Experimental setup and tools

Ismael Douha Prieto, Eloi Cruz Harillo - Grup 1305

Q1 2019/20

1 Introduction

Durant aquesta pràctica hem treballat amb les diferents eines que proporciona el sistema per tal de familiaritzar-me amb l'entorn de treball de cara a les futures entregues. Amb aquestes eines hem pogut descobrir les característiques del hardware emprat, entendre les escalabilitats del codi o com descomposar el codi per obtenir millors paral·lelismes, entre d'altres.

2 Node architecture and memory

Primer de tot he fet les pertinents comprovacions del hardware a emprar durant les practiques. Aquest està composat per 8 nodes diferents amb 3 arquitectures diferents (1-4, 5 i 6-8). Tots presenten similituts en certs aspectes, malgrat que en la majoria presenten diferencies fent-lo més apropiats per a un treballs que no pas uns altres.

	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	2395Mhz	2600Mhz	1700Mhz
L1-I cache size (per-core)	32KB	32KB	32KB
L1-D cache size (per-core)	32KB	32KB	32KB
L2 cache size (per-core)	256KB	256KB	256KB
Last-level cache size (per-socket)	12MB	15MB	20MB
Main memory size (per socket)	12GB	32GB	16GB
Main memory size (per node)	24GB	64GB	32GB

Table 1: Informació rellevant de l'arquitectura i la memòria dels diferents nodes.

Gracies a l'eina "lstopo" amb els pertinents flags, obtenim un esquema de l'arquitectura que ens dona una visió més adient de com estan estructurats els nodes.

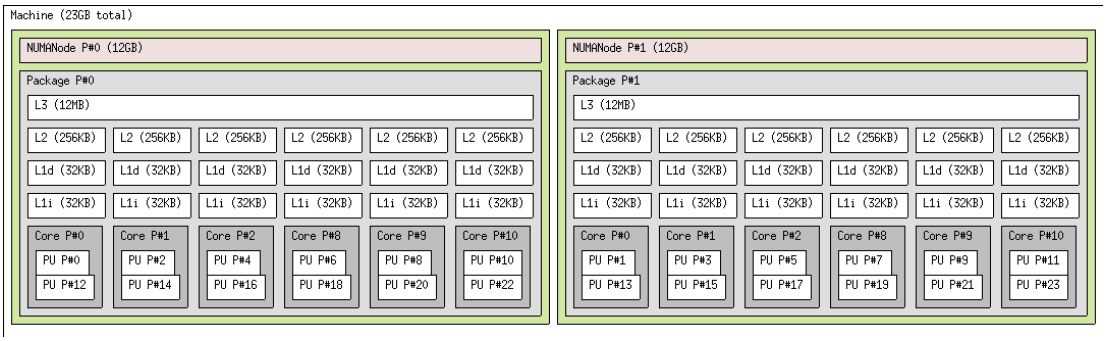


Figure 1: Arquitectura del node Boada-1.

3 Strong vs. weak scalability

Un cop ja coneixem com es el sistema amb el qual treballarem, cal realitzar unes certes proves per saber l'escalabilitat que ens proporciona.

Començant per la "strong scalability", la qual variant el nombre de threads en un problema de mida fixa es comprova si el temps d'execució varia de forma proporcional. Si observem la figura 2 podem veure que ens proporciona una bona escalabilitat.

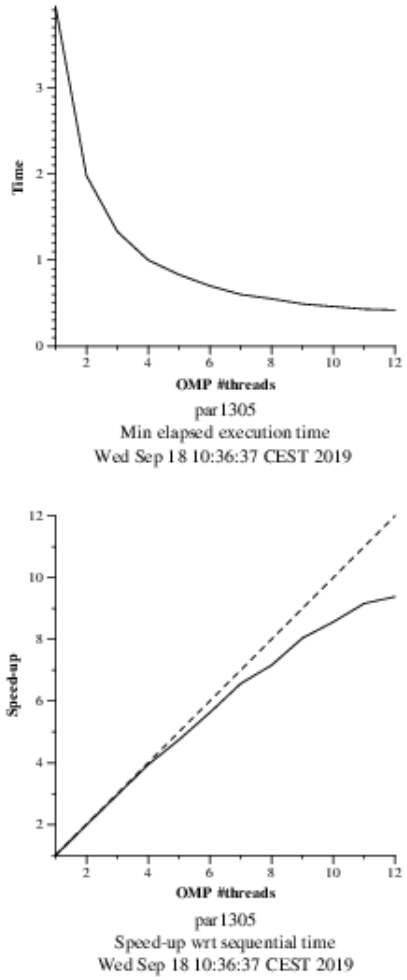


Figure 2: Grafica del speed up i execution time al node boada-2.

Per a la "weak scalability", on la mida del problema es proporcional al nombre de threads, també s'han fet les pertinents proves i trobem que presenta una bona eficiencia.

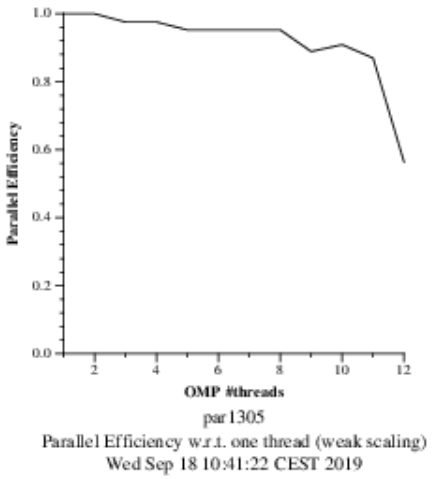


Figure 3: Grafica de la eficiencia del paral·lelisme al node boada-2.

Com a conclusió de l'escalabilitat trobem que son sistemes amb una bona escalabilitat, malgrat que no són perfectes (es normal degut als overheads i demes).

4 Analysis of task decompositions for 3DFFT

Després de coneixer com funciona el sistema a emprar, passem a treballar la descomposició de tasques, les quals ens permeten crear "paquets" de treball els quals podem distribuïr entre els diferents nodes, permeten la paral·lelització del programa.

A més de la taula 2 on es pot apreciar la millora de rendiment quan augmenta el paralelisme, per a les versions 4 i 5 podem apreciar a les grafiques la millora del rendiment en funció dels procesadors emprats. Més concretament, a la versió 5, la millora aplicada es realitzada als bucles del plans xy i zx, on generant aquesta granularitat més petita obtenim més paral·lelisme.

