PACO Lab1 - Informe

Ixent Cornella, Arnau Roca (PACO1201)

SESSIÓ 1

L'arquitectura BOADA consta de nodes amb diferents components. En el nostre cas, el node boada11 conté 2 sockets, dividint la memòria principal equitativament. Després, es consta amb quatre nivells de caché, dels quals 3 són individuals per cada *core*, mentre que la de major nivell es comparteix entre els 10 cores que conté cada socket. Arriba fins a 3,2 GHz de freqüència, amb un mínim de 1 GHz.

	Node utilitzat: boada-11
Number of sockets per node	2
Number of cores per socket	10
Number of threads per core	2
Maximum core frequency	3,2 GHz
L1-I cache size (per-core)	32KB
L1-D cache size (per-core)	32KB
L2 cache size (per-core)	1024KB
Last-level cache size (per-socket)	14 MB
Main memory size (per socket)	47GB
Main memory size (per node)	94GB

Figura 1: Taula mostrant les dades del node boada-11.

Ara es comparará l'execució serial de un programa que calcula pi emprant el mètode d'aproximació amb integrals, d'una banda la versió interactiva, i de l'altra, la versió amb cues.

#threads	Timing information interactive				Timing information queue			
	user (s)	system (s)	elapsed (s)	% of CPU	user (s)	system (s)	elapsed (s)	% of CPU
1	2.37	0.00	2.37	99	0.68	0.00	0.70	98
4	2.38	0.00	1.19	199	0.69	0.00	0.19	359
8	2.41	0.05	1.23	199	0.74	0.00	0.11	651
16	2.42	0.09	1.26	199	0.80	0.00	0.08	1004
20	2.43	0.08	1.26	198	0.84	0.00	0.06	1323

Figura 2: Taula comparant el temps i rendiment de l'execució en funció dels threads per a execució interactiva i no interactiva.

Veient la taula resultant, queda clar que l'execució no interactiva (a la dreta) és més ràpida que la interactiva, això és degut a que el temps d'usuari i sistema es redueixen significativament. A més, a mesura que s'incrementen els threads, sembla que mentre que la versió interactiva arriba a un límit d'ús de CPU, la no interactiva aprofita fins a un 1323% d'ús de la CPU, reduint així el temps emprat fins a 60 ms.

Pel que fa a la versió escalable de l'execució de *submit-strong-omp.sh*, es pot veure clarament com quan augmentem el nombre de threads, fins a 40, hi ha un patró: En els primers 20 threads, el temps d'execució es disminueix progressivament, fins que arriba a 21, en aquest cas, el temps d'execució augmenta, i el Speed-Up també és redueix. Concretament, aquí estan els resultats:

#threads	Speedup		
1	0.99569428402073584120	**2**	1.96747422680412371134
3	2.88058262196655877370	**4**	3.83567349080778700193
5	4.48390591552237247428	**6**	5.34559035207158140439
7	6.14291462138891124165	**8**	6.95745779704098269349
9	7.57279691375034444750	**10**	8.32899330811754437009
11	9.08221079487620130342	**12**	9.69768229681280523953
13	10.46060384597816653724	**14**	11.22030964201724588450
15	11.87749809833344858585	**16**	12.42457999529811743855
17	12.58226503552853270822	**18**	13.34476730634760313883
19	14.00383196428935406942	**20**	14.55104201965435445611
21	8.32818560899922420480	**22**	8.66394279877425944841
23	8.99917479861156591787	**24**	9.35463754697456565546
25	9.68155684572459275125	**26**	10.12529843487487841542
27	10.46984959083220310571	**28**	10.83081628148942207648
29	11.09761100970779692774	**30**	11.36545905707196029776
31	11.73227459016393442622	**32**	12.05971563981042654028
33	12.40103245370203241760	**34**	12.80385396671574200040
35	13.03882942382145297198	**36**	13.34684124640609215945
37	13.76837675350701402805	**38**	14.02384111367394011144
39	14.45612927660648907966	**40**	14.57604752307202715604

Pel que fa a escalabilitat dèbil, tant la mida del problema com els processadors disponibles augmenten progressivament. Per tant, el temps d'execució serà similar sempre, i nosaltres ho hem comprovat experimentalment veient que el temps d'execució sempre estava a prop de 0.08s, tot i que augmentava quan més gran es feia el problema.

