David Martos
Mariona Farré
Grupo: paco1208
Curso: 2022-23

PACO:
Laboratorio 1

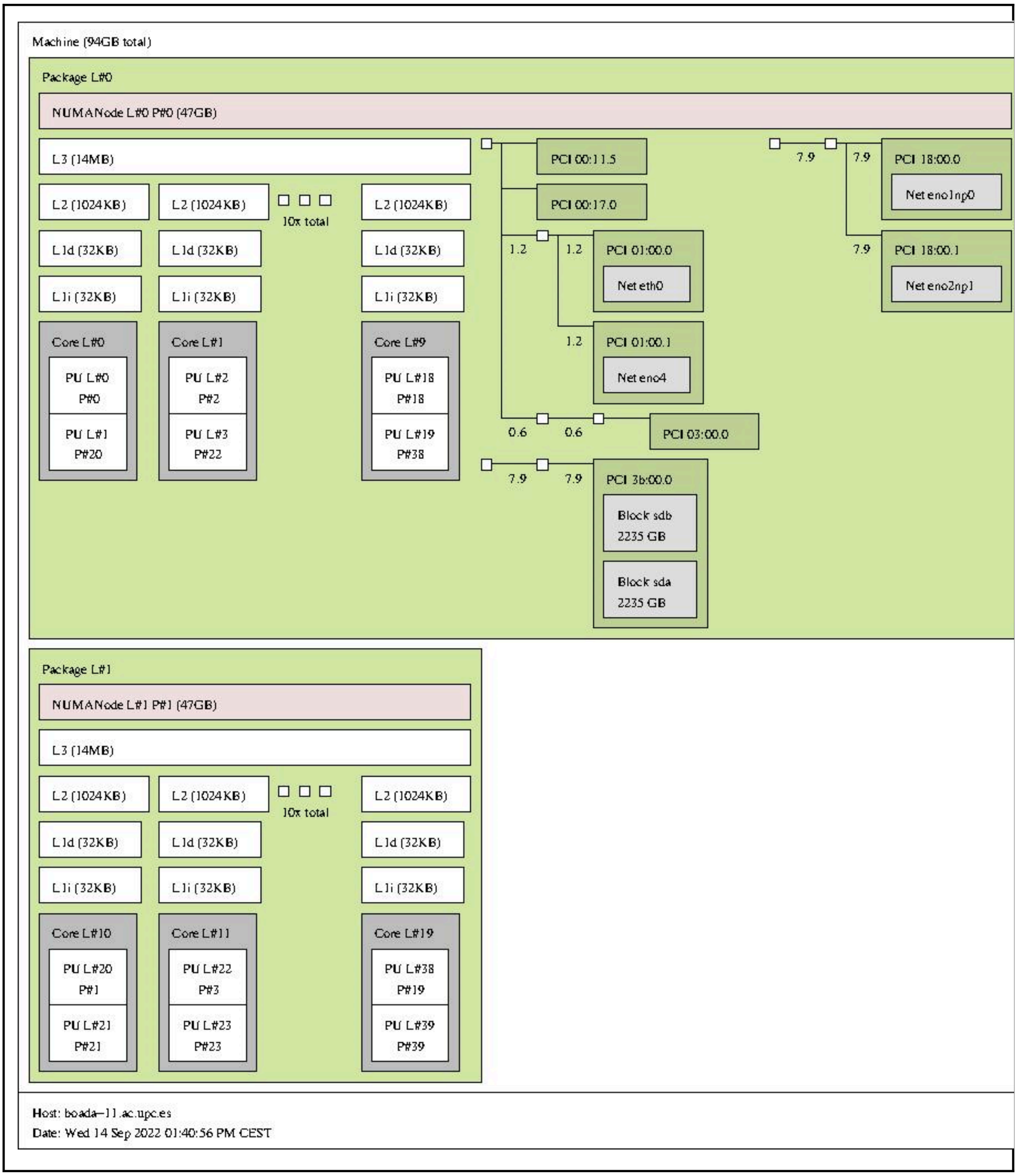
Setup Experimental:

- **Arquitectura de nodos y memoria:**

For the deliverable: Based on the lscpu and lstopo results and .fig image, we ask you to fill in the table of the deliverable in your lab report indicating which are the number of sockets, cores per socket and threads per core in a specific node; the amount of main memory in a specific node, and each NUMAnode; and the cache memory hierarchy (L1, L2 and L3), private or shared to each core/socket.

Include in the document the script file that you have used to obtain the values in this table. Also include in the description the architectural diagram for one of the nodes boada-11 to boada-14 as obtained when using the lstopo command. Appropriately comment whatever you consider appropriate.

	Any of the nodes among boada-11 to boada-14
Number of sockets per node	2
Number of cores per socket	8
Number of threads per core	2
Maximum core frequency	1300 MHz
L1-I cache size (per-core)	512KB
L1-D cache size (per-core)	512KB
L2 cache size (per-core)	4MB
Last-level cache size (per-socket)	40MB
Main memory size (per socket)	16GB
Main memory size (per node)	8GB


Machine (94GB total)
Package L#0
NUMANode L#0 P#0 (47GB)
L3 (14MB)
L2 (1024KB)
L2 (1024KB)
10x total
L2 (1024KB)
L1d (32KB)
L1d (32KB)
L1d (32KB)
L1i (32KB)
L1i (32KB)
L1i (32KB)
Core L#0
PU L#0
P#0
PU L#1
P#20
Core L#1
PU L#2
P#2
PU L#3
P#22
Core L#9
PU L#18
P#18
PU L#19
P#38
PCI 00:11.5
PCI 00:17.0
1.2
1.2
PCI 01:00.0
Net eth0
1.2
PCI 01:00.1
Net eno4
0.6
0.6
PCI 03:00.0
7.9
7.9
PCI 3b:00.0
Block sdb
2235 GB
Block sda
2235 GB
7.9
7.9
PCI 18:00.0
Net eno1np0
7.9
PCI 18:00.1
Net eno2np1
Package L#1
NUMANode L#1 P#1 (47GB)
L3 (14MB)
L2 (1024KB)
L2 (1024KB)
10x total
L2 (1024KB)
L1d (32KB)
L1d (32KB)
L1d (32KB)
L1i (32KB)
L1i (32KB)
L1i (32KB)
Core L#10
PU L#20
P#1
PU L#21
P#21
Core L#11
PU L#22
P#3
PU L#23
P#23
Core L#19
PU L#38
P#19
PU L#39
P#39
Host: boada-11.ac.upc.es
Date: Wed 14 Sep 2022 01:40:56 PM CEST

- **Compilación y ejecución de programas OpenMP**

For the deliverable: Draw a table in your deliverable showing the user and system CPU time, the elapsed time, and the % of CPU used in the two scenarios (interactive and queued) and with the number of threads enumerated before. Do you observe a major difference between the interactive and queued execution? Include the table in the Deliverable commenting the results observed

La mayor diferencia entre la ejecución interactive versus la queued, es que la ejecución interactive necesita más tiempo de ejecución y no usa procentaje muy elevado de la CPU, que comparado con ejecución a base de queued, termina mucho más rápido pero usa un elevado porcentaje de la CPU.

Fill in table below to show the user and system CPU time, the elapsed time, and the % of CPU used in these two scenarios and with 1, 4, 8 and 12 threads. Reason the differences you observe. In addition, for the case of the queued scenario, explain what strong (both up to 20 and 40 threads) and weak scalability refer to, exemplifying your explanation with the execution time and speed–up plots that you obtained for pi omp.c.

#threads	**Timing information Interactive**				**Timing information Queued**			
	user	**system**	**elapsed**	**% of CPU**	**user**	**system**	**elapsed**	**% of CPU**
1	2.36	0.0	0:02.37	99%	0.69	0.00	0:00.70	98%
4	2.37	0.01	0:01.19	199%	0.70	0.00	0:00.19	357%
8	2.40	0.04	0:01.23	199%	0.74	0.00	0:00.11	646%
16	2.44	0.11	0:01.28	199%	0.79	0.00	0.00.07	1121%
20	2.46	0.10	0:01.29	199%	0.83	0.00	0:00.06	1330%

En esta tabla podemos observar, que los tiempos de una ejecución Interactive ronda al minuto y usan el porcentaje de la CPU de un rango entre el 99 y 199% mientras que la ejecución queued, el tiempo se recorta a menos de un minuto, solo transcurriendo de mediana 10 segundos (menos la de 1 thread) pero utilizando un gran valor de porcentaje de la CPU de un rango entre el 98 al 1330%.

- **Escalabilidad Fuerte versus Débil**

For the deliverable: Include all the scalability plots generated in the Deliverable reasoning about the differences observed between the two scenarios

WEAK	**STRONG**
icc -g -O3 -std=c99 -march=native -fopenmp pi_omp.c -o pi_omp Executing pi_omp with one thread	rm -rf pi_seq pi_omp icc -g -O3 -std=c99 -march=native pi_seq.c -o pi_seq icc -g -O3 -std=c99 -march=native -fopenmp pi_omp.c -o

Run 0...Elapsed time = 0.074626
Run 1...Elapsed time = 0.076099
Run 2...Elapsed time = 0.074909
ELAPSED TIME MIN OF 3 EXECUTIONS =0.074626

pi_omp 100000000 1 20 3
Starting OpenMP executions...
Executing pi_omp with 1 threads
Run 0 with size 100000000 ...Elapsed time = 0.076053
Run 1 with size 100000000 ...Elapsed time = 0.075751
Run 2 with size 100000000 ...Elapsed time = 0.075459
ELAPSED TIME Min OF 3 EXECUTIONS =0.075459

Executing pi_omp with 2 threads
Run 0 with size 200000000 ...Elapsed time = 0.075978
Run 1 with size 200000000 ...Elapsed time = 0.076175
Run 2 with size 200000000 ...Elapsed time = 0.076252
ELAPSED TIME Min OF 3 EXECUTIONS =0.075978

Executing pi_omp with 3 threads
Run 0 with size 300000000 ...Elapsed time = 0.077130
Run 1 with size 300000000 ...Elapsed time = 0.077144
Run 2 with size 300000000 ...Elapsed time = 0.077248
ELAPSED TIME Min OF 3 EXECUTIONS =0.077130

Executing pi_omp with 4 threads
Run 0 with size 400000000 ...Elapsed time = 0.076761
Run 1 with size 400000000 ...Elapsed time = 0.076929
Run 2 with size 400000000 ...Elapsed time = 0.076163
ELAPSED TIME Min OF 3 EXECUTIONS =0.076163

Executing pi_omp with 5 threads
Run 0 with size 500000000 ...Elapsed time = 0.079841
Run 1 with size 500000000 ...Elapsed time = 0.080562
Run 2 with size 500000000 ...Elapsed time = 0.079603
ELAPSED TIME Min OF 3 EXECUTIONS =0.079603