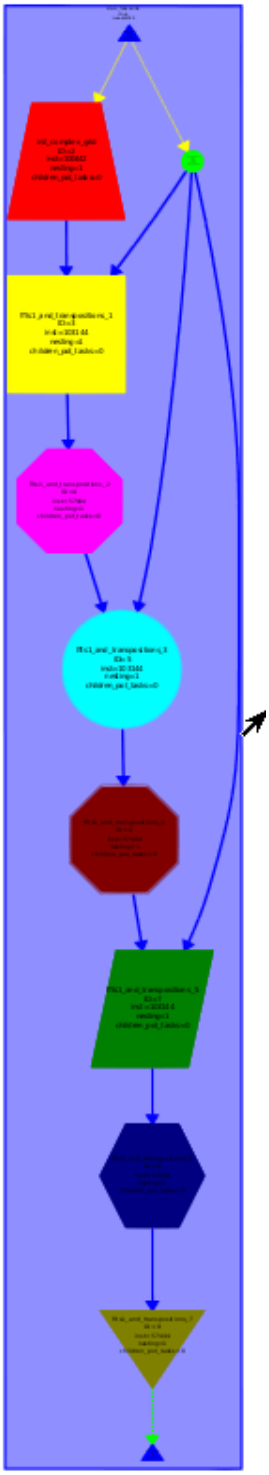


Figure 4: Descomposició de tasques de la versió 1 del programa.

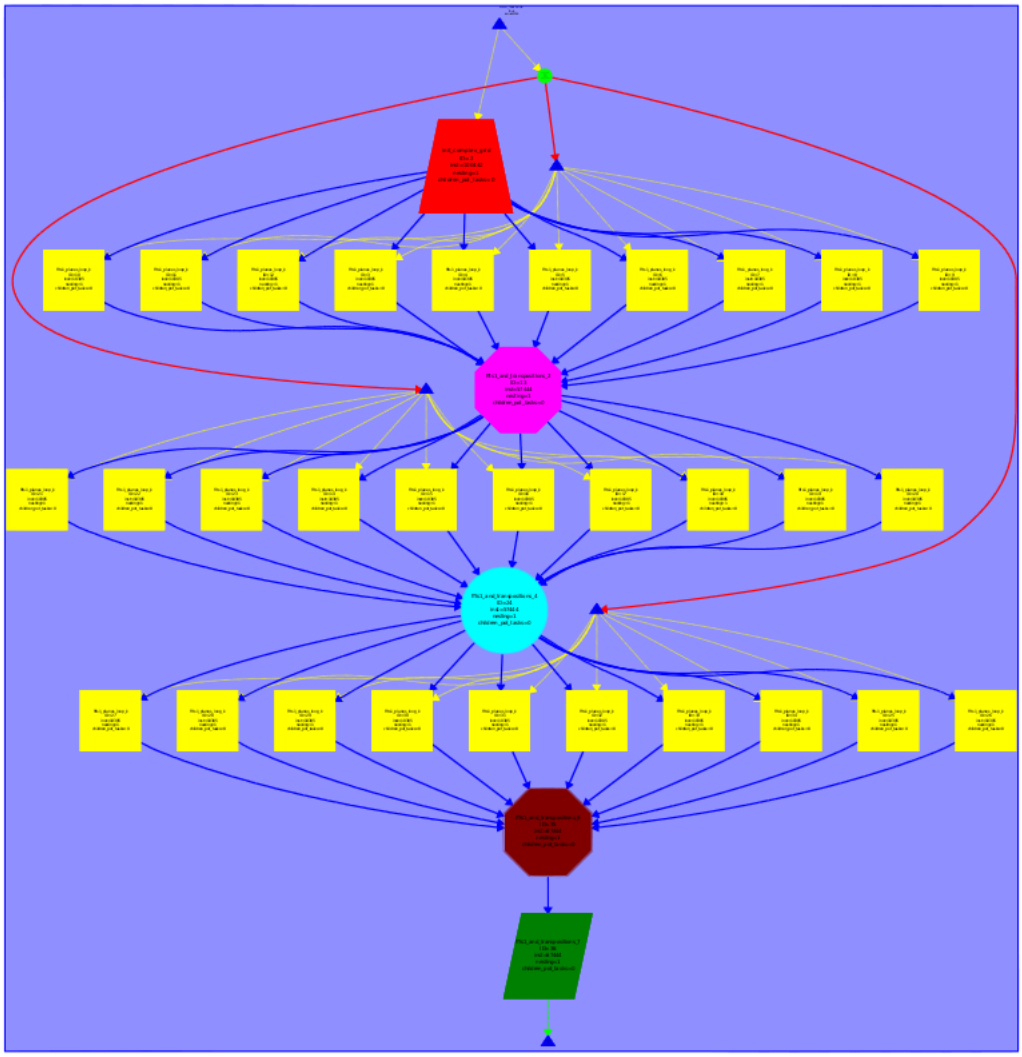


Figure 5: Descomposició de tasques de la versió 2 del programa.

