PACO Lab3 - Informe

Ixent Cornella, Arnau Roca (PACO1201)

SESSIÓ 1

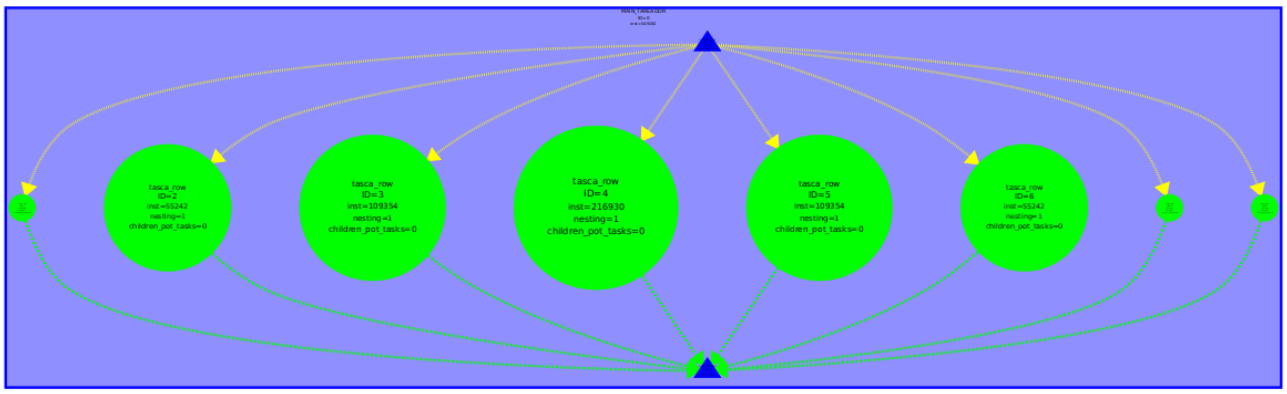


Figura 1: Gràfic de tasques de mandel.c inicial

Les característiques més importants són que només hi ha 8 tasques, degut a configuració del tareador, i que estan mal distribuïdes per que hi ha zones de càlcul que requereixen moltes més instruccions.

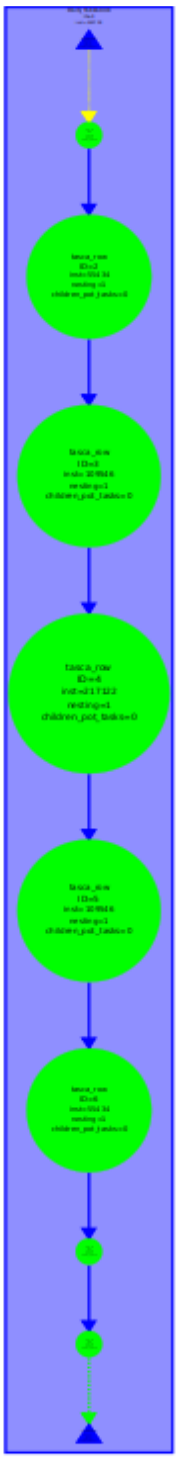


Figura 2: Gràfic de tasques de mandel.c, amb l'opció -d

Podem observar que amb l'opció -d no hi ha paral·lelisme, de manera que hi ha una dependència seqüencial de les tasques. També hi ha 8 tasques com abans. La variable "output2display" està activada, hi ha més zones del codi que s'han d'executar, causant aquestes dependències.

```
if (output2display) {
                /* Scale color and display point  */
                long  color  =  (long)  ((k-1)  *  scale_color)  +
min_color;
                if (setup_return == EXIT_SUCCESS) {
                XSetForeground (display, gc, color);
                XDrawPoint (display, win, gc, col, row);
                }
           }
```

aquesta és la part del codi que s'executa si la variable output2display està activada. Per veure exactament la dependència que ens fa que el codi sigui així, ens hem de fixar bé en el codi. El XSetForeground el que fa és posar el color a un pixel de la pantalla i el XDrawPoint el mostra per pantalla. Aquí podem veure clarament la dependència ja que necessita el punt pintat abans de poder mostrar-lo.

Per solucionar això podriem usar el #pragma omp critical, però no seria molt òptim ja que ens podria crear overheads.

Figura 3: Gràfic de tasques de mandel.c amb l'opció -h

Passa el mateix que abans, però amb la variable "output2histogram". La variable "output2histogram" està activada, hi ha més zones del codi que s'han d'executar, causant aquestes dependències

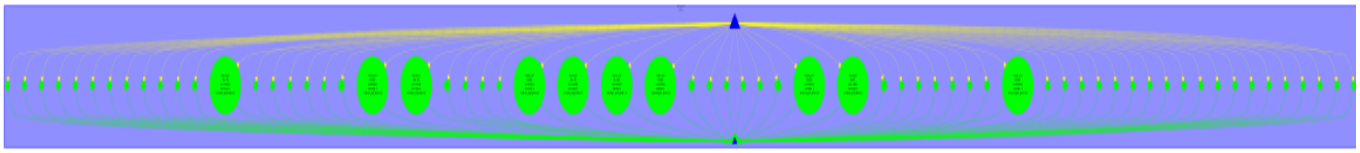


Figura 4: Gràfic de tasques de mandel.c sense opcions, però amb files.