Executing pi_omp with 6 threads
Run 0 with size 600000000 ...Elapsed time = 0.079914
Run 1 with size 600000000 ...Elapsed time = 0.080315
Run 2 with size 600000000 ...Elapsed time = 0.081304
ELAPSED TIME Min OF 3 EXECUTIONS =0.079914

Executing pi_omp with 7 threads
Run 0 with size 700000000 ...Elapsed time = 0.080367
Run 1 with size 700000000 ...Elapsed time = 0.080598
Run 2 with size 700000000 ...Elapsed time = 0.080531
ELAPSED TIME Min OF 3 EXECUTIONS =0.080367

Executing pi_omp with 8 threads
Run 0 with size 800000000 ...Elapsed time = 0.080748
Run 1 with size 800000000 ...Elapsed time = 0.081048
Run 2 with size 800000000 ...Elapsed time = 0.081314
ELAPSED TIME Min OF 3 EXECUTIONS =0.080748

Executing pi_omp with 9 threads
Run 0 with size 900000000 ...Elapsed time = 0.082612
Run 1 with size 900000000 ...Elapsed time = 0.082217
Run 2 with size 900000000 ...Elapsed time = 0.082337
ELAPSED TIME Min OF 3 EXECUTIONS =0.082217

Executing pi_omp with 10 threads
Run 0 with size 1000000000 ...Elapsed time = 0.082602
Run 1 with size 1000000000 ...Elapsed time = 0.081995
Run 2 with size 1000000000 ...Elapsed time = 0.082006
ELAPSED TIME Min OF 3 EXECUTIONS =0.081995

Executing pi_omp with 11 threads
Run 0 with size 1100000000 ...Elapsed time = 0.083006
Run 1 with size 1100000000 ...Elapsed time = 0.082444
Run 2 with size 1100000000 ...Elapsed time = 0.082230
ELAPSED TIME Min OF 3 EXECUTIONS =0.082230

Executing pi_omp with 12 threads

pi_omp
Executing pi_seq sequentially
Run 0...Elapsed time = 0.690294
Run 1...Elapsed time = 0.691486
Run 2...Elapsed time = 0.691793
ELAPSED TIME MIN OF 3 EXECUTIONS =0.690294

pi_omp 1000000000 1 20 3
Starting OpenMP executions...
Executing pi_omp with 1 threads
Run 0...Elapsed time = 0.693676
Run 1...Elapsed time = 0.692947
Run 2...Elapsed time = 0.694017
ELAPSED TIME MIN OF 3 EXECUTIONS =0.692947

Executing pi_omp with 2 threads
Run 0...Elapsed time = 0.351592
Run 1...Elapsed time = 0.352092
Run 2...Elapsed time = 0.352397
ELAPSED TIME MIN OF 3 EXECUTIONS =0.351592

Executing pi_omp with 3 threads
Run 0...Elapsed time = 0.240279
Run 1...Elapsed time = 0.236628
Run 2...Elapsed time = 0.240183
ELAPSED TIME MIN OF 3 EXECUTIONS =0.236628

Executing pi_omp with 4 threads
Run 0...Elapsed time = 0.180238
Run 1...Elapsed time = 0.181476
Run 2...Elapsed time = 0.181677
ELAPSED TIME MIN OF 3 EXECUTIONS =0.180238

Executing pi_omp with 5 threads
Run 0...Elapsed time = 0.153324
Run 1...Elapsed time = 0.152609
Run 2...Elapsed time = 0.152762
ELAPSED TIME MIN OF 3 EXECUTIONS =0.152609

Executing pi_omp with 6 threads
Run 0...Elapsed time = 0.128628
Run 1...Elapsed time = 0.128821
Run 2...Elapsed time = 0.129142
ELAPSED TIME MIN OF 3 EXECUTIONS =0.128628

Executing pi_omp with 7 threads
Run 0...Elapsed time = 0.112157
Run 1...Elapsed time = 0.112387
Run 2...Elapsed time = 0.112155
ELAPSED TIME MIN OF 3 EXECUTIONS =0.112155

Executing pi_omp with 8 threads
Run 0...Elapsed time = 0.099469
Run 1...Elapsed time = 0.100864
Run 2...Elapsed time = 0.100364
ELAPSED TIME MIN OF 3 EXECUTIONS =0.099469

Executing pi_omp with 9 threads
Run 0...Elapsed time = 0.090794
Run 1...Elapsed time = 0.090360
Run 2...Elapsed time = 0.089716
ELAPSED TIME MIN OF 3 EXECUTIONS =0.089716

Executing pi_omp with 10 threads
Run 0...Elapsed time = 0.082819
Run 1...Elapsed time = 0.082955
Run 2...Elapsed time = 0.083025
ELAPSED TIME MIN OF 3 EXECUTIONS =0.082819

Executing pi_omp with 11 threads
Run 0...Elapsed time = 0.075541
Run 1...Elapsed time = 0.076605
Run 2...Elapsed time = 0.076468
ELAPSED TIME MIN OF 3 EXECUTIONS =0.075541

Run 0 with size 1200000000 ...Elapsed time = 0.082031 Run 1 with size 1200000000 ...Elapsed time = 0.082241 Run 2 with size 1200000000 ...Elapsed time = 0.083371 ELAPSED TIME Min OF 3 EXECUTIONS =0.082031 Executing pi_omp with 13 threads Run 0 with size 1300000000 ...Elapsed time = 0.083085 Run 1 with size 1300000000 ...Elapsed time = 0.083388 Run 2 with size 1300000000 ...Elapsed time = 0.083423 ELAPSED TIME Min OF 3 EXECUTIONS =0.083085 Executing pi_omp with 14 threads Run 0 with size 1400000000 ...Elapsed time = 0.083585 Run 1 with size 1400000000 ...Elapsed time = 0.082949 Run 2 with size 1400000000 ...Elapsed time = 0.083506 ELAPSED TIME Min OF 3 EXECUTIONS =0.082949 Executing pi_omp with 15 threads Run 0 with size 1500000000 ...Elapsed time = 0.084637 Run 1 with size 1500000000 ...Elapsed time = 0.083336 Run 2 with size 1500000000 ...Elapsed time = 0.083908 ELAPSED TIME Min OF 3 EXECUTIONS =0.083336 Executing pi_omp with 16 threads Run 0 with size 1600000000 ...Elapsed time = 0.085942 Run 1 with size 1600000000 ...Elapsed time = 0.085410 Run 2 with size 1600000000 ...Elapsed time = 0.084825 ELAPSED TIME Min OF 3 EXECUTIONS =0.084825 Executing pi_omp with 17 threads Run 0 with size 1700000000 ...Elapsed time = 0.089002 Run 1 with size 1700000000 ...Elapsed time = 0.088420 Run 2 with size 1700000000 ...Elapsed time = 0.088772 ELAPSED TIME Min OF 3 EXECUTIONS =0.088420 Executing pi_omp with 18 threads Run 0 with size 1800000000 ...Elapsed time = 0.088755 Run 1 with size 1800000000 ...Elapsed time = 0.088784 Run 2 with size 1800000000 ...Elapsed time = 0.088247 ELAPSED TIME Min OF 3 EXECUTIONS =0.088247 Executing pi_omp with 19 threads Run 0 with size 1900000000 ...Elapsed time = 0.089360 Run 1 with size 1900000000 ...Elapsed time = 0.088515 Run 2 with size 1900000000 ...Elapsed time = 0.088743 ELAPSED TIME Min OF 3 EXECUTIONS =0.088515 Executing pi_omp with 20 threads Run 0 with size 2000000000 ...Elapsed time = 0.089357 Run 1 with size 2000000000 ...Elapsed time = 0.088352 Run 2 with size 2000000000 ...Elapsed time = 0.088327 ELAPSED TIME Min OF 3 EXECUTIONS =0.088327 Resultat de l'experiment (tambe es troben a ./executiontime-boada-13.txt i ./efficiency-boada-13.txt) #threads Elapsed min 1 0.075459 2 0.075978 3 0.077130 4 0.076163 5 0.079603 6 0.079914 7 0.080367 8 0.080748 9 0.082217 10 0.081995 11 0.082230 12 0.082031 13 0.083085 14 0.082949 15 0.083336 16 0.084825 17 0.088420 18 0.088247	Executing pi_omp with 12 threads Run 0...Elapsed time = 0.070135 Run 1...Elapsed time = 0.070201 Run 2...Elapsed time = 0.070505 ELAPSED TIME MIN OF 3 EXECUTIONS =0.070135 Executing pi_omp with 13 threads Run 0...Elapsed time = 0.064975 Run 1...Elapsed time = 0.065226 Run 2...Elapsed time = 0.066048 ELAPSED TIME MIN OF 3 EXECUTIONS =0.064975 Executing pi_omp with 14 threads Run 0...Elapsed time = 0.061125 Run 1...Elapsed time = 0.062040 Run 2...Elapsed time = 0.061038 ELAPSED TIME MIN OF 3 EXECUTIONS =0.061038 Executing pi_omp with 15 threads Run 0...Elapsed time = 0.058484 Run 1...Elapsed time = 0.059889 Run 2...Elapsed time = 0.058156 ELAPSED TIME MIN OF 3 EXECUTIONS =0.058156 Executing pi_omp with 16 threads Run 0...Elapsed time = 0.055531 Run 1...Elapsed time = 0.054851 Run 2...Elapsed time = 0.055139 ELAPSED TIME MIN OF 3 EXECUTIONS =0.054851 Executing pi_omp with 17 threads Run 0...Elapsed time = 0.054892 Run 1...Elapsed time = 0.053801 Run 2...Elapsed time = 0.054570 ELAPSED TIME MIN OF 3 EXECUTIONS =0.053801 Executing pi_omp with 18 threads Run 0...Elapsed time = 0.052432 Run 1...Elapsed time = 0.052277 Run 2...Elapsed time = 0.051478 ELAPSED TIME MIN OF 3 EXECUTIONS =0.051478 Executing pi_omp with 19 threads Run 0...Elapsed time = 0.050478 Run 1...Elapsed time = 0.049663 Run 2...Elapsed time = 0.064199 ELAPSED TIME MIN OF 3 EXECUTIONS =0.049663 Executing pi_omp with 20 threads Run 0...Elapsed time = 0.048094 Run 1...Elapsed time = 0.047890 Run 2...Elapsed time = 0.047751 ELAPSED TIME MIN OF 3 EXECUTIONS =0.047751 Resultat de l'experiment (tambe es troben a ./elapsed-boada-13.txt i ./speedup-boada-13.txt) #threads Elapsed min 1 0.692947 2 0.351592 3 0.236628 4 0.180238 5 0.152609 6 0.128628 7 0.112155 8 0.099469 9 0.089716 10 0.082819 11 0.075541 12 0.070135 13 0.064975 14 0.061038 15 0.058156 16 0.054851