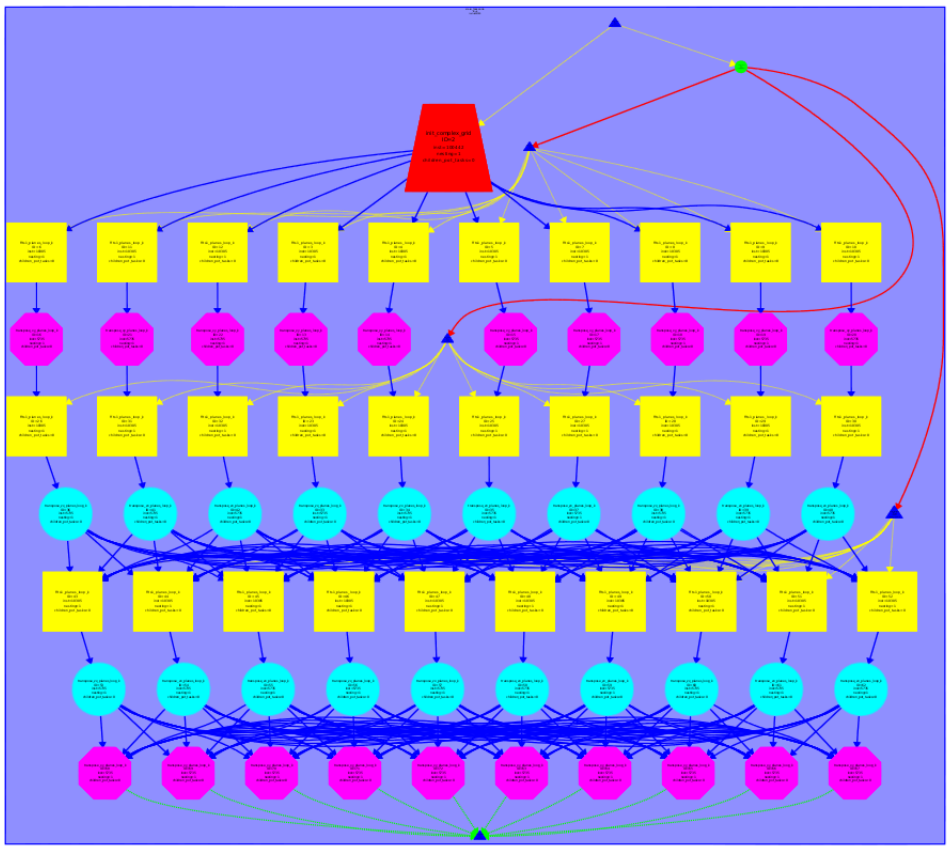


Figure 6: Descomposició de tasques de la versió 3 del programa.

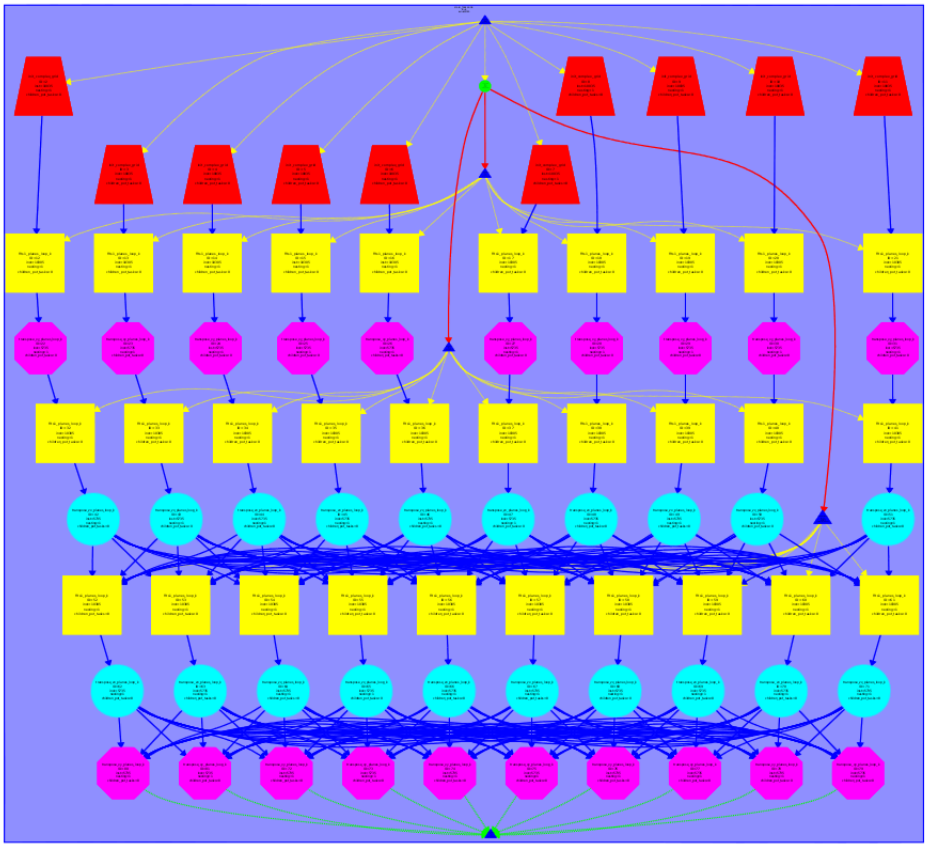


Figure 7: Descomposició de tasques de la versió 4 del programa.

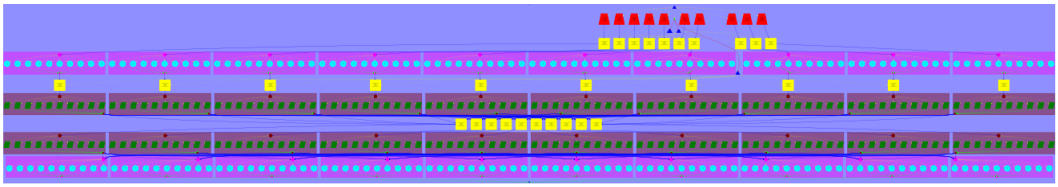


Figure 8: Descomposició de tasques de la versió 5 del programa.

Version	T_1	T_∞	Parallelism
seq	639,707 us	639,707 us	1
v1	639,780 us	639,707 us	1
v2	639,780 us	361,190 us	1,77
v3	639,780 us	154,354 us	4,14
v4	639,780 us	64,018 us	10
v5	639.780	45.650	14

Table 2: Taula del possibles paral·lelismes que es poden obtenir amb les diferents versions.

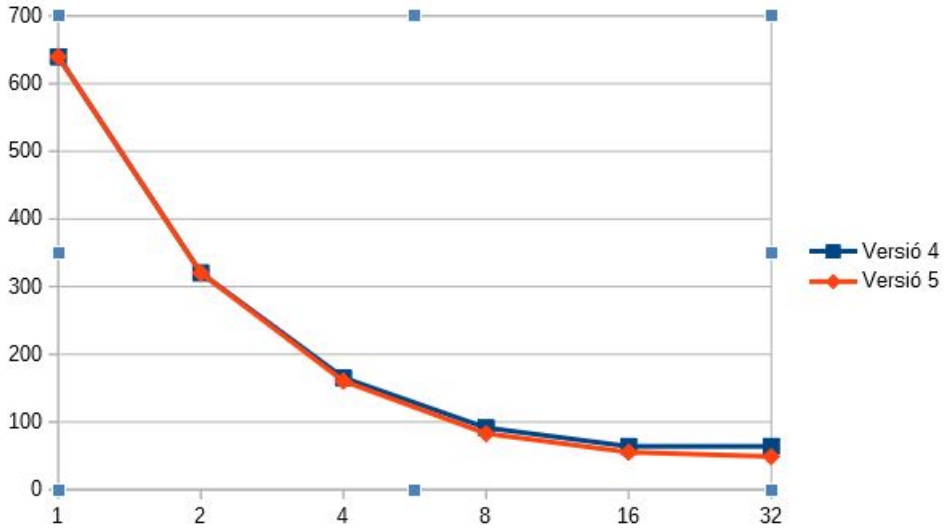


Figure 9: Gràfica de la "strong scalability de les versions 4 i 5 del programa.

