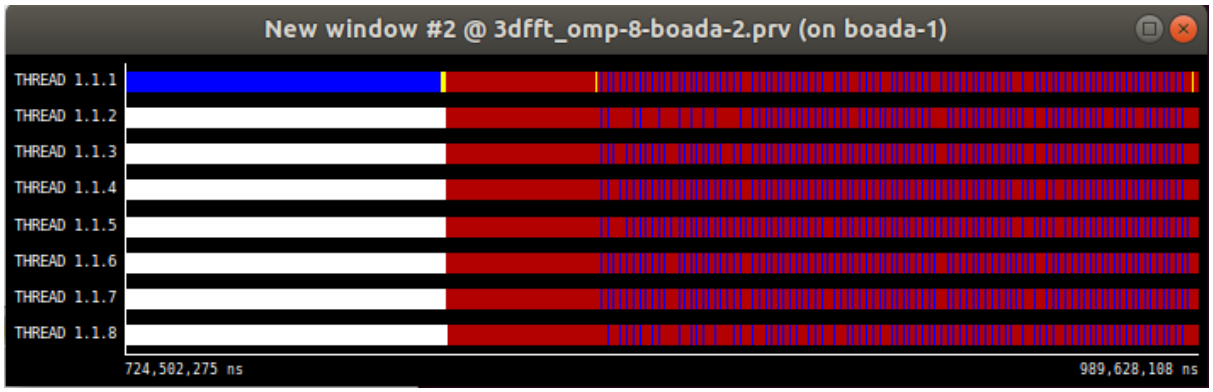
Finally you should comment about the (strong) scalability plots (execution time and speed–up) that are obtained when varying the number of threads for the three parallel versions that you have analysed.

Versión inicial:

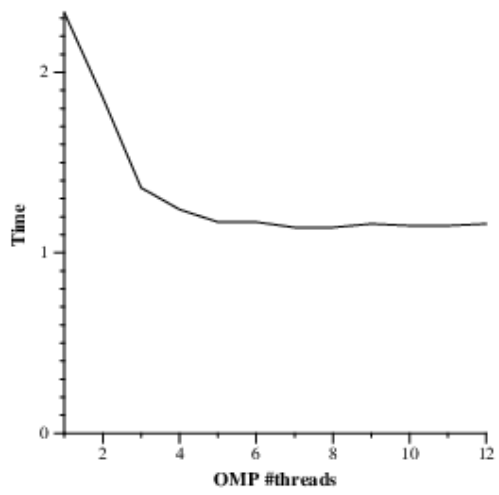
wxparaver con 1 thread:

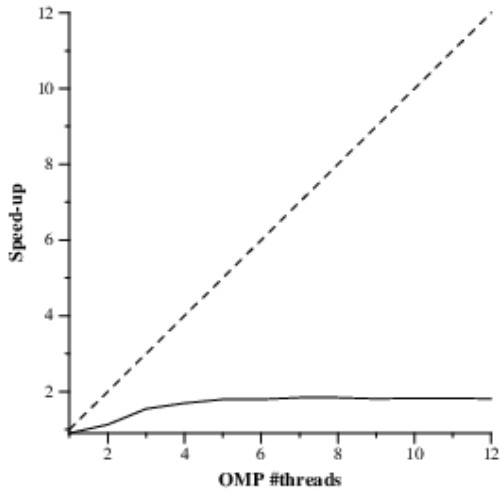


wxparaver con 8 threads:



Strong scalability



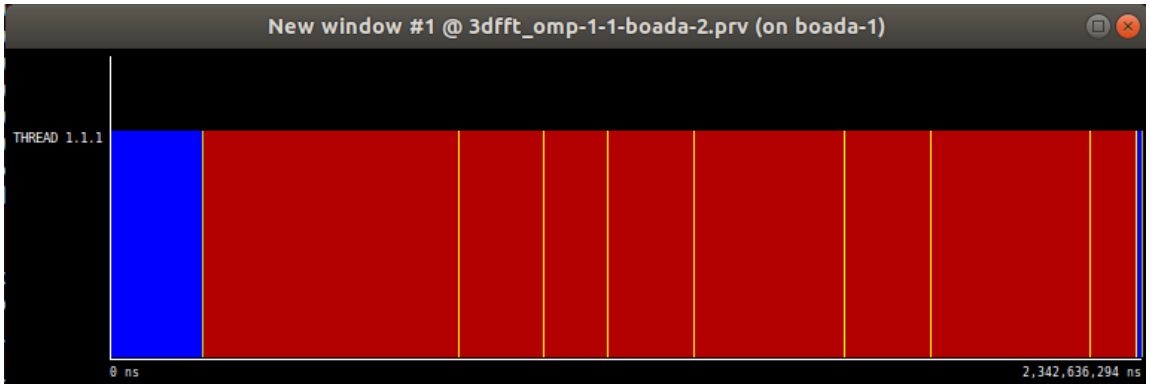


Observaciones:

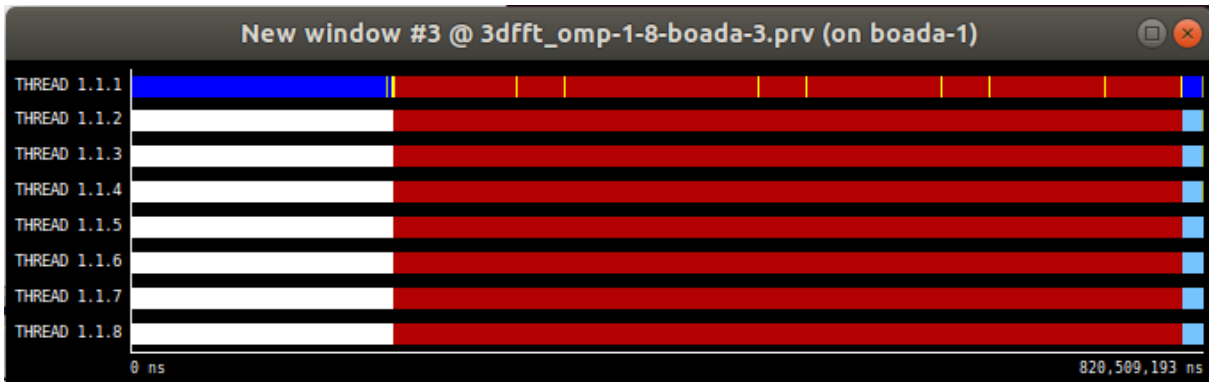
Tenemos una parte paralelizable y una no paralelizable, a cada thread reducimos la parte paralelizable hasta que llega un punto que esta parte es insignificante pero la parte no paralelizable nunca se reducirá por lo tanto si nuestro tiempo no paralelizable es grande solo podremos optimizarlo hasta cierto punto.

Versión con φ mejorado:

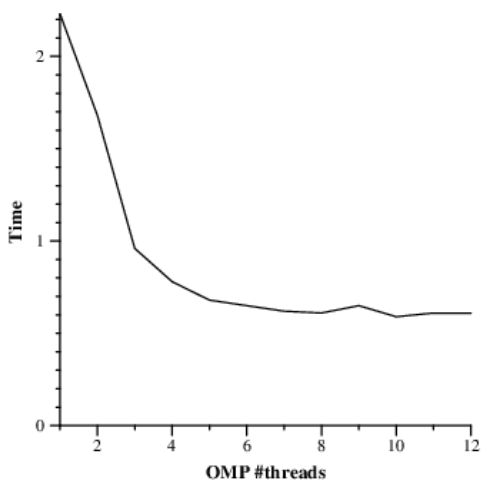
wxparaver con 1 thread:



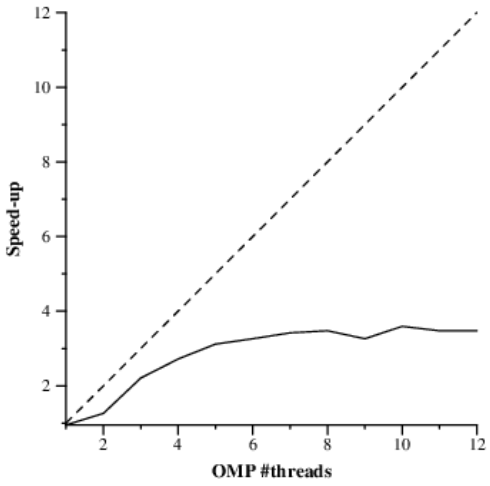
wxparaver con 8 thread:



Strong scalability



paco0013
Min elapsed execution time
Sun Oct 6 12:41:53 CEST 2019



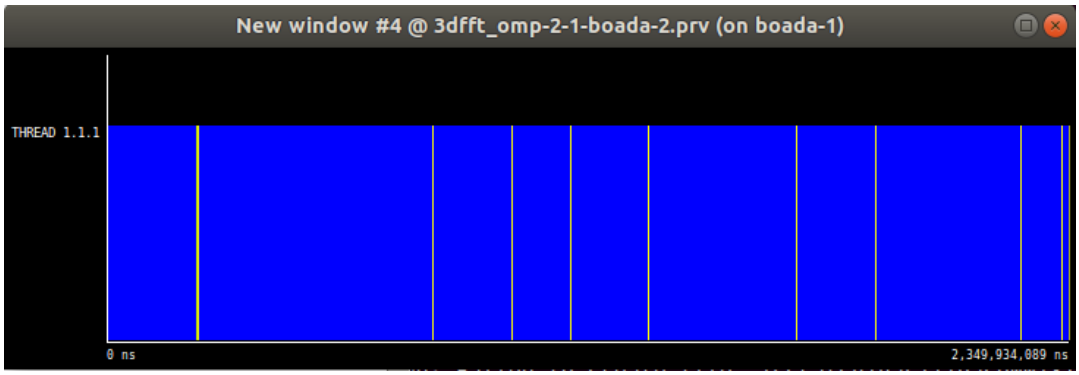
paco0013
Speed-up wrt sequential time
Sun Oct 6 12:41:53 CEST 2019