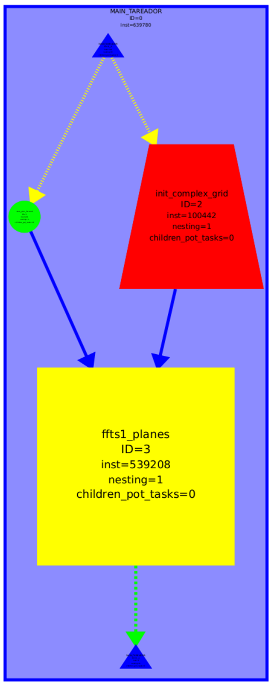
19 0.088515 20 0.088327 #threads Speedup 1 .98896089267019175976 2 .98220537524020111084 3 .96753532996240114093 4 .97981959744232763940 5 .93747723075763476250 6 .93382886603098330705 7 .92856520711237199347 8 .92418388071531183434 9 .90767116289818407385 10 .91012866638209646929 11 .90752766630183631278 12 .90972924869866270068 13 .89818860203406150327 14 .89966123762794005955 15 .89548334453297494480 16 .87976422045387562628 17 .84399457136394480886 18 .84564914388024522080 19 .84308874202112636276 20 .84488321804204829780	17 0.053801 18 0.051478 19 0.049663 20 0.047751 #threads Speedup 1 .99617142436578843692 2 1.96333818744453798721 3 2.91721182615751305847 4 3.82990268422863103230 5 4.52328499629772818116 6 5.36659203283888422427 7 6.15482145245419285809 8 6.93979028642089495219 9 7.69421284943599803825 10 8.33497144375083012352 11 9.13800452734276750373 12 9.84236116061880658729 13 10.62399384378607156598 14 11.30924997542514499164 15 11.86969530229039136116 16 12.58489362090025706003 17 12.83050500920057248006 18 13.40949531838843777924 19 13.89956305499063689265 20 14.45611610228058051140

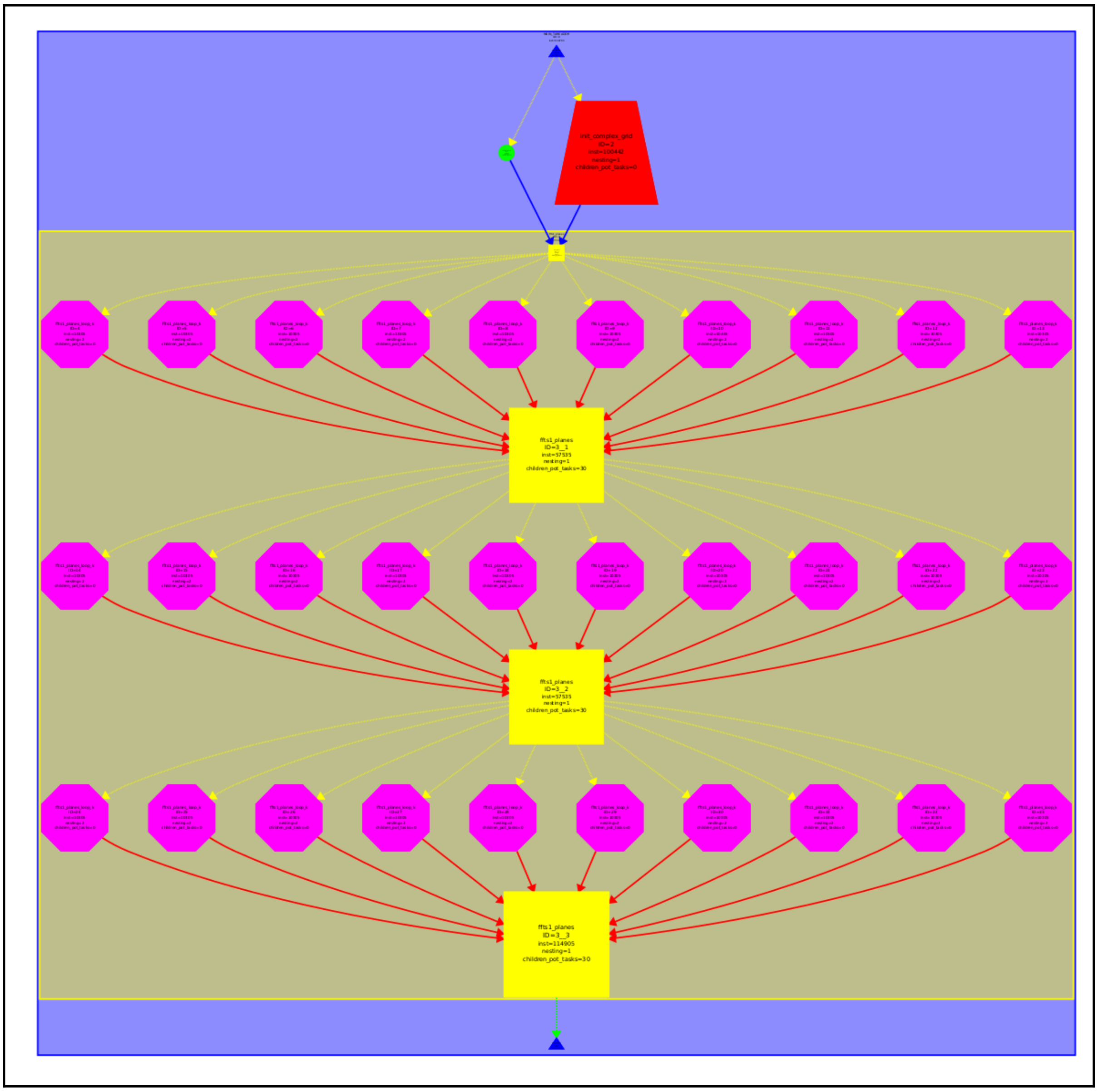
En la weak scalability, el speedup nos perjudica en cuanto al rendimiento, en cambio, en strong scalability, el speedup y el rendimiento mejoran con el número de hilos utilizados.

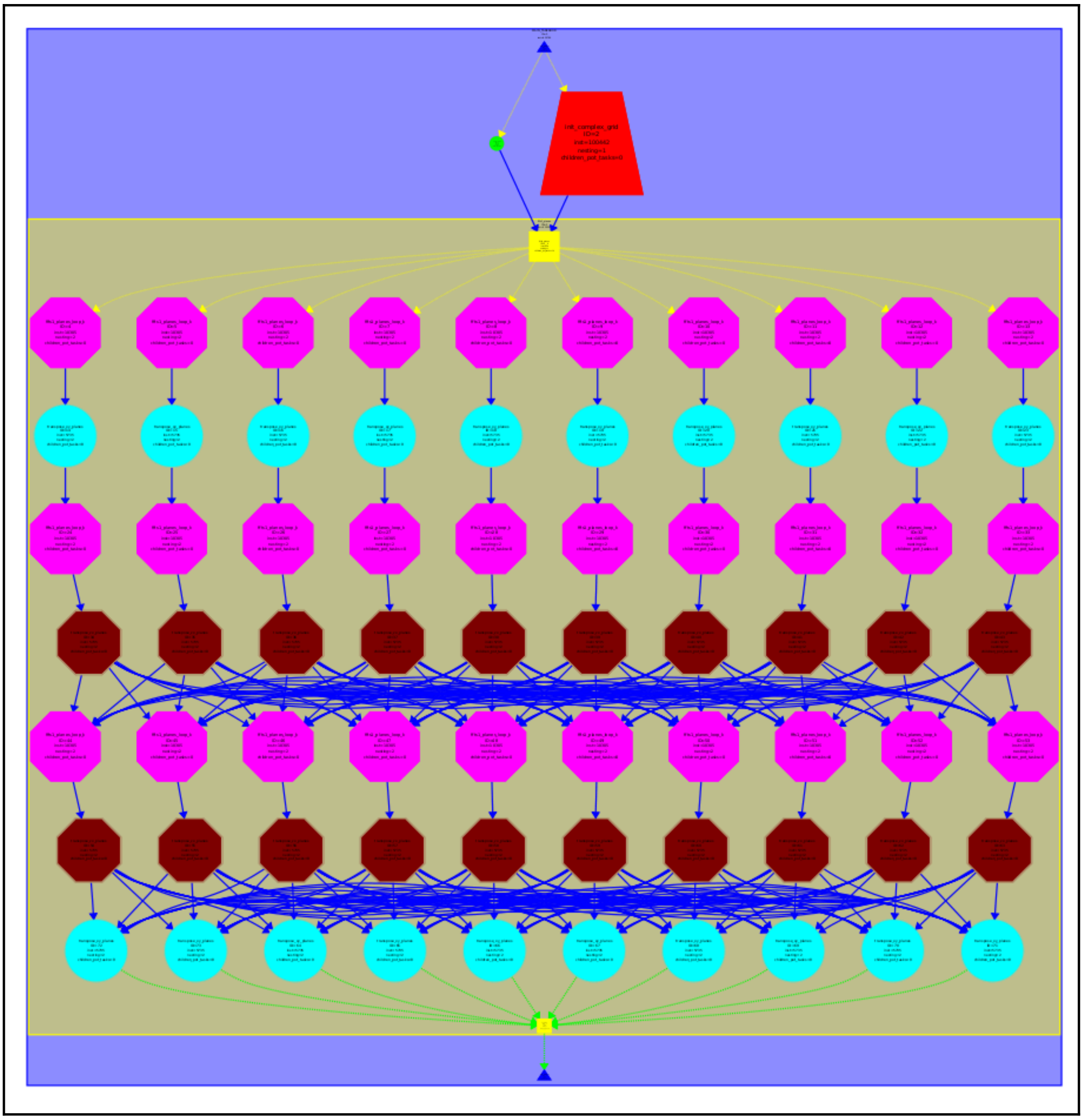
Tasca de análisi sistemático de descomposición con el Tareador