5 Understanding the parallel execution of 3DFFT

En aquest apartat analitzem l'execució de tres versions del programa 3DFFT. A la Taula 3 adjuntada al final del document està el resum dels temps d'execució, paral·lelisme i speedup obtinguts en les diverses execucions. Seguidament analitzarem els resultats obtinguts.

5.1 Initial version 3DFFT

En primer lloc hem de calcular el paral·lelisme del programa. Per calcular-lo hem procedit utilitzant la següent fórmula.

$$\phi = Tpar/(Tseq + Tpar)$$

En aquesta fórmula trobem dos variables. La primera Tpar que correspon amb la part paral·lela del codi. En segon lloc trobem Tseq que correspon amb la part seqüencial del codi. Per poder obtenir aquests valors hem utilitzat la versió instrumentada del script submit i per això els temps d'execució són una mica superiors als reals. Com es pot veure també hem calculat speedup que patiria el programa si disposes del nombre màxim de recursos necessaris. Per fer-ho suposem que el temps d'execució de la part paral·lela de codi tendeix a 0 i per tant nomes tenim en compte la part seqüencial del codi.

$$Tpar = 1593441\mu s = 1.593s(instrumentada)$$

$$Tseq = 198975{+}578758{+}13{+}16{+}14{+}17{+}13{+}14{+}17060 = 794880\mu s = 0.795s(instrumentada)$$

$$T1 = Tpar + Tseq = 2388321\mu s = 2.388s(instrumentada)$$

$$\phi = Tpar/(Tseq + Tpar) = 1.593/(0.795 + 1.593) = 0.667$$

$$S\infty = T1/Tseq = 2.388/0.795 = 3$$

Procediment obtenció Tpar i Tseq: per obtenir la informació del temps d'execució del codi hem fet servir la següent comada:
`qsub -l execution submit-omp-i.sh 3dfft-omp.c 1`
Amb el software paraver hem carregat la traça generada (fitxer amb l'extensió ".prv"). Fent això hem obtingut la informació que es mostra a les imatges 10, 11 i 12.

La part seqüencial del codi es mostra a les imatges 10 de color blau. Està format per diverses àrees. La primera correspon a la inicialització del codi i dura fins al primer fork. La última correspon a la finalització del codi a partir de l'últim join. Finalment trobem totes les seqüencies del codi entre el join de una tasca i el fork de la següent tasca. Aquests petits intervals de temps es veuen reflectits a la figura número 12.

El temps d'execució de la part paral·lela correspon a la suma del temps d'execució de les les 7 tasques creades pel programa i estan representades de color vermell a la imatge 10. El color vermell no ha de portar a confusió. En aquest cas el paraver ens pinta les zones de color vermell perquè la granularitat de les tasques es massa petita i hi ha molt temps de sincronització entre les diferents execucions de codi perquè les tasques són molt dependents entre si (accedeixen a mateixes posicions de memòria).

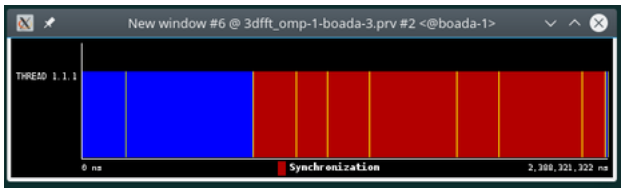
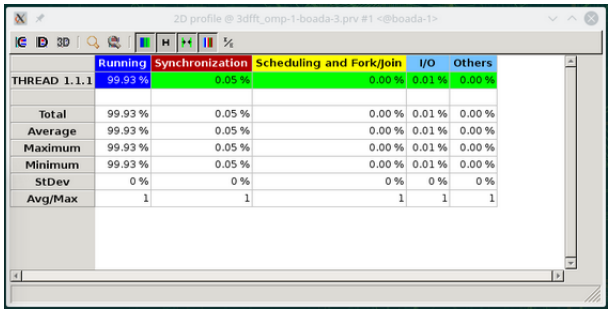


Figure 10: Timeline 3dfft sense modificacions en un processador.



	Running	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	99.93 %	0.05 %	0.00 %	0.01 %	0.00 %
Total	99.93 %	0.05 %	0.00 %	0.01 %	0.00 %
Average	99.93 %	0.05 %	0.00 %	0.01 %	0.00 %
Maximum	99.93 %	0.05 %	0.00 %	0.01 %	0.00 %
Minimum	99.93 %	0.05 %	0.00 %	0.01 %	0.00 %
StDev	0 %	0 %	0 %	0 %	0 %
Avg/Max	1	1	1	1	1

Figure 11: Informació descomposada 3dfft sense modificacions en un processador.

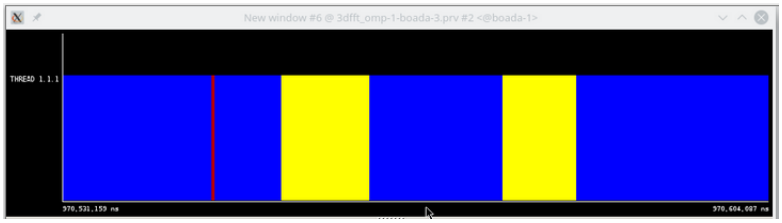


Figure 12: Zoom timeline 3dfft join-fork.