Observaciones

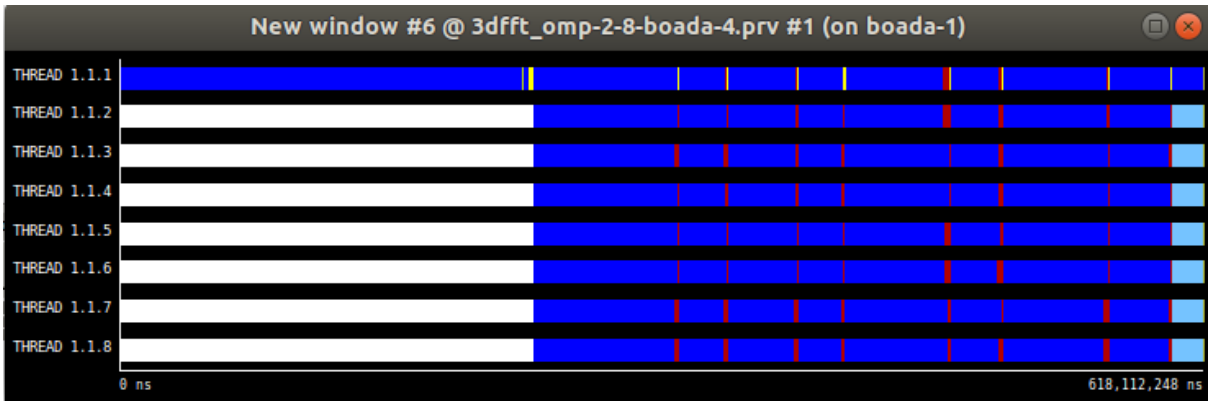
En este apartado, hemos aumentado la parte paralelizable lo que hace que la parte no paralelizable sea casi nula pero al ser la granularidad tan alta el programa tiene que dividir las tareas lo que genera un coste (overhead). Este overhead a partir de ciertos threads no mejora porque la creación de toda estas tareas requiere un coste bastante significativo de tiempo.

Versión final reduciendo el overhead:

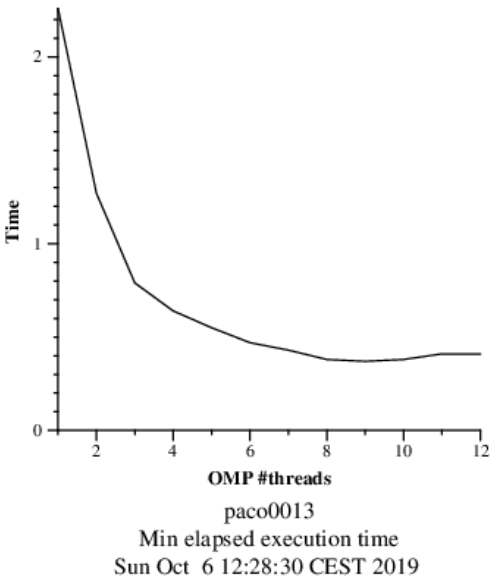
wxparaver con 1 thread:

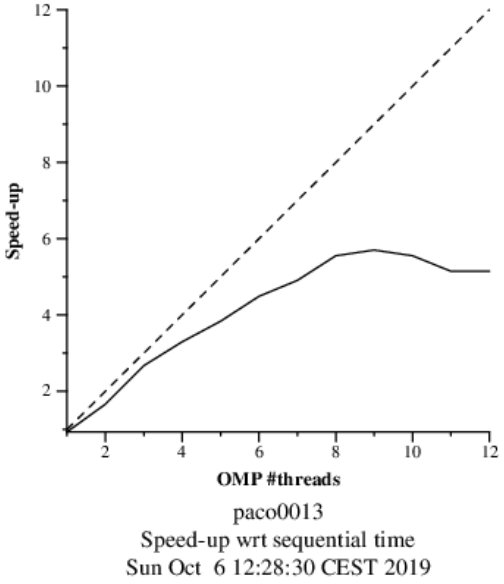


wxparaver con 8 thread:



Strong scalability





Observaciones

En este apartado, hemos disminuido la granularidad lo que causa que se creen menos tareas y así disminuya el overhead.