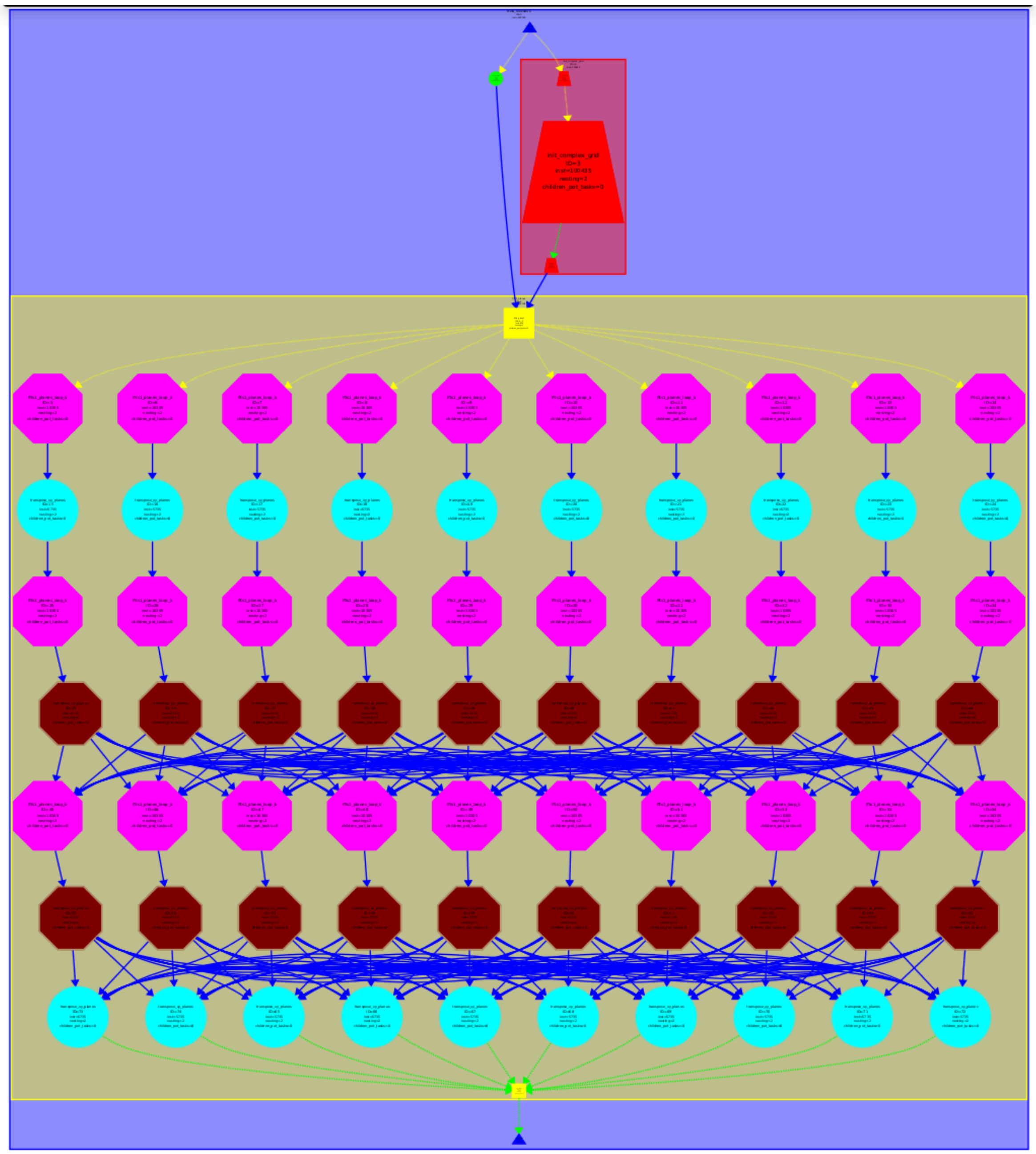
- **Descomposición de tascas para 3DFFT**

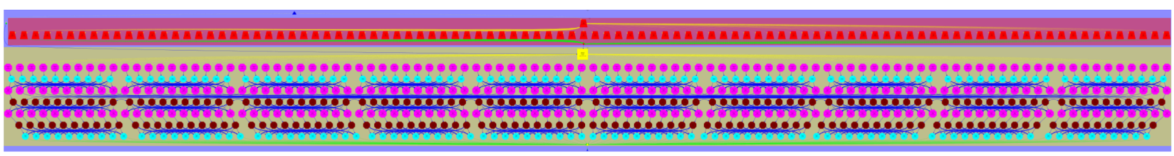
For the deliverable: In the Deliverable you will have to include all the task dependence graphs obtained, the table for the T1, T∞ and the potential parallelism metrics for all the versions explored and the scalability plots for versions v4 and v5, commenting how performance is improving in the process. You should also include the relevant part(s) of the code to understand why v5 is able to scale to a higher number of processors compared to v4.




init_complex_grid
ID=2
inst=100442
nesting=1
children_pot_tasks=0
ffts1_planes
ID=3__1
inst=57535
nesting=1
children_pot_tasks=30
ffts1_planes
ID=3__2
inst=57535
nesting=1
children_pot_tasks=30
ffts1_planes
ID=3__3
inst=114905
nesting=1
children_pot_tasks=30


init_complex_grid
ID=2
inst=100442
nesting=1
children_pot_tasks=0

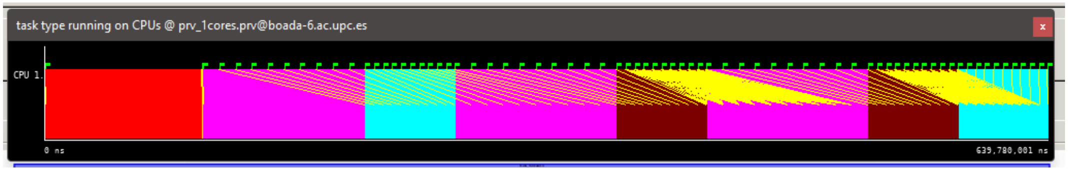
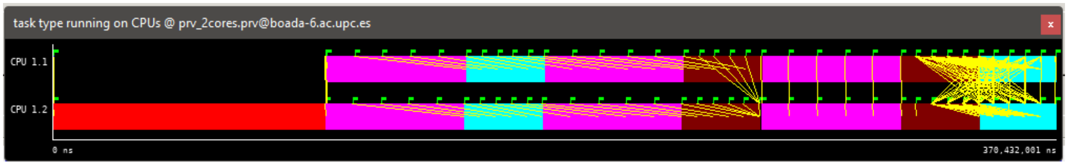
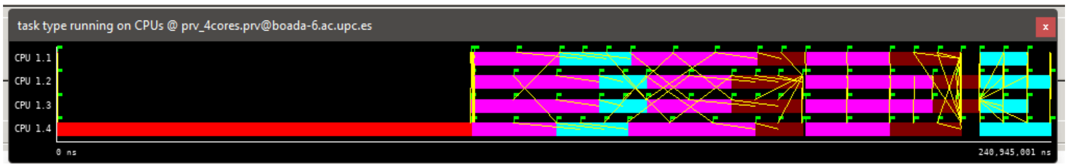
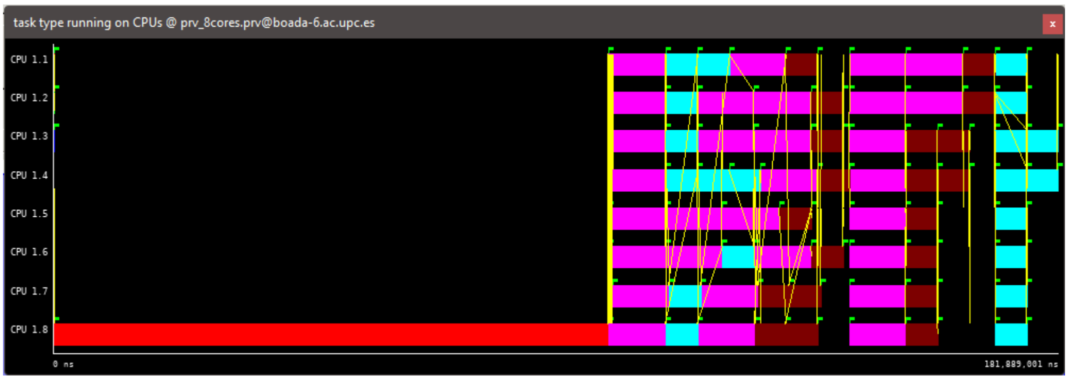


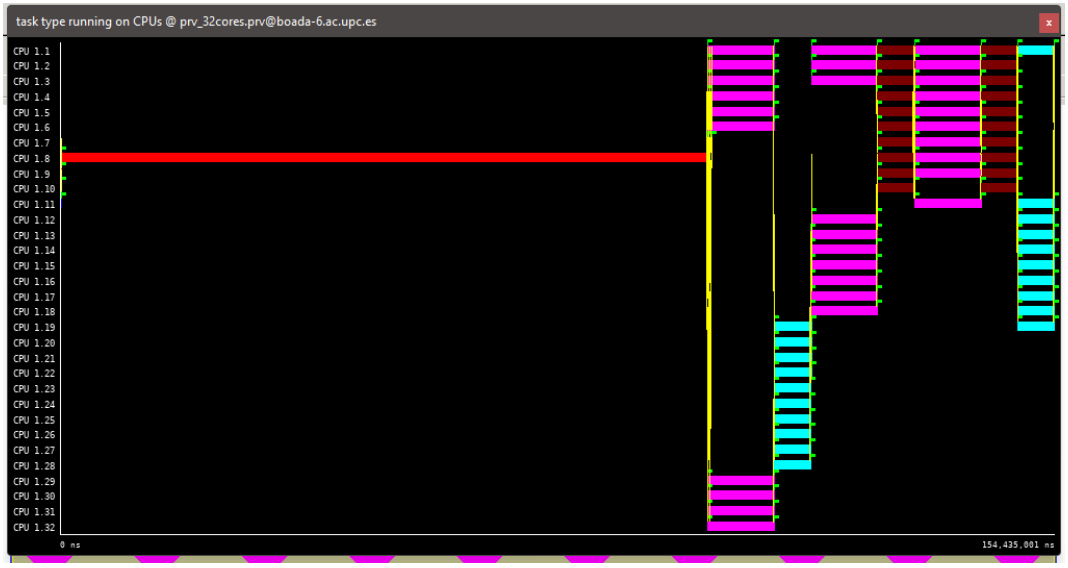
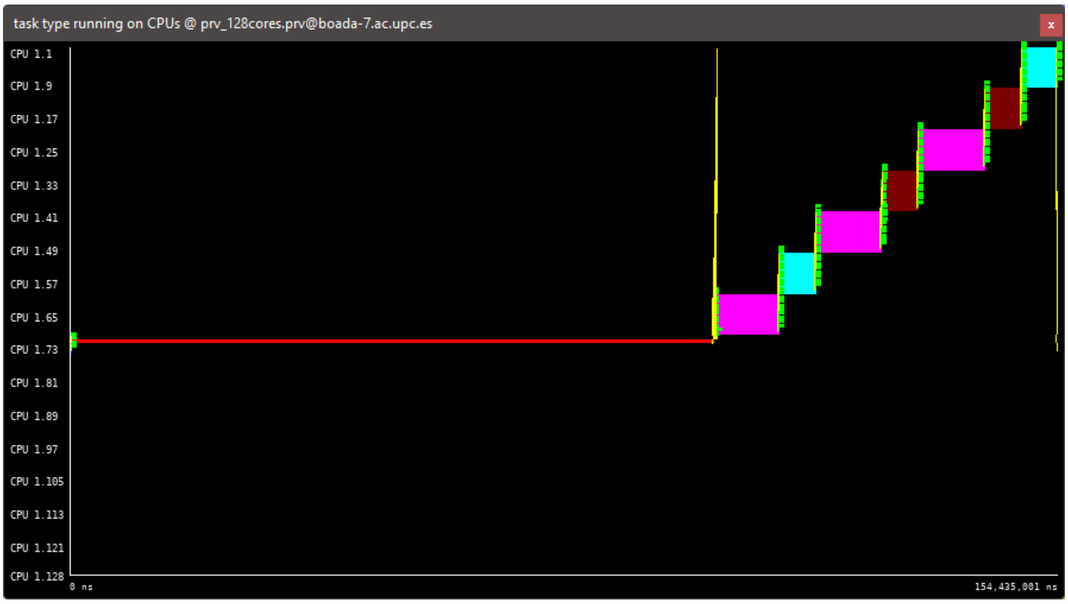