En segon lloc hem calculat el temps d'execució del programa en 8 processadors i hem calculat l'speedup respecte l'execució en un sol processador. Per fer-ho hem procedit com al apartat anterior. Hem executat la següent comanda:
`qsub -l execution submit-omp-i.sh 3dfft-omp.c 8`
Fent això hem executat la versió instrumentada del script submit per poder veure al paraver la informació desitjada. La imatge 13 mostra el cronograma de l'execució amb 8 processadors de 3dfft-omp. Com podem observar les parts que hem definit com a paral·leles en l'apartat anterior s'executen de forma paral·lela en els 8 processadors disponibles. A la imatge 14 observem que el temps de sincronització i el temps d'scheduling agafen un pes important. Per tant les dos mètriques obtingudes són les següents:

$$T8 = 1733764\mu s = 1.733s(instrumentada)$$

$$S8 = T1/T8 = 2.388/1.733 = 1.377$$

Seguidament hem procedit a obtenir el temps d'execució real del programa ja que en els apartats anteriors la instrumentació afegeix un overhead en el temps d'execució. Per obtenir aquestes dades hem executat les següents comandes:
`qsub -l execution submit-omp.sh 3dfft-omp.c 1`
`qsub -l execution submit-omp.sh 3dfft-omp.c 8`
Aquesta programa treu un output anomenat "3dfft-omp-8-boada-2.txt" on s'indica entre d'altres camps el temps d'execució del programa. Com podem veure el temps d'execució es redueix perquè eliminem el temps d'instrumentació del programa.

$$T1 = 1746370\mu s = 1.746s$$

$$T8 = 546592\mu s = 0.547s$$

$$S8 = T1/T8 = 1.746/0.547 = 3.195$$

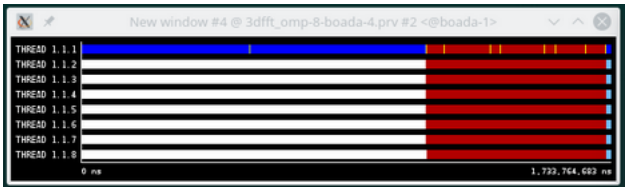
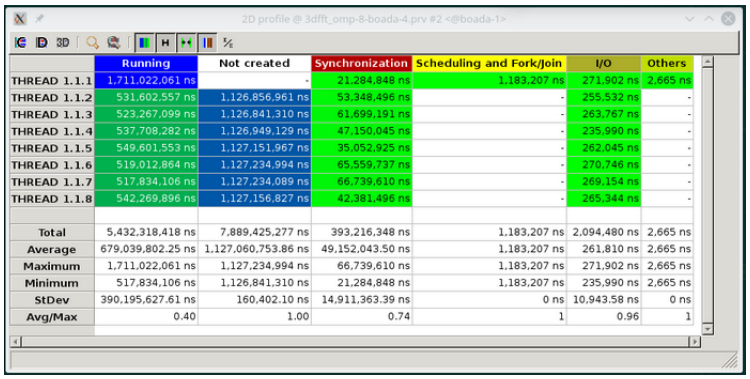


Figure 13: Timeline 3dfft sense modificacions en 8 processadors.



	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	1,711,022,061 ns	-	21,284,848 ns	1,183,207 ns	271,902 ns	2,665 ns
THREAD 1.1.2	531,602,557 ns	1,126,856,961 ns	53,348,496 ns	-	255,532 ns	-
THREAD 1.1.3	523,267,099 ns	1,126,841,310 ns	61,699,191 ns	-	263,767 ns	-
THREAD 1.1.4	537,708,282 ns	1,126,949,129 ns	47,150,045 ns	-	235,990 ns	-
THREAD 1.1.5	549,601,553 ns	1,127,151,967 ns	35,052,925 ns	-	262,045 ns	-
THREAD 1.1.6	519,012,864 ns	1,127,234,994 ns	65,559,737 ns	-	270,746 ns	-
THREAD 1.1.7	517,834,106 ns	1,127,234,089 ns	66,739,610 ns	-	269,154 ns	-
THREAD 1.1.8	542,269,896 ns	1,127,156,827 ns	42,381,496 ns	-	265,344 ns	-
Total	5,432,318,418 ns	7,889,425,277 ns	393,216,348 ns	1,183,207 ns	2,094,480 ns	2,665 ns
Average	679,039,802.25 ns	1,127,060,753.86 ns	49,152,043.50 ns	1,183,207 ns	261,810 ns	2,665 ns
Maximum	1,711,022,061 ns	1,127,234,994 ns	66,739,610 ns	1,183,207 ns	271,902 ns	2,665 ns
Minimum	517,834,106 ns	1,126,841,310 ns	21,284,848 ns	1,183,207 ns	235,990 ns	2,665 ns
StDev	390,195,627.61 ns	160,402.10 ns	14,911,363.39 ns	0 ns	10,943.58 ns	0 ns
Avg/Max	0.40	1.00	0.74	1	0.96	1

Figure 14: Informació descomposada 3dfft sense modificacions en 8 processadors.

La figura 15 mostra el plot de strong scalability. Per obtenirlo hem executat les següents comandes:
`qsub -l execution submit-strong-omp.sh`

```
ps2pdf 3dfft-omp-1-12-3-strong-boada-2.ps
xpdf 3dfft-pcfomp-1-12-3-strong-boada-2.pdf
```

Les gràfiques de la figura 15 mostren la strong scalability del programa. Això significa que ens mostren com es redueix el temps d'execució a mida que augmentem els recursos dels que disposa la màquina on s'executa el codi. Podem veure que l'speedup augmenta en cada execució fins que arribem a 6 processadors obtenint un speed-up de aproximadament 2. Aquest valor s'allunya del valor Sinf = 3 que representa l'speedup del programa en el cas ideal en que el temps d'execució de la part paral·lela del codi tendeix a 0 si es disposa d'un número de recursos infinit. Això pot ser degut a que hi ha molta dependència entre les tasques creades i per tant hi ha molt overhead.

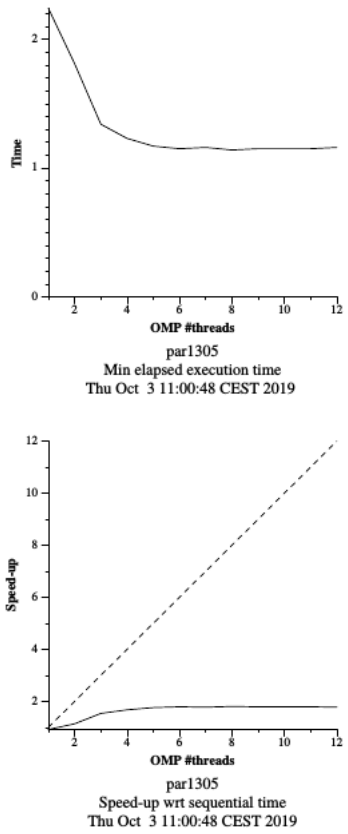


Figure 15: Strong scalability plot.