SESSIÓ 2

Ara mirarem com es pot comparar el paral·lelisme de gra fi amb el de gra “gros”. És a dir, com es pot comparar el repartiment de tasques entre els processadors per tal d’optimitzar el temps d’execució.

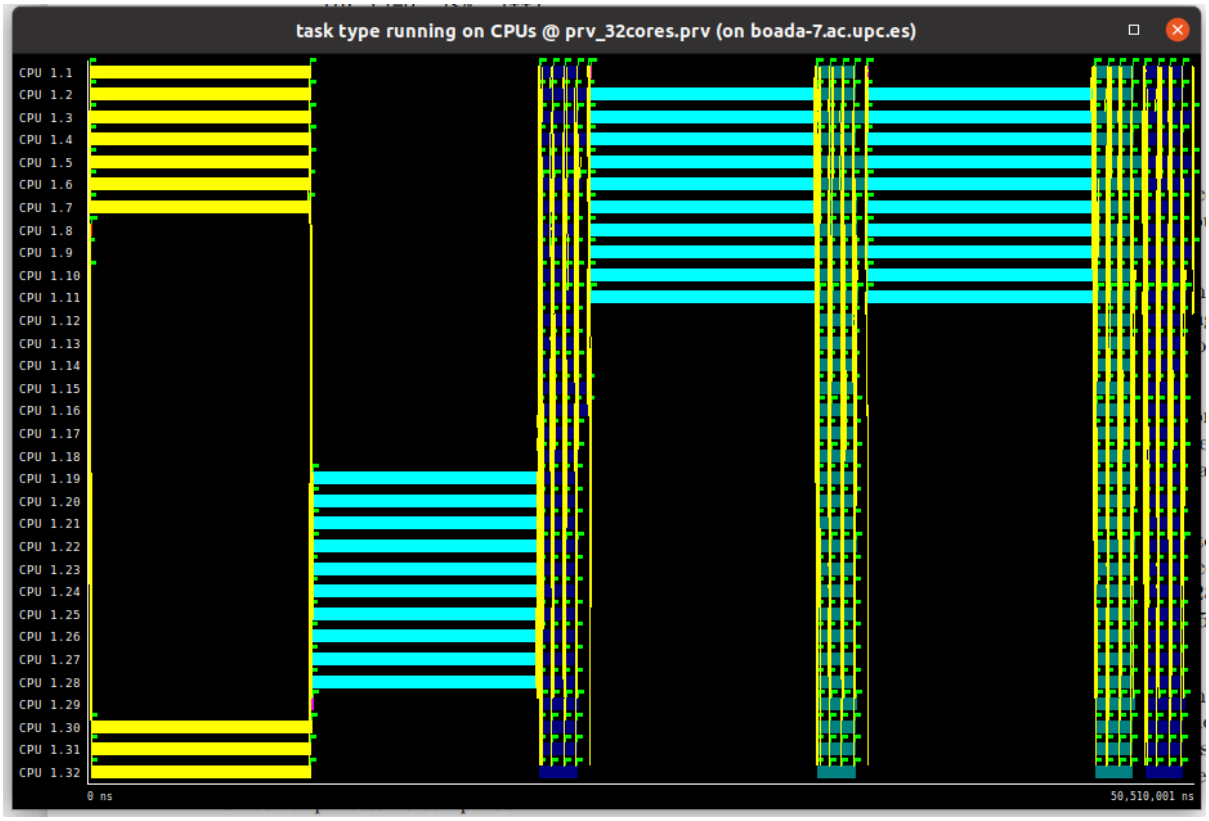
La següent taula mostra com segons el gra (seq = seqüencial, v5 = gra molt fi [creant cada tasca per iteracions de cada bucle]), el programa (una Transformada de Fourier Ràpida) s’executa més o menys ràpid. També s’indica el paral·lelisme de cada versió.