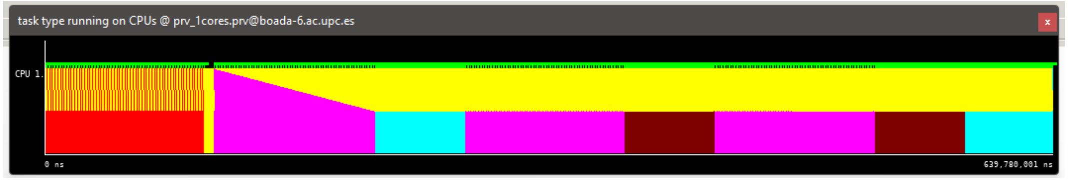
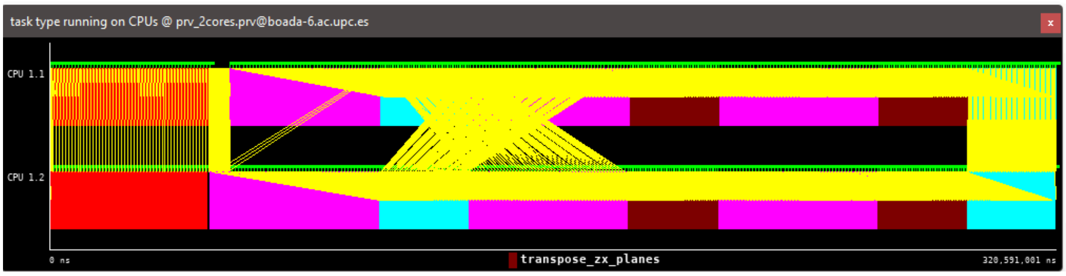
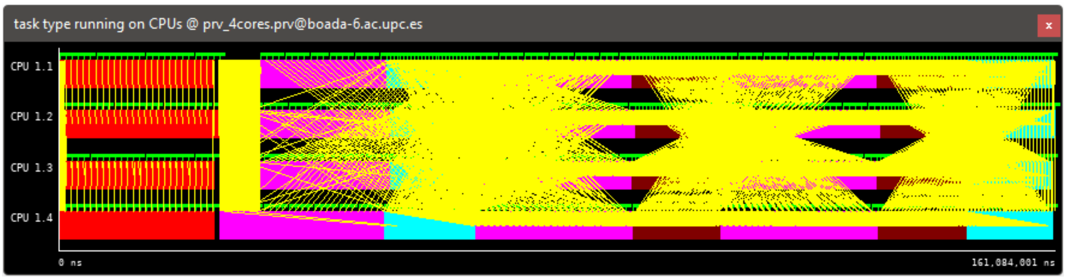
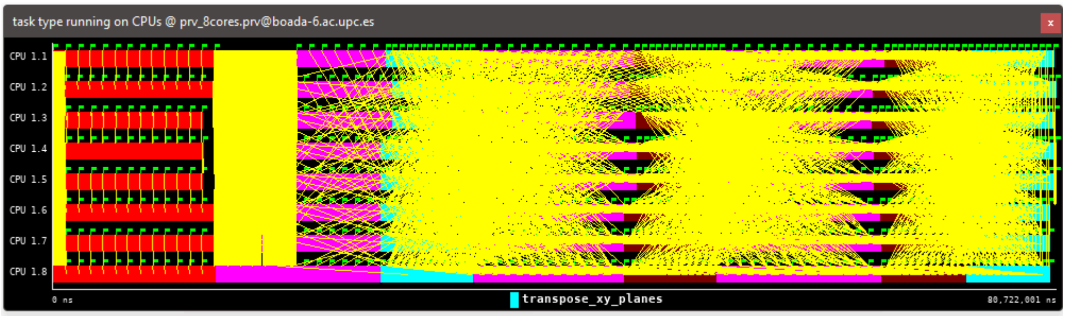
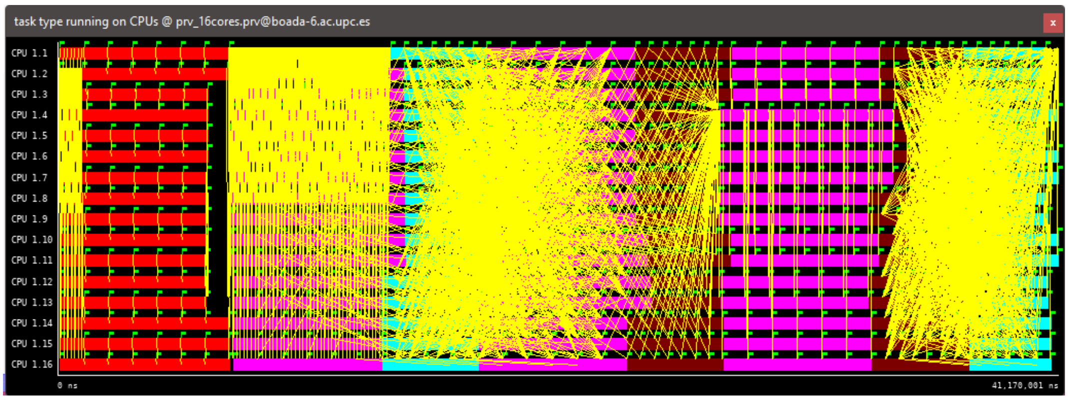
In this part of the report you should summarise the main conclusions from the analysis of task decompositions for the 3DFFT program. Backup your conclusions with the task dependence graphs and the following table properly filled in with the information obtained in the laboratory session for the initial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

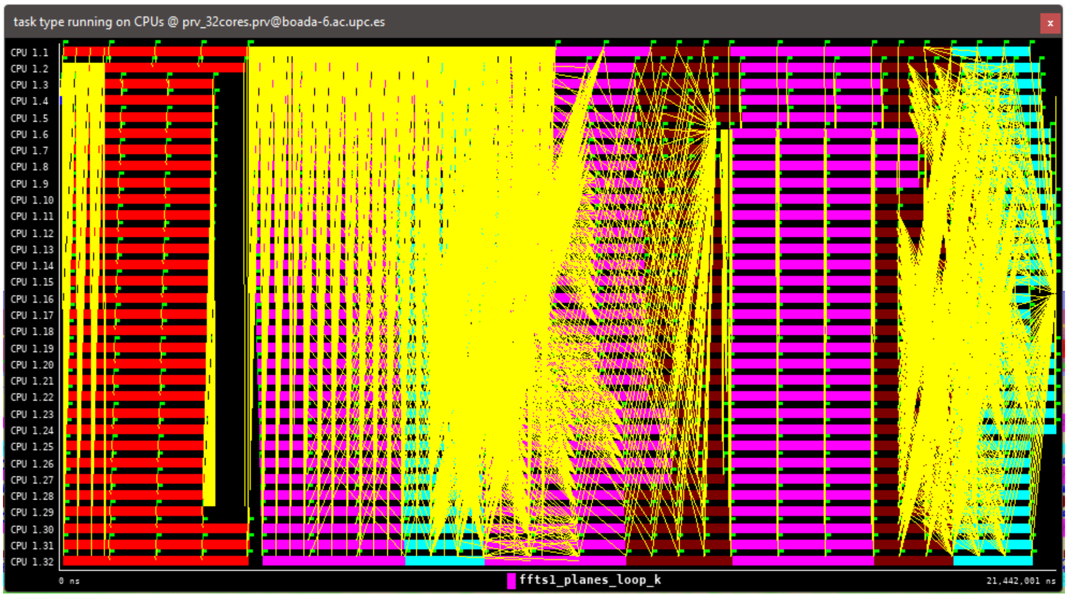
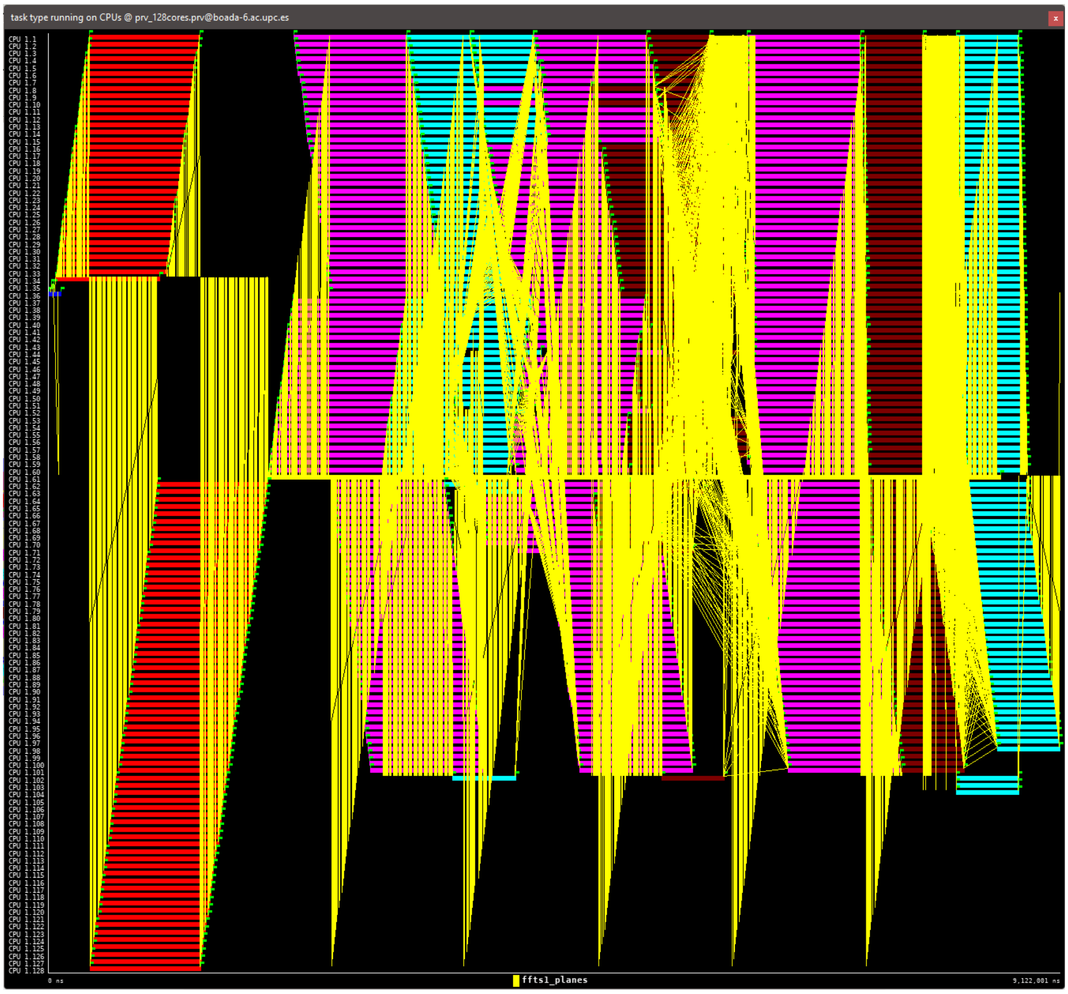
Version	T1	T∞	Parallelism
seq	639780001 ns	639780001 ns	1
v1	639780001 ns	639780001 ns	1
v2	639780001 ns	361472001 ns	1,7699
v3	639780001 ns	154437001 ns	4,1426
v4	639780001 ns	154435001 ns	4,1427
v5	639780001 ns	9122001 ns	70,1359

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 speedup when using 1, 2, 4, 8, 16, 32 and 128 (as ∞) processors, as reported by the simulation module inside Tareador . You should also include the relevant(s) part(s) of the code to understand why v5 is able to scale to a higher number of processors compared to v4.