5.2 Improving parallelism

En l'apartat anterior la inicialització de la matriu es fa de forma seqüencial. Aquest fet fa que perdem paral·lelisme. En aquest apartat fem que la creació de la matriu també es faci de forma paral·lela i això donarà lloc a que els temps d'execució variïn en tots els casos. Per fer-ho anirem a la funció `init_complex_gridi` descomposarem el segon bucle en tasques individuals. Seguidament indicarem els nous valors dels temps i adjuntarem les captures de pantalla del Paraver que mostren els resultats.

$$Tpar = 2101890\mu s = 2.101s(instrumentada)$$

$$Tseq = 208764{+}23{+}17{+}15{+}17{+}17{+}12{+}15{+}18417 = 227297\mu s = 0.227s(instrumentada)$$

$$T1 = Tpar + Tseq = 2329187\mu s = 2.329s(instrumentada)$$

$$\phi = Tpar/(Tseq + Tpar) = 2.101/2.329 = 0.902$$

$$T8 = 979394\mu s = 0.979s(instrumentada)$$

$$S8 = T1/T8 = 2.329/0.979 = 2.379$$

$$S\infty = T1/Tseq = 2.329/0.227 = 10.26$$

La diferència més important d'aquesta execució es pot veure en la figura 16. Com podem veure la part seqüencial que correspon amb la inicialització de codi és molt més petita que en l'apartat anterior. Això es veu reflectit directament en el paral·lelisme del programa que augmenta 0.3.m Com que molta més part del codi pot ser executada de forma paral·lela l'speedup quan executem el programa en 8 processadors es superior al de l'apartat anterior. Degut a aquest augment del paral·lelisme la part seqüencial del codi és inferior i per això l'speedup del programa si s'executes amb els màxims recursos augmenta dràsticament.

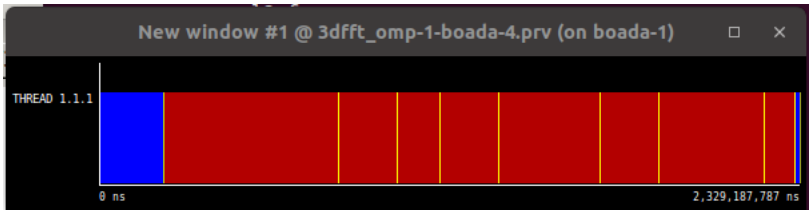
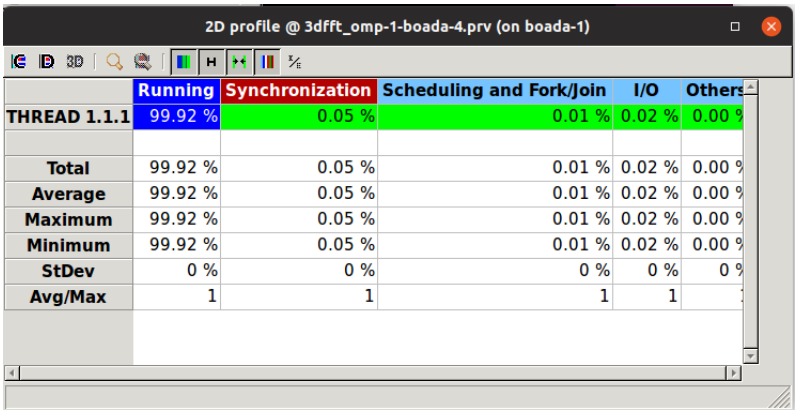


Figure 16: Timeline 3dfft paral·lelisme millorat en un processador.



	Running	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	99.92 %	0.05 %	0.01 %	0.02 %	0.00 %
Total	99.92 %	0.05 %	0.01 %	0.02 %	0.00 %
Average	99.92 %	0.05 %	0.01 %	0.02 %	0.00 %
Maximum	99.92 %	0.05 %	0.01 %	0.02 %	0.00 %
Minimum	99.92 %	0.05 %	0.01 %	0.02 %	0.00 %
StDev	0 %	0 %	0 %	0 %	0 %
Avg/Max	1	1	1	1	1

Figure 17: Informació descomposada 3dfft paral·lelisme millorat en un processador.

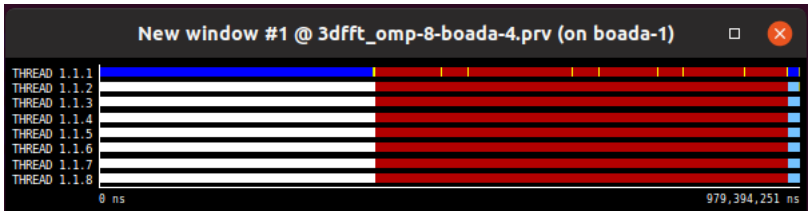
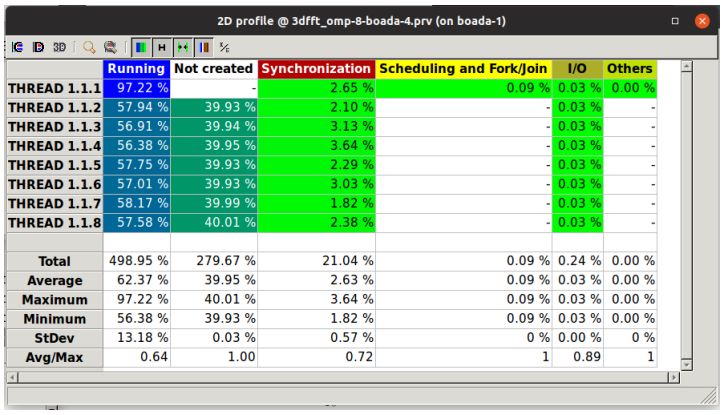


Figure 18: Timeline 3dfft paral·lelisme millorat en 8 processadors.