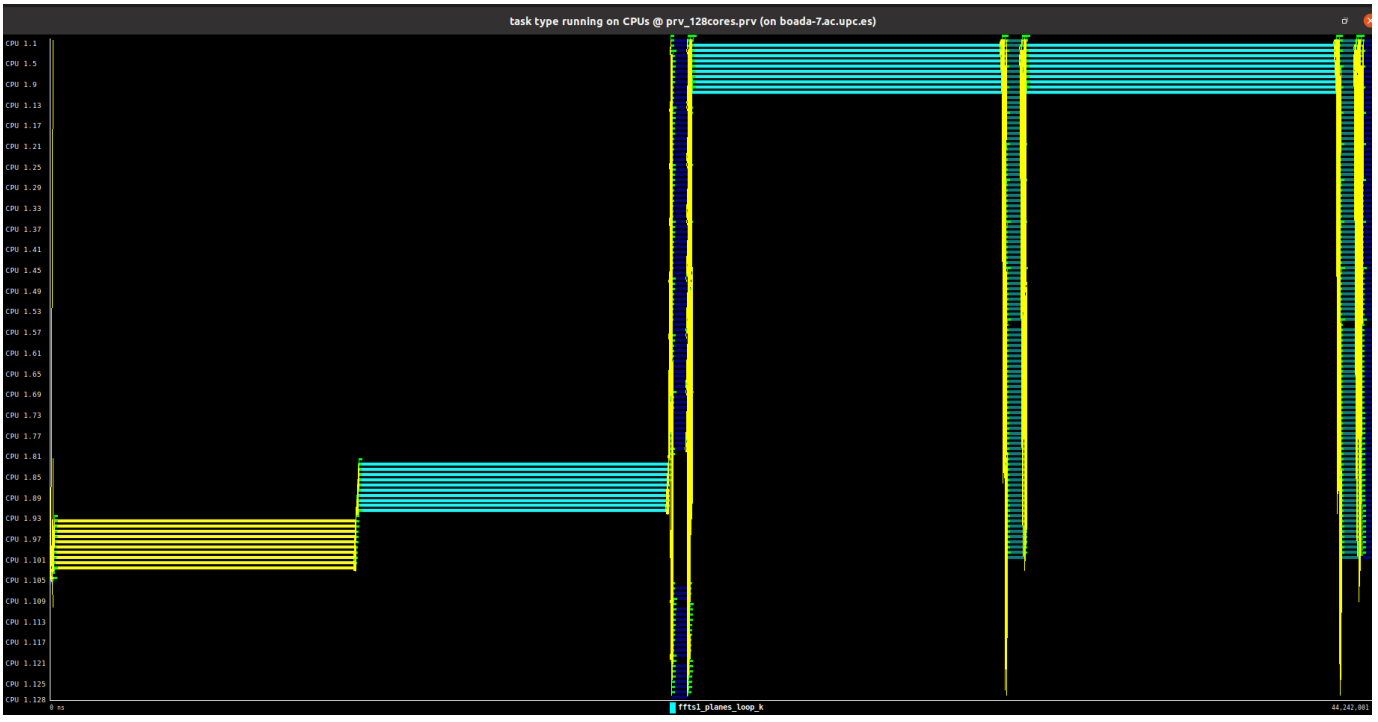
Version	T1 (unitats de temps)	T∞ (unitats de temps)	Parallelism
seq	639780	639760	1.00003
v1	639780	639760	1.00003
v2	639780	361337	1.77059
v3	639780	154971	4.12838
v4	639780	64750	9.88077
v5	639780	44242	14.4609

Figura 3: Taula comparant el temps d’execució de cada versió del programa, a cada qual de més gra fi. T1 representa el temps d’execució del programa amb un processador, T∞ amb “infinits” processadors.