	V4
1	
2	
4	
8	

16	task type running on CPUs @ prv_16cores.prv@boada-6.ac.upc.es CPU 1.1 CPU 1.2 CPU 1.3 CPU 1.4 CPU 1.5 CPU 1.6 CPU 1.7 CPU 1.8 CPU 1.9 CPU 1.10 CPU 1.11 CPU 1.12 CPU 1.13 CPU 1.14 CPU 1.15 CPU 1.16 0 ns 154,435,001 ns
32	task type running on CPUs @ prv_32cores.prv@boada-6.ac.upc.es CPU 1.1 CPU 1.2 CPU 1.3 CPU 1.4 CPU 1.5 CPU 1.6 CPU 1.7 CPU 1.8 CPU 1.9 CPU 1.10 CPU 1.11 CPU 1.12 CPU 1.13 CPU 1.14 CPU 1.15 CPU 1.16 CPU 1.17 CPU 1.18 CPU 1.19 CPU 1.20 CPU 1.21 CPU 1.22 CPU 1.23 CPU 1.24 CPU 1.25 CPU 1.26 CPU 1.27 CPU 1.28 CPU 1.29 CPU 1.30 CPU 1.31 CPU 1.32 0 ns 154,435,001 ns
128	task type running on CPUs @ prv_128cores.prv@boada-7.ac.upc.es CPU 1.1 CPU 1.9 CPU 1.17 CPU 1.25 CPU 1.33 CPU 1.41 CPU 1.49 CPU 1.57 CPU 1.65 CPU 1.73 CPU 1.81 CPU 1.89 CPU 1.97 CPU 1.105 CPU 1.113 CPU 1.121 CPU 1.128 0 ns 154,435,001 ns

	V5
1	task type running on CPUs @ prv_1cores.prv@boada-6.ac.upc.es
2	task type running on CPUs @ prv_2cores.prv@boada-6.ac.upc.es CPU 1.1 CPU 1.2 0 ns transpose_zx_planes 320,551,001 ns
4	task type running on CPUs @ prv_4cores.prv@boada-6.ac.upc.es CPU 1.1 CPU 1.2 CPU 1.3 CPU 1.4 0 ns 161,084,001 ns
8	task type running on CPUs @ prv_8cores.prv@boada-6.ac.upc.es CPU 1.1 CPU 1.2 CPU 1.3 CPU 1.4 CPU 1.5 CPU 1.6 CPU 1.7 CPU 1.8 0 ns transpose_xy_planes 80,722,001 ns
16	task type running on CPUs @ prv_16cores.prv@boada-6.ac.upc.es 0 ns 41,170,001 ns

32	 task type running on CPUs @ prv_32cores.prv@boada-6.ac.upc.es 0 ns ffts1_planes_loop_k 21,442,001 ns
128	 task type running on CPUs @ prv_128cores.prv@boada-6.ac.upc.es 0 ns ffts1_planes

Codi V5:

```
void start_plan_forward(fftwf_complex in_fftw[][N][N], fftwf_plan *p1d) {
#if TEST
    p = fftwf_plan_dft_3d(N, N, N, (fftwf_complex *)in_fftw, (fftwf_complex *)in_fftw,
FFTW_FORWARD, FFTW_ESTIMATE);
#endif
    *p1d = fftwf_plan_dft_1d(N, (fftwf_complex *)in_fftw, (fftwf_complex *)in_fftw,
FFTW_FORWARD, FFTW_ESTIMATE);
}

void init_complex_grid(fftwf_complex in_fftw[][N][N]) {
  int k,j,i;
  for (k = 0; k < N; k++) {
    for (j = 0; j < N; j++) {
 tareador_start_task("init_complex_grid");
      for (i = 0; i < N; i++)
      {
        in_fftw[k][j][i][0] = (float)
(sin(M_PI*((float)i)/64.0)+sin(M_PI*((float)i)/32.0)+sin(M_PI*((float)i/16.0)));
        in_fftw[k][j][i][1] = 0;
#if TEST
        out_fftw[k][j][i][0]= in_fftw[k][j][i][0];
        out_fftw[k][j][i][1]= in_fftw[k][j][i][1];
#endif
      }
  tareador_end_task("init_complex_grid");
    }
  }
}
```

Entender la ejecución de progamas con OpenMP

- ***Modelfactors: Análisis global***

For the deliverable: Based on the output metrics of modelfactors we ask you the following questions: Is the scalability appropriate? Is the overhead due to synchronization neglegible? Is this overhead affecting the execution time per explicit task? Which is the parallel fraction (φ) for this version of the program? Is the efficiency for the parallel regions appropriate? Which is the factor that is negatively influencing the most?

No es apropiada, podemos ver como va decreciendo cuanto mayor es el numero de hilos de procesamiento, como observamos en la tabla, su eficiencia decrece muy rápido.

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	1.32	0.77	0.81	1.26	1.50
Speedup	1.00	1.73	1.64	1.05	0.89
Efficiency	1.00	0.43	0.21	0.09	0.06

Table 1: Analysis done on Wed Sep 21 01:01:29 PM CEST 2022, paco1208

En esta otra tabla podemos observar que existe un claro problema de sincronización, es decir, se existe una paralelización excesiva, ya que observamos que el overhead de sincronización es de un 74,17%, un valor elevadísimo. La fracción paralela es de un 83,40%.

Overview of the Efficiency metrics in parallel fraction, ϕ=83.40%					
Number of processors	1	4	8	12	16
Global efficiency	98.81%	49.24%	24.09%	8.82%	5.45%
Parallelization strategy efficiency	98.81%	89.00%	87.04%	73.41%	55.56%
Load balancing	100.00%	98.35%	97.72%	98.14%	97.63%
In execution efficiency	98.81%	90.50%	89.07%	74.80%	56.91%
Scalability for computation tasks	100.00%	55.33%	27.67%	12.01%	9.81%
IPC scalability	100.00%	68.88%	50.94%	39.42%	40.12%
Instruction scalability	100.00%	98.33%	96.19%	94.13%	92.18%
Frequency scalability	100.00%	81.70%	56.48%	32.37%	26.51%

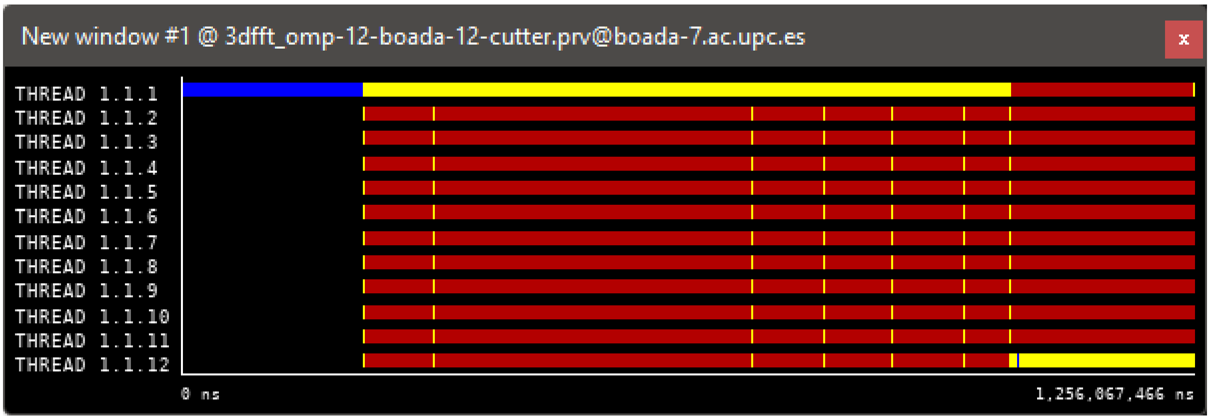
Table 2: Analysis done on Wed Sep 21 01:01:29 PM CEST 2022, paco1208

Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	17920.0	71680.0	143360.0	215040.0	286720.0
LB (number of explicit tasks executed)	1.0	0.93	0.85	0.83	0.83
LB (time executing explicit tasks)	1.0	0.99	0.98	0.98	0.98
Time per explicit task (average us)	60.85	27.51	27.51	42.26	38.82
Overhead per explicit task (synch %)	0.16	10.81	12.86	32.53	74.17
Overhead per explicit task (sched %)	1.04	1.53	1.99	3.66	5.8
Number of taskwait/taskgroup (total)	1792.0	1792.0	1792.0	1792.0	1792.0

Table 3: Analysis done on Wed Sep 21 01:01:29 PM CEST 2022, paco1208

- **Obtener detalles de paralelización haciendo Paraver**

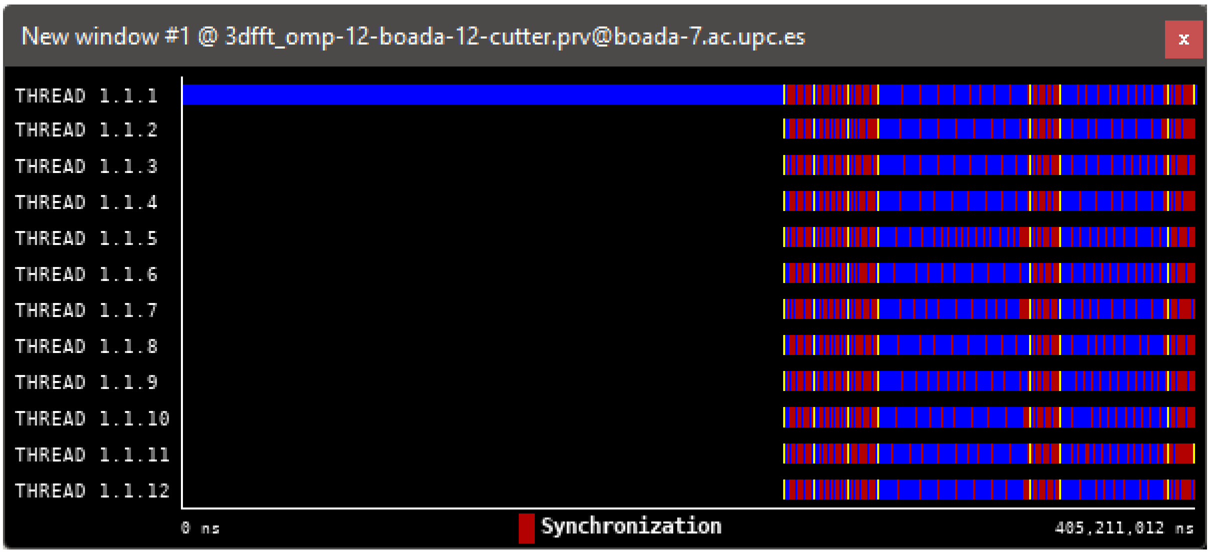
For the deliverable: There are two key factors that influence the overall scalability and final performance. Looking at the two timelines windows we can see these two factors:
1) there is one function that is not parallelised and
2) there is a thread state that is predominant along the timeline window. At this moment, the second one seems to more important because there is a strong scalability problem (slowdown from 12 threads) due to the number of tasks and synchronizations in the program. Do you think this overhead problem is constant or function of the number of threads? Why? Look again at Table 4.3 of modelfactors execution you have just done.