	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	97.22 %	-	2.65 %	0.09 %	0.03 %	0.00 %
THREAD 1.1.2	57.94 %	39.93 %	2.10 %	-	0.03 %	-
THREAD 1.1.3	56.91 %	39.94 %	3.13 %	-	0.03 %	-
THREAD 1.1.4	56.38 %	39.95 %	3.64 %	-	0.03 %	-
THREAD 1.1.5	57.75 %	39.93 %	2.29 %	-	0.03 %	-
THREAD 1.1.6	57.01 %	39.93 %	3.03 %	-	0.03 %	-
THREAD 1.1.7	58.17 %	39.99 %	1.82 %	-	0.03 %	-
THREAD 1.1.8	57.58 %	40.01 %	2.38 %	-	0.03 %	-
Total	498.95 %	279.67 %	21.04 %	0.09 %	0.24 %	0.00 %
Average	62.37 %	39.95 %	2.63 %	0.09 %	0.03 %	0.00 %
Maximum	97.22 %	40.01 %	3.64 %	0.09 %	0.03 %	0.00 %
Minimum	56.38 %	39.93 %	1.82 %	0.09 %	0.03 %	0.00 %
StDev	13.18 %	0.03 %	0.57 %	0 %	0.00 %	0 %
Avg/Max	0.64	1.00	0.72	1	0.89	1

Figure 19: Informació descomposada 3dfft paral·lelisme millorat en 8 processadors.

Les gràfiques de la figura 20 mostren la strong scalability del programa. Això

significa que ens mostren com es redueix el temps d'execució a mida que augmentem els recursos dels que disposa la màquina on s'executa el codi. Podem veure que l'speedup augmenta en cada execució fins que arribem a 8 processadors obtenint un speed-up de aproximadament 3.5. Aquest valor s'allunya molt del valor Sinf = 10.2 que representa l'speedup del programa en el cas ideal en que el temps d'execució de la part paral·lela del codi tendeix a 0 si es disposa d'un número de recursos infinit. Això pot ser degut a que hi ha molta dependència entre les tasques creades i per tant hi ha molt overhead.

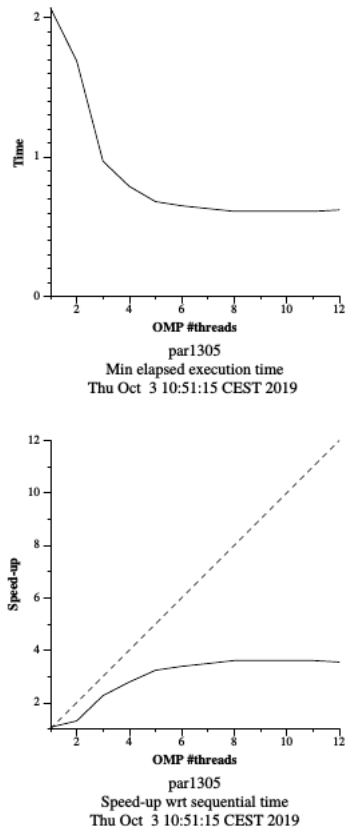


Figure 20: Strong scalability plot.

5.3 Reducing parallelisation overheads

A l'apartat anterior per millorar el temps d'execució hem procedit augmentat el paral·lelisme del programa. Això vol dir que hem buscat una zona del codi que podia ser executada de forma paral·lela i l'hem programat. En aquest apartat canviarem l'estratègia per millorar el temps d'execució. El que farem serà reduir el overhead creat en la sincronització de les diferents tasques. Per

fer-ho reduirem la quantitat de tasques creades (augmentant la granularitat de cadascuna d'elles). Aquest augment de granularitat el farem canviant la creació de tasques del segon bucle de totes les funcions de tractament de la matriu al primer bucle. Els resultats obtinguts són els següents.

$$Tpar = 2144494\mu s = 2.144s(instrumentada)$$

$$Tseq = 207386{+}23{+}16{+}15{+}17{+}13{+}12{+}18{+}17688 = 225184\mu s = 0.225s(instrumentada)$$

$$T1 = Tpar + Tseq = 2369678\mu s = 2.369s(instrumentada)$$

$$\phi = Tpar/(Tseq + Tpar) = 2.144/2.369 = 0.905$$

$$T8 = 638272\mu s = 0.638s(instrumentada)$$

$$S8 = T1/T8 = 2.369/0.638 = 3.71$$

$$S\infty = T1/Tseq = 2.369/0.225 = 10.53$$

Com es pot veure en les dades donades el paral·lelisme i conseqüentment el speedup en infinits processadors són pràcticament iguals que en el apartat anterior. Tanmateix quan executem el programa en 8 processadors el temps d'execució de la part paral·lela del codi es redueix ja que les tasques que creem ara són molt més independents que en l'apartat anterior i això redueix el temps de sincronització. A la figura 21 ara el temps de la zona paral·lela ja no es de color vermell sinó que es de color blau el que indica temps real d'execució. A la figura 24 podem observar que el temps de sincronització ocupa un 0.13% en canvia a la figura 19 de l'apartat anterior ocupava un 2.65% del temps total.

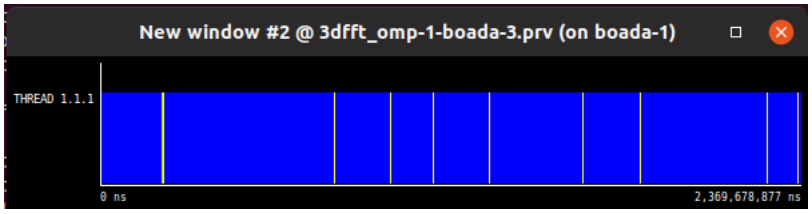
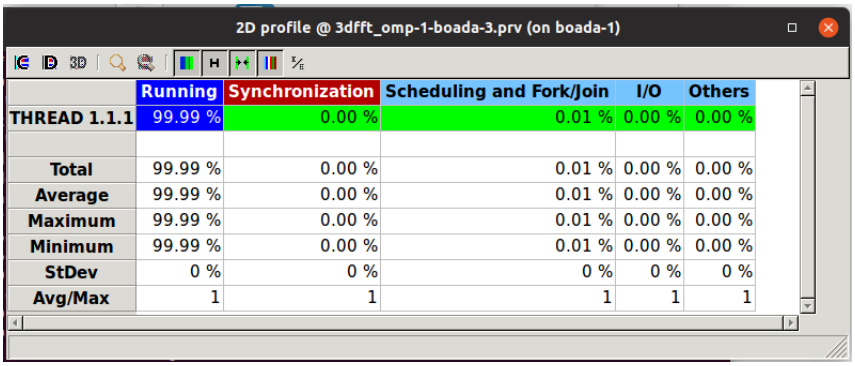


Figure 21: Timeline 3dfft granuralitat augmentada en un processador.