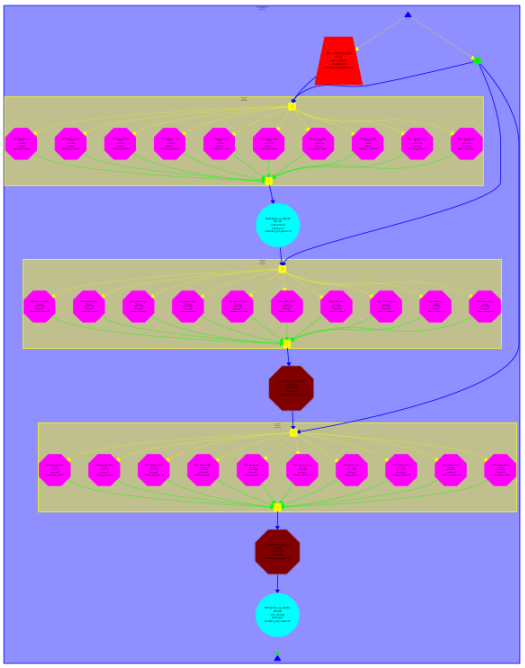
Com es pot observar, T1 és constant. Això és degut a que tot i augmentar el gra del programa, amb un sol procesador no es pot tenir paral·lelisme. Per tant, té sentit que sempre sigui el mateix valor. D’altra banda, quan més gra hi ha, menys dura l’execució del programa, augmentant també el paral·lelisme.

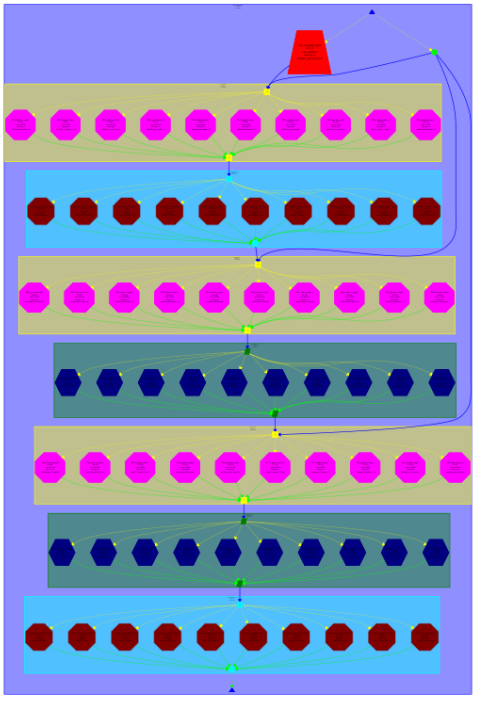
L’escalabilitat també té un rol en aquesta execució: Quants més processadors (**T∞**), més escalat fort hi ha, reduint així també l’execució del programa. No hi ha escalabilitat dèbil ja que utilitzem sempre el mateix problema.