Com podem veure, el gràfic és similar al de la figura 1, però amb més tasques, cosa que té sentit ja que ara ho executem per files, i conseqüentment, moltes més tasques. També podem veure que tenim una granularitat molt irregular, ja que hi ha tasques molt petites i d'altres de molt grans

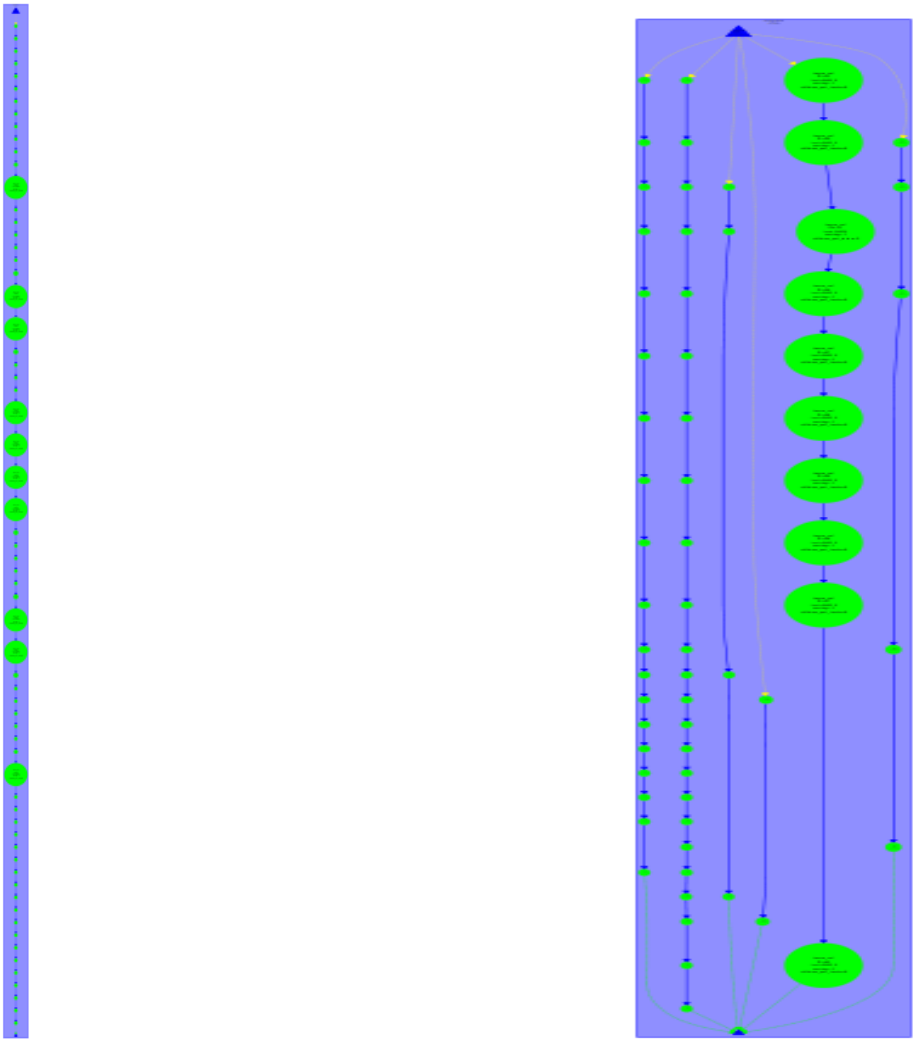


Figura 5 i 6: Gràfic de tasques de mandel.c amb l'opció -d i -h, respectivament

Amb les opcions -d i -h, podem observar que l'estil també és similar a les figures 2 i 3, però amb moltes més tasques, respectant la coherencia amb la figura anterior.

Per poder solucionar les dependències que ja hem vist en l'apartat anterior hem afegit, com ja deiem allà les dues instruccions de #pragma omp critical, just abans de les dues funcions (XSet i XDraw).

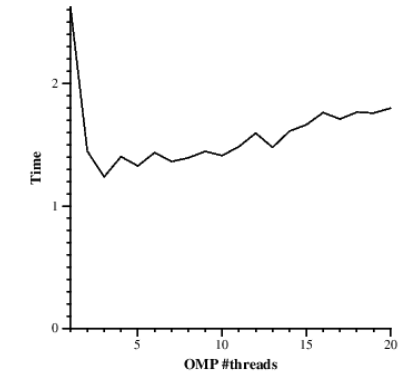
Primerament fem obtenim els resultats amb la versió que ens surt al document, tenint el task com ens demanen. Llavors un cop fet els dos submits obtenim les següent taules i gràfiques (el pdf de dins del extrae i el postscript):

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	0.52	0.34	0.30	0.32	0.37
Speedup	1.00	1.54	1.75	1.65	1.43
Efficiency	1.00	0.39	0.22	0.14	0.09