	Running	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	99.99 %	0.00 %	0.01 %	0.00 %	0.00 %
Total	99.99 %	0.00 %	0.01 %	0.00 %	0.00 %
Average	99.99 %	0.00 %	0.01 %	0.00 %	0.00 %
Maximum	99.99 %	0.00 %	0.01 %	0.00 %	0.00 %
Minimum	99.99 %	0.00 %	0.01 %	0.00 %	0.00 %
StDev	0 %	0 %	0 %	0 %	0 %
Avg/Max	1	1	1	1	1

Figure 22: Informació descomposada 3dfft granularitat augmentada en un processador.

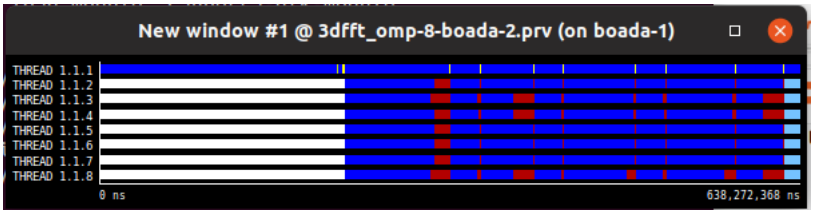
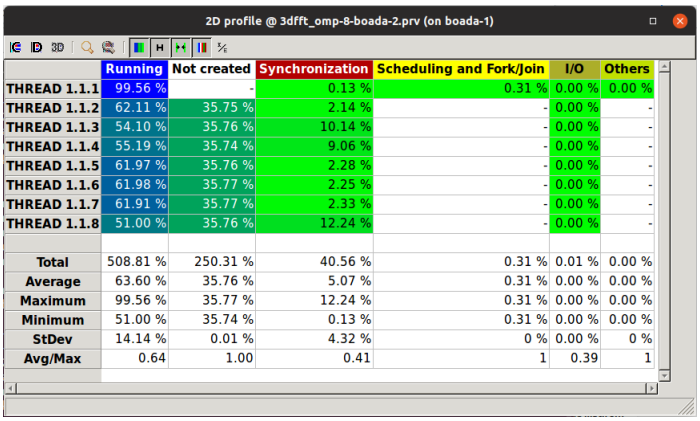


Figure 23: Timeline 3dfft granularitat augmentada en 8 processadors.



	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	99.56 %	-	0.13 %	0.31 %	0.00 %	0.00 %
THREAD 1.1.2	62.11 %	35.75 %	2.14 %	-	0.00 %	-
THREAD 1.1.3	54.10 %	35.76 %	10.14 %	-	0.00 %	-
THREAD 1.1.4	55.19 %	35.74 %	9.06 %	-	0.00 %	-
THREAD 1.1.5	61.97 %	35.76 %	2.28 %	-	0.00 %	-
THREAD 1.1.6	61.98 %	35.77 %	2.25 %	-	0.00 %	-
THREAD 1.1.7	61.91 %	35.77 %	2.33 %	-	0.00 %	-
THREAD 1.1.8	51.00 %	35.76 %	12.24 %	-	0.00 %	-
Total	508.81 %	250.31 %	40.56 %	0.31 %	0.01 %	0.00 %
Average	63.60 %	35.76 %	5.07 %	0.31 %	0.00 %	0.00 %
Maximum	99.56 %	35.77 %	12.24 %	0.31 %	0.00 %	0.00 %
Minimum	51.00 %	35.74 %	0.13 %	0.31 %	0.00 %	0.00 %
StDev	14.14 %	0.01 %	4.32 %	0 %	0.00 %	0 %
Avg/Max	0.64	1.00	0.41	1	0.39	1

Figure 24: Informació descomposada 3dfft granularitat augmentada en 8 processadors.

Les gràfiques de la figura 25 mostren la strong scalability del programa. Això significa que ens mostren com es redueix el temps d'execució a mida que augmentem els recursos dels que disposa la màquina on s'executa el codi. Podem

veure que l'speedup augmenta en cada execució fins que arribem a 10 processadors obtenint un speed-up de aproximadament 6. Aquest valor s'allunya molt del valor Sinf = 10.5 que representa l'speedup del programa en el cas ideal en que el temps d'execució de la part paral·lela del codi tendeix a 0 si es disposa d'un número de recursos infinit. Això pot ser degut a que la granuralitat de les tasques creades és massa gran.

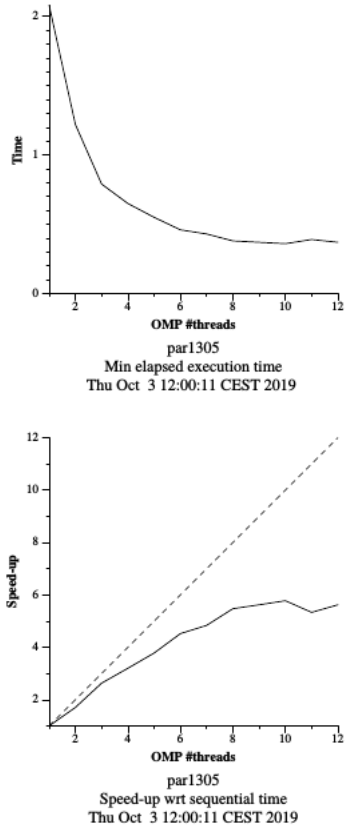


Figure 25: Strong scalability plot.

Version	ϕ	S_∞	T_1	T_8	S_8
initial version in 3dfft_omp.c	0.667	3	2.388s	1.733s	1.378
new version with improved ϕ	0.902	10.26	2.329s	0.979s	2.379
final version with reduced parallelisation overheads	0.905	10.53	2.369s	0.638s	3.71

Table 3: Taula amb els valors de les execucions INSTRUMENTADES de les tres versions del codi.

En conclusió resumirem les dues millores possibles que s'ens han presentat a la pràctica per millorar el temps d'execució d'un programa en diversos threads. En primer lloc millorant el paral·lelisme (trobant seccions del codi que poden ser executades de forma paral·lela). En segon lloc reduint el temps de sincronització entre tasques quan aquestes tenen dependència entre elles (per exemple accedint a les mateixes posicions de memòria). En aquesta pràctica per aconseguir-ho hem reduint el nombre de tasques generades augmentant així la seva granularitat.