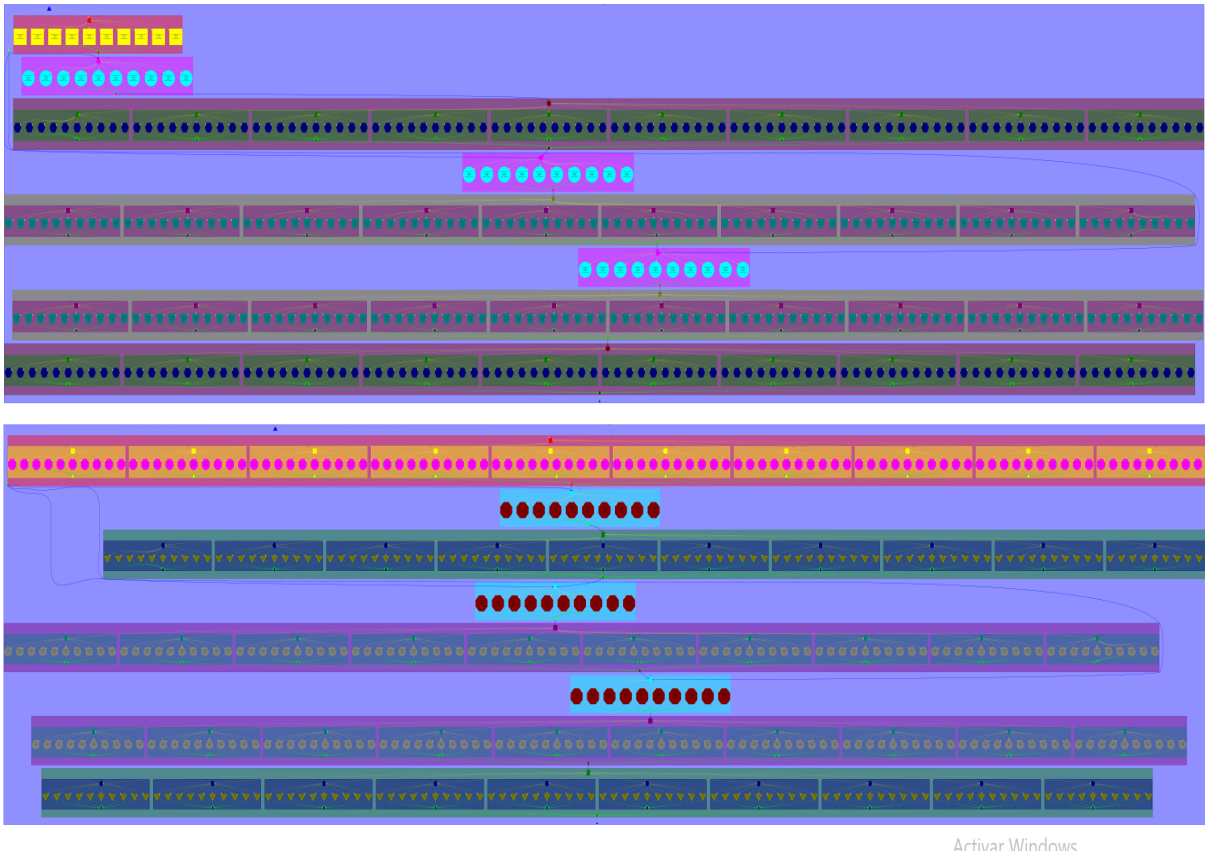




Diagrames que mostren el temps d'execució i la repartició de tasques de la versió 4 i 5.







En ordre: graf de dependències de la versió 1, 2, 3, 4 i 5, respectivament.

```
void init_complex_grid(fftwf_complex in_fftw[][N][N]) {
  int k,j,i;

  for (k = 0; k < N; k++) {
    tareador_start_task("init_complex_grid_loop_k");
    for (j = 0; j < N; j++) {
        tareador_start_task("init_complex_grid_loop_j"); //aixo ho he posat jo
      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_loop_j");  //aixo tmb
    }
    tareador_end_task("init_complex_grid_loop_k");
  }
}
```

En aquesta última imatge veiem com hem modificat el codi de la tasca init_complex_grid per augmentar el paral·lelisme, introduint dos creacions de tasques per fer més fi el gra.

SESSIÓ 3

Versió inicial de 3dfft:

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	1.34	0.78	0.81	1.27	1.48
Speedup	1.00	1.72	1.64	1.06	0.90
Efficiency	1.00	0.43	0.21	0.09	0.06

Table 1: Analysis done on Wed Sep 21 12:56:44 PM CEST 2022, paco1201

Overview of the Efficiency metrics in parallel fraction, ϕ=83.00%					
Number of processors	1	4	8	12	16
Global efficiency	98.80%	48.88%	23.94%	8.78%	5.55%
Parallelization strategy efficiency	98.80%	89.21%	86.96%	73.34%	56.10%
Load balancing	100.00%	98.30%	97.79%	98.05%	97.19%
In execution efficiency	98.80%	90.76%	88.92%	74.80%	57.73%
Scalability for computation tasks	100.00%	54.79%	27.54%	11.97%	9.89%
IPC scalability	100.00%	69.25%	51.01%	39.05%	40.58%
Instruction scalability	100.00%	98.34%	96.20%	94.13%	92.16%
Frequency scalability	100.00%	80.46%	56.11%	32.57%	26.45%