Como podemos ver en la imagen, el problema mas destacado es de sincronización (rojo). En el grafico podemos apreciar los dos factores que hacen que una tarea no sea paralelizable, en el primer tramo (azul) la función no esta paralelizada y corre en un solo hilo. En el segundo tramo, donde empezamos a ver paralelización, vemos que la mayoría del tiempo se pierde sincronizando resultados, eso se debe a un exceso de paralelización de las tareas.

- **Tascas Implícitas versus Explícitas**

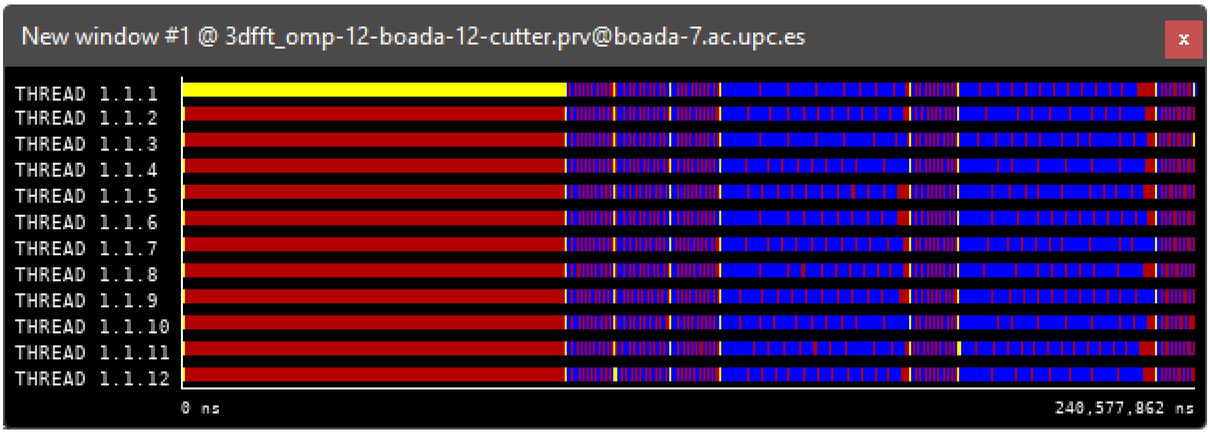
For the deliverable: Once you have seen the histograms and timeline windows of the explicit task durations, what kind of explicit task granularity do you think you have: fine or coarse grain?



Hemos mejorado el tiempo en un 300% Debido a que hemos reducido la granularidad respecto a la versión anterior,por eso ya no perdemos tanto tiempo en la sincronía.
Ahora el problema que tenemos es otro, la primera función es secuencial y no la tenemos paralizada.

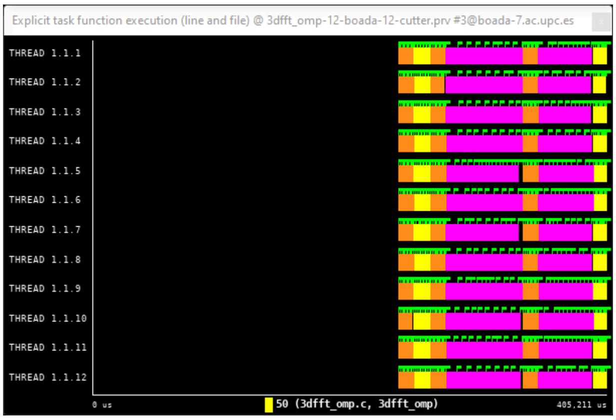
- **Reducción de costes generales de Paralelización**

For the deliverable: Have we improved the overall performance? Is there any slowdown? Do you observe any major difference on the overheads of the explicit tasks for the initial and optimized versions? On the other hand, why do you think you can not achieve better speedup for 12 or more threads? Hint: Compute the maximum speedup that you achieve with the current parallel fraction that you are parallelising (see modelfactors results). This value is limiting the maximum speedup that we can achieve for the machine we have.



Hemos mejorado el rendimiento de la primera parte del código, pero con demasiada granularidad. El código sería más ágil si disminuyera mos la granularidad de la primera función. Como hemos visto anteriormente el problema está en las sincronía, cuantos más hilos de procesamiento usemos más sincronía tendrá que haber entre ellos por lo cual se verá afectado nuestro rendimiento.

For the deliverable: Do you observe any major difference on the duration of the implicit and explicit tasks for the initial and optimized versions? What is the function that is limiting the maximum speedup that you can achieve? Which functions in the program are or not parallelized? Which is the duration of the sequential execution for those user function not parallelised?



Como podemos observar en las imágenes las tareas implícitas son más largas que las explícitas. Por otra parte las explícitas tienen una duración más corta pero hay muchas más.

En las versiones no optimizadas, veríamos como existen un mayor número de tareas explicitas, que pese a durar menos, al haber más el rendimiento es peor.

- **Mejora de φ y análisis**

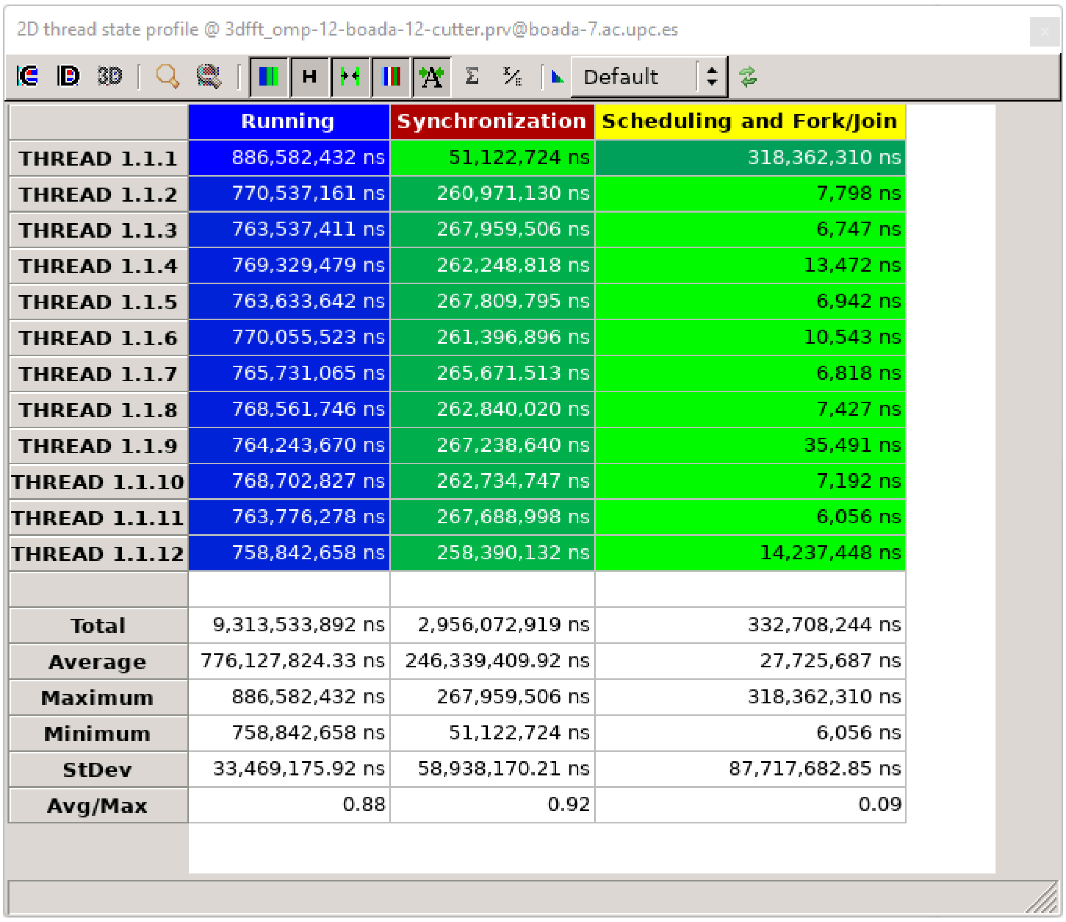
For the deliverable: Have the speed–up and efficiency metrics improved? What is the new value for φ? Compare the three version: initial, reducing overhead, and improving φ under the point of view of strong scalability

In this final section of your report you should comment the parallel performance evolution that you observed for the three OpenMP parallel versions of 3DFFT. Along the document we have indicated questions and analysis to be done for the deliverable. You should comment about the (strong) scalability plots (execution time and speed–up) when varying the number of threads for the three parallel versions of 3DFFT. Support your explanations with the results reported by modelfactors and Paraver and fill in the following table.

Hemos observado qué la granularidad de las paralización es muy importante ya que si nos pasamos de granularidad solo conseguiremos que el tiempo de sincronía y de fork aumente mucho por lo que perderemos rendimiento. En la segunda versión hemos alcanzado una granularidad muy correcta y el rendimiento se ha visto aumentado por mucho, El problema que nos hemos encontrado era la primera función que no estaba paralizada cosa que hemos mejorado en la última versión la hemos paralizado con una granularidad muy fina. Si quisiéramos acabar de optimizar el código personalmente creemos que habría que reducir la granularidad de la tercera versión en la primera función ya que nos pasa algo parecido a lo que nos pasaba en la primera función y es que se pierden muchísimo tiempo sincronizando tareas.