Table 2: Analysis done on Wed Sep 21 12:56:44 PM CEST 2022, paco1201

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	0.91	0.76	0.87	0.74
LB (time executing explicit tasks)	1.0	0.99	0.99	0.98	0.98
Time per explicit task (average us)	61.08	27.88	27.75	42.55	38.63
Overhead per explicit task (synch %)	0.17	10.56	12.93	32.62	72.53
Overhead per explicit task (sched %)	1.04	1.51	2.03	3.71	5.68
Number of taskwait/taskgroup (total)	1792.0	1792.0	1792.0	1792.0	1792.0

Table 3: Analysis done on Wed Sep 21 12:56:44 PM CEST 2022, paco1201

Taules mostrant les mètriques de la primera execució.

Is the scalability appropriate?

No, l'escalabilitat no és apropiada ja que disminueix l'eficiència segons augmentem el número de processadors, això és degut en part al overhead.

Is the overhead due to synchronization neglegible?

No, ja que es produeix molt overhead al haver molts processadors, fins a 72% de sincronització com es pot veure a la taula 3.

Is this overhead affecting the execution time per explicit task?

Sí, és el factor que més retrasa l'execució.

Which is the parallel fraction (φ) for this version of the program?

φ = 83%.

Is the efficiency for the parallel regions appropriate?

No, disminueix a mesura que augmenta el número de processadors.

Which is the factor that is negatively influencing the most?

Té dos problemes, l'estratègia de paralelització i l'escalabilitat (més gran aquest). Els overheads fan que realment sigui més lent el programa quant més gra fi és, per tant disminueix l'eficiència quan més paral·lelisme hi ha.

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.

Mirant les dues finestres de línies de temps podem veure aquests dos factors: 1) hi ha una funció que no està paral·lelitzada i 2) hi ha un estat de thread que predomina al llarg de la finestra de la línia de temps. Per tant, realment depèn del número de threads, ja que l'overhead es el temps afegit per la sincronització quan hi ha paral·lelisme, a més de la mala paralelització.

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?

Tenim granularitat de tasques fina (tot i que no el màxim de fina ja que es podria afegir més paral·lelisme als bucles), tenim segmentació a la part del bucle més interna, fent que hi hagin més tasques que si l'instrucció estigués fora d'aquest bucle.

Versió 2 de 3dfft:

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?

La eficiencia ha millorat molt, de fet s'ha reduït el temps gairebé 4 cops amb 16 processadors. Això implica que no hi ha cap "slowdown", ja que el temps d'execució millora amb l'ús de més processadors. L'overhead també s'ha reduït dràsticament, passant de 72.5% a un 3%. Això era el principal factor que retrassava l'execució. Tot i això, la fracció paralela φ continua més o menys igual, ja que la part paral·lelitzable és igual, amb una petita diferència en T_1 (creiem que ve donada ja que la part de crear tasques per cada loop és redueix al ficar la paralelitzacio fora del bucle intern), però tot i això, la part no paral·lelitzable és pràcticament la mateixa (init programa).

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	1.28	0.55	0.45	0.41	0.38
Speedup	1.00	2.33	2.84	3.17	3.35
Efficiency	1.00	0.58	0.35	0.26	0.21

Table 1: Analysis done on Wed Sep 21 02:09:36 PM CEST 2022, paco1201

Overview of the Efficiency metrics in parallel fraction, ϕ=82.46%					
Number of processors	1	4	8	12	16
Global efficiency	99.96%	79.12%	61.31%	53.11%	46.57%
Parallelization strategy efficiency	99.96%	97.50%	96.71%	96.60%	96.87%
Load balancing	100.00%	97.90%	97.58%	97.81%	98.32%
In execution efficiency	99.96%	99.60%	99.11%	98.77%	98.52%
Scalability for computation tasks	100.00%	81.14%	63.40%	54.98%	48.08%
IPC scalability	100.00%	81.29%	67.38%	60.57%	53.20%
Instruction scalability	100.00%	99.99%	99.98%	99.97%	99.96%
Frequency scalability	100.00%	99.83%	94.11%	90.80%	90.41%

Table 2: Analysis done on Wed Sep 21 02:09:36 PM CEST 2022, paco1201

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	0.97	0.92	0.93	0.78
LB (time executing explicit tasks)	1.0	0.98	0.98	0.99	0.99
Time per explicit task (average us)	15108.75	4654.84	2978.69	2289.63	1963.71
Overhead per explicit task (synch %)	0.01	2.49	3.27	3.34	3.01
Overhead per explicit task (sched %)	0.03	0.03	0.04	0.04	0.05
Number of taskwait/taskgroup (total)	7.0	7.0	7.0	7.0	7.0

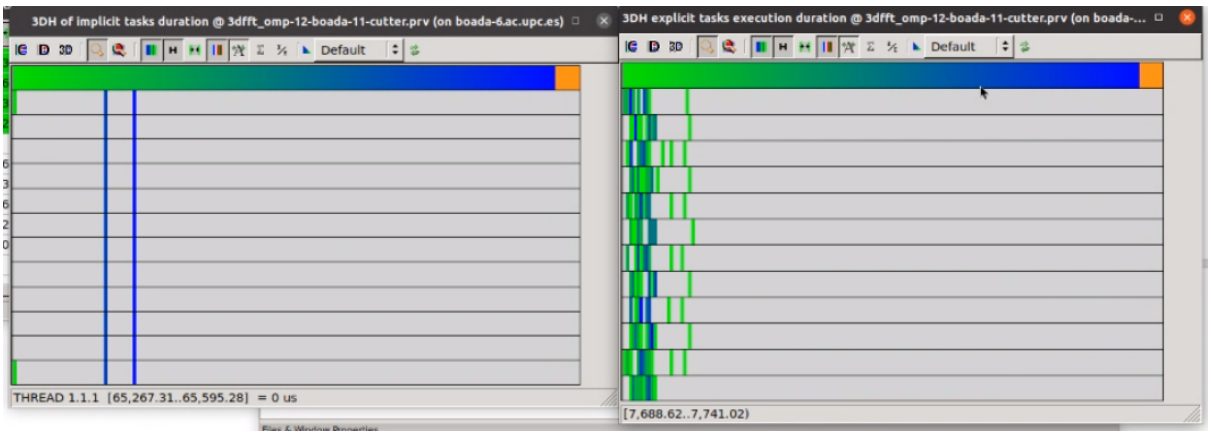
Table 3: Analysis done on Wed Sep 21 02:09:36 PM CEST 2022, paco1201

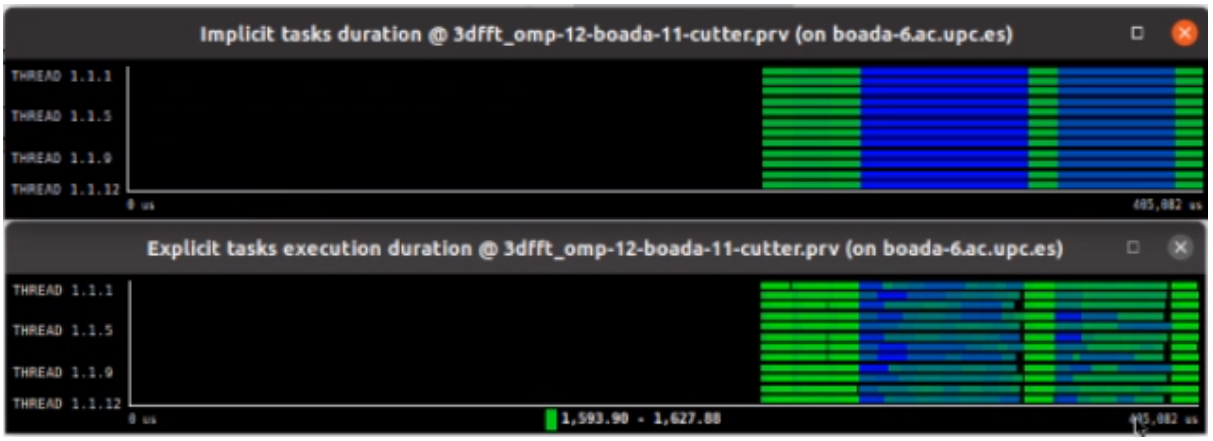
Do you observe any major difference on the duration of the implicit and explicit tasks for the initial and optimized versions? What is the function that is limiting the maximum speedup that you can achieve? Which functions in the program are or not parallelized?