Version	φ	ideal S_{12}	T_1	T_{12}	real S_{12}
V1	83,4 %	10,008	12602315055 ns	1050192921 ns	1,05
V2	83,18 %	9,9816	2211838714 ns	184319892,8 ns	3,08
V3	99,96 %	11,9952	2880363240 ns	240030270 ns	5,54

V1



2D thread state profile @ 3dfft_omp-12-boada-12-cutter.prv@boada-7.ac.upc.es

	Running	Synchronization	Scheduling and Fork/Join
THREAD 1.1.1	886,582,432 ns	51,122,724 ns	318,362,310 ns
THREAD 1.1.2	770,537,161 ns	260,971,130 ns	7,798 ns
THREAD 1.1.3	763,537,411 ns	267,959,506 ns	6,747 ns
THREAD 1.1.4	769,329,479 ns	262,248,818 ns	13,472 ns
THREAD 1.1.5	763,633,642 ns	267,809,795 ns	6,942 ns
THREAD 1.1.6	770,055,523 ns	261,396,896 ns	10,543 ns
THREAD 1.1.7	765,731,065 ns	265,671,513 ns	6,818 ns
THREAD 1.1.8	768,561,746 ns	262,840,020 ns	7,427 ns
THREAD 1.1.9	764,243,670 ns	267,238,640 ns	35,491 ns
THREAD 1.1.10	768,702,827 ns	262,734,747 ns	7,192 ns
THREAD 1.1.11	763,776,278 ns	267,688,998 ns	6,056 ns
THREAD 1.1.12	758,842,658 ns	258,390,132 ns	14,237,448 ns
Total	9,313,533,892 ns	2,956,072,919 ns	332,708,244 ns
Average	776,127,824.33 ns	246,339,409.92 ns	27,725,687 ns
Maximum	886,582,432 ns	267,959,506 ns	318,362,310 ns
Minimum	758,842,658 ns	51,122,724 ns	6,056 ns
StDev	33,469,175.92 ns	58,938,170.21 ns	87,717,682.85 ns
Avg/Max	0.88	0.92	0.09

V2

2D thread state profile @ 3dfft_omp-12-boada-12-cutter.prv@boada-7.ac.upc.es

Default

	Running	Synchronization	Scheduling and Fork/Join
THREAD 1.1.1	399,351,073 ns	5,512,717 ns	347,222 ns
THREAD 1.1.2	158,094,522 ns	6,166,045 ns	9,016 ns
THREAD 1.1.3	159,090,949 ns	5,173,394 ns	25,656 ns
THREAD 1.1.4	159,994,656 ns	4,266,000 ns	10,678 ns
THREAD 1.1.5	158,154,375 ns	5,973,529 ns	133,048 ns
THREAD 1.1.6	162,760,856 ns	1,401,378 ns	7,810 ns
THREAD 1.1.7	155,849,708 ns	8,327,438 ns	6,100 ns
THREAD 1.1.8	160,224,189 ns	3,974,869 ns	9,450 ns
THREAD 1.1.9	160,641,826 ns	3,626,205 ns	26,359 ns
THREAD 1.1.10	158,014,236 ns	6,186,680 ns	9,762 ns
THREAD 1.1.11	159,756,908 ns	4,130,893 ns	345,601 ns
THREAD 1.1.12	158,381,463 ns	5,847,336 ns	6,767 ns
Total	2,150,314,761 ns	60,586,484 ns	937,469 ns
Average	179,192,896.75 ns	5,048,873.67 ns	78,122.42 ns
Maximum	399,351,073 ns	8,327,438 ns	347,222 ns
Minimum	155,849,708 ns	1,401,378 ns	6,100 ns
StDev	66,400,504.37 ns	1,657,108.96 ns	124,614.25 ns
Avg/Max	0.45	0.61	0.22

V3

2D thread state profile @ 3dfft_omp-12-boada-12-cutter.prv@boada-7.ac.upc.es

Default

	Running	Synchronization	Scheduling and Fork/Join
THREAD 1.1.1	142,971,959 ns	11,019,476 ns	86,586,427 ns
THREAD 1.1.2	174,049,571 ns	65,916,166 ns	12,468 ns
THREAD 1.1.3	172,764,313 ns	67,072,492 ns	135,917 ns
THREAD 1.1.4	173,810,678 ns	66,156,985 ns	15,523 ns
THREAD 1.1.5	169,173,744 ns	70,794,858 ns	12,028 ns
THREAD 1.1.6	173,326,951 ns	66,643,915 ns	15,486 ns
THREAD 1.1.7	173,305,850 ns	66,532,216 ns	138,941 ns
THREAD 1.1.8	173,161,019 ns	66,808,028 ns	13,967 ns
THREAD 1.1.9	172,658,780 ns	67,312,157 ns	12,669 ns
THREAD 1.1.10	174,259,619 ns	65,708,017 ns	13,974 ns
THREAD 1.1.11	170,167,738 ns	69,581,265 ns	228,573 ns
THREAD 1.1.12	172,671,270 ns	67,075,464 ns	234,736 ns
Total	2,042,321,492 ns	750,621,039 ns	87,420,709 ns
Average	170,193,457.67 ns	62,551,753.25 ns	7,285,059.08 ns
Maximum	174,259,619 ns	70,794,858 ns	86,586,427 ns
Minimum	142,971,959 ns	11,019,476 ns	12,028 ns
StDev	8,335,050.89 ns	15,603,112.96 ns	23,910,407.02 ns
Avg/Max	0.98	0.88	0.08

V1

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	1.32	0.77	0.81	1.26	1.50
Speedup	1.00	1.73	1.64	1.05	0.89
Efficiency	1.00	0.43	0.21	0.09	0.06

Table 1: Analysis done on Wed Sep 21 01:01:29 PM CEST 2022, paco1208

Overview of the Efficiency metrics in parallel fraction, ϕ=83.40%					
Number of processors	1	4	8	12	16
Global efficiency	98.81%	49.24%	24.09%	8.82%	5.45%
Parallelization strategy efficiency	98.81%	89.00%	87.04%	73.41%	55.56%
Load balancing	100.00%	98.35%	97.72%	98.14%	97.63%
In execution efficiency	98.81%	90.50%	89.07%	74.80%	56.91%
Scalability for computation tasks	100.00%	55.33%	27.67%	12.01%	9.81%
IPC scalability	100.00%	68.88%	50.94%	39.42%	40.12%
Instruction scalability	100.00%	98.33%	96.19%	94.13%	92.18%
Frequency scalability	100.00%	81.70%	56.48%	32.37%	26.51%

Table 2: Analysis done on Wed Sep 21 01:01:29 PM CEST 2022, paco1208

Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	17920.0	71680.0	143360.0	215040.0	286720.0
LB (number of explicit tasks executed)	1.0	0.93	0.85	0.83	0.83
LB (time executing explicit tasks)	1.0	0.99	0.98	0.98	0.98
Time per explicit task (average us)	60.85	27.51	27.51	42.26	38.82
Overhead per explicit task (synch %)	0.16	10.81	12.86	32.53	74.17
Overhead per explicit task (sched %)	1.04	1.53	1.99	3.66	5.8
Number of taskwait/taskgroup (total)	1792.0	1792.0	1792.0	1792.0	1792.0

Table 3: Analysis done on Wed Sep 21 01:01:29 PM CEST 2022, paco1208

V2

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	1.25	0.56	0.45	0.41	0.37
Speedup	1.00	2.22	2.79	3.08	3.40
Efficiency	1.00	0.55	0.35	0.26	0.21

Table 1: Analysis done on Wed Sep 21 07:34:02 PM CEST 2022, paco1208

Overview of the Efficiency metrics in parallel fraction, ϕ=83.18%					
Number of processors	1	4	8	12	16
Global efficiency	99.96%	76.90%	58.71%	52.49%	44.97%
Parallelization strategy efficiency	99.96%	96.98%	96.96%	96.75%	97.15%
Load balancing	100.00%	97.66%	97.83%	97.57%	97.92%
In execution efficiency	99.96%	99.31%	99.10%	99.16%	99.22%
Scalability for computation tasks	100.00%	79.30%	60.55%	54.26%	46.29%
IPC scalability	100.00%	80.05%	64.76%	59.61%	50.98%
Instruction scalability	100.00%	99.99%	99.98%	99.97%	99.96%
Frequency scalability	100.00%	99.07%	93.53%	91.04%	90.84%

Table 2: Analysis done on Wed Sep 21 07:34:02 PM CEST 2022, paco1208

Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	70.0	280.0	560.0	840.0	1120.0
LB (number of explicit tasks executed)	1.0	0.95	0.9	0.91	0.82
LB (time executing explicit tasks)	1.0	0.98	0.99	0.98	0.98
Time per explicit task (average us)	14800.6	4665.83	3054.92	2272.84	1998.07
Overhead per explicit task (synch %)	0.0	3.04	3.01	3.17	2.72
Overhead per explicit task (sched %)	0.03	0.03	0.04	0.04	0.06
Number of taskwait/taskgroup (total)	7.0	7.0	7.0	7.0	7.0

Table 3: Analysis done on Wed Sep 21 07:34:02 PM CEST 2022, paco1208

V3

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	1.33	0.41	0.27	0.24	0.29
Speedup	1.00	3.25	5.00	5.54	4.66
Efficiency	1.00	0.81	0.63	0.46	0.29

Table 1: Analysis done on Wed Sep 21 08:21:57 PM CEST 2022, paco1208

Overview of the Efficiency metrics in parallel fraction, ϕ=99.96%					
Number of processors	1	4	8	12	16
Global efficiency	99.85%	81.12%	62.52%	46.20%	29.15%
Parallelization strategy efficiency	99.85%	95.37%	92.00%	70.88%	50.79%
Load balancing	100.00%	97.94%	97.41%	95.67%	94.98%
In execution efficiency	99.85%	97.38%	94.45%	74.09%	53.47%
Scalability for computation tasks	100.00%	85.06%	67.96%	65.18%	57.40%
IPC scalability	100.00%	87.34%	73.50%	73.34%	64.75%
Instruction scalability	100.00%	99.80%	99.54%	99.29%	99.03%
Frequency scalability	100.00%	97.58%	92.88%	89.51%	89.52%

Table 2: Analysis done on Wed Sep 21 08:21:57 PM CEST 2022, paco1208

Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	2630.0	10520.0	21040.0	31560.0	42080.0
LB (number of explicit tasks executed)	1.0	0.91	0.95	0.87	0.83
LB (time executing explicit tasks)	1.0	0.98	0.98	0.98	0.97
Time per explicit task (average us)	505.96	148.7	93.05	64.68	55.08
Overhead per explicit task (synch %)	0.01	4.5	7.96	36.77	90.18
Overhead per explicit task (sched %)	0.13	0.33	0.71	4.28	6.69
Number of taskwait/taskgroup (total)	263.0	263.0	263.0	263.0	263.0

Table 3: Analysis done on Wed Sep 21 08:21:57 PM CEST 2022, paco1208