No, la duració és la mateixa en els dos.

La funció inicial ens està limitant el speedup, per la seva sequencialitat.

La única funció no paraleliztada és la de inicialitzar, que també és bastant llarga, per tant provoca més temps d'execució.





Histogrames i duració de les tasques explícites i implícites.

Versió final de 3dfft:

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.

Sí! El speed-up i l'eficiència han millorat, passant de un 3.17 a un 5.38 (speedup). El nou valor de φ és de 99.94%. Sense dubte, amb molta diferència de les anteriors versions.

Versió	φ	S_{12} Ideal	T_1	T_{12}	S_{12} Real
initial version in 3dfft omp.c	83%	1.1	1.34	1.27	1.06
new version with reduced parallelisation overheads	82.46%	3.2	1.29	0.41	3.17
final version with improved φ	99.94%	5.4	1.29	0.24	5.38

En aquesta taula es pot veure com la versió paralelitzada sense tant overhead i amb la funció inicial també paralelitzada és superior a les demés, ja que s'executa gairebé 5 cops més ràpid que la primera versió. Això és degut a que hem escalat el programa de forma forta i a més, hem pogut veure com en la versió 2 no ens ha servit tant, però sí en la final.

Comparació de les 3 versions de 3dfft amb la mateixa escala de temps. (De dalt a abaix: versió final, 2, inicial).

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	1.29	0.41	0.25	0.24	0.27
Speedup	1.00	3.19	5.13	5.38	4.73
Efficiency	1.00	0.80	0.64	0.45	0.30

Table 1: Analysis done on Thu Sep 22 05:18:34 PM CEST 2022, paco1201

Overview of the Efficiency metrics in parallel fraction, ϕ=99.94%					
Number of processors	1	4	8	12	16
Global efficiency	99.85%	79.60%	64.11%	44.84%	29.53%
Parallelization strategy efficiency	99.85%	95.20%	91.94%	71.63%	52.68%
Load balancing	100.00%	97.50%	96.50%	95.75%	95.33%
In execution efficiency	99.85%	97.64%	95.27%	74.81%	55.26%
Scalability for computation tasks	100.00%	83.61%	69.73%	62.60%	56.06%
IPC scalability	100.00%	86.00%	76.05%	70.36%	63.05%
Instruction scalability	100.00%	99.80%	99.55%	99.28%	99.03%
Frequency scalability	100.00%	97.42%	92.11%	89.61%	89.78%

Table 2: Analysis done on Thu Sep 22 05:18:34 PM CEST 2022, paco1201

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	0.92	0.95	0.87	0.87
LB (time executing explicit tasks)	1.0	1.0	0.98	0.97	0.97
Time per explicit task (average us)	490.84	146.75	87.98	65.33	54.71
Overhead per explicit task (synch %)	0.01	4.69	7.99	35.45	83.55
Overhead per explicit task (sched %)	0.13	0.34	0.75	4.11	6.24
Number of taskwait/taskgroup (total)	263.0	263.0	263.0	263.0	263.0

Table 3: Analysis done on Thu Sep 22 05:18:34 PM CEST 2022, paco1201

Taules mostrant els resultats de l'execució de l'última versió.