Table 1: Analysis done on Wed Nov 9 06:58:55 PM CET 2022, paco1201

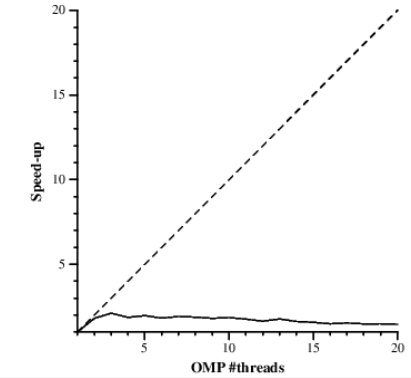
Overview of the Efficiency metrics in parallel fraction, ϕ=99.90%					
Number of processors	1	4	8	12	16
Global efficiency	94.90%	36.61%	20.78%	13.05%	8.47%
Parallelization strategy efficiency	94.90%	45.19%	28.64%	18.65%	13.52%
Load balancing	100.00%	90.24%	54.51%	37.87%	21.14%
In execution efficiency	94.90%	50.07%	52.53%	49.23%	63.96%
Scalability for computation tasks	100.00%	81.02%	72.56%	69.99%	62.64%
IPC scalability	100.00%	76.27%	71.88%	70.80%	63.27%
Instruction scalability	100.00%	105.61%	106.41%	106.52%	105.75%
Frequency scalability	100.00%	100.59%	94.86%	92.81%	93.63%

Table 2: Analysis done on Wed Nov 9 06:58:55 PM CET 2022, paco1201



Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	102400.0	102400.0	102400.0	102400.0	102400.0
LB (number of explicit tasks executed)	1.0	0.78	0.81	0.86	0.91
LB (time executing explicit tasks)	1.0	0.88	0.88	0.88	0.94
Time per explicit task (average us)	4.36	5.13	5.39	5.44	5.44
Overhead per explicit task (synch %)	0.0	109.27	283.05	526.48	884.47
Overhead per explicit task (sched %)	5.97	32.22	25.67	28.89	23.66
Number of taskwait/taskgroup (total)	0.0	0.0	0.0	0.0	0.0

Table 3: Analysis done on Wed Nov 9 06:58:55 PM CET 2022, paco1201



Aquestes son les taules i gràfiques que ens surten, resultants de utilitzar el task. Podem veure que l'escalabilitat no és gens apropiada, en la taula de la dreta, ja que apart que puja molt poc el speedup en augmentar els processadors, arribat als 8 threads no augmenta més, sinó que disminueix. Per tant podem afirmar que ni el speedup ni la escalabilitat son adequats.

Ara obrim el paraver per veure més detalls sobre l'execució.

Histogram task execution @ mandel-omp-12-boada-11-cutter.prv (on boa...

Default

	104 (mandel-omp.c, mandel-omp)
THREAD 1.1.1	9,950
THREAD 1.1.2	9,138
THREAD 1.1.3	9,712
THREAD 1.1.4	9,177
THREAD 1.1.5	9,953
THREAD 1.1.6	8,962
THREAD 1.1.7	9,583
THREAD 1.1.8	99
THREAD 1.1.9	9,375
THREAD 1.1.10	8,757
THREAD 1.1.11	9,199
THREAD 1.1.12	8,495
Total	102,400
Average	8,533.33
Maximum	9,953
Minimum	99
StDev	2,579.05
Avg/Max	0.86

Histogram task instantiation @ mandel-omp-12-boada-11-cu

Default

	104 (mandel-omp.c, mandel-omp)
THREAD 1.1.1	-
THREAD 1.1.2	-
THREAD 1.1.3	-
THREAD 1.1.4	-
THREAD 1.1.5	-
THREAD 1.1.6	-
THREAD 1.1.7	-
THREAD 1.1.8	102,400
THREAD 1.1.9	-
THREAD 1.1.10	-
THREAD 1.1.11	-
THREAD 1.1.12	-
Total	102,400
Average	102,400
Maximum	102,400
Minimum	102,400
StDev	0
Avg/Max	1

Com podem veure en les imatges, tenim un thread, en aquesta cas el 8, que crea totes les tasques. Llavors això comporta un desbalanceig bastant gran de la càrrega però que es assumible degut al nombre de tasques que tenim. Podem veure que la

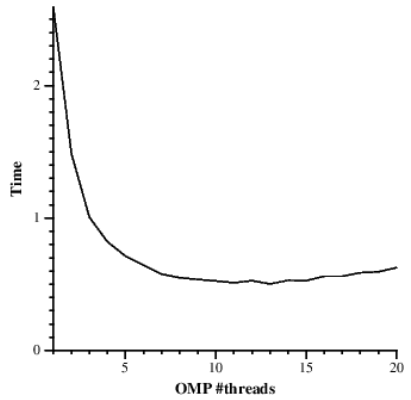
mitjana tot i ser més petita està bastant prop de tots els valors excepte del creador. Amb aquestes dues imatges podem veure molt exemplificat.

Ara utilitzarem el **TaskLoop** per optimitzar i compararem amb el **Task:**

Primer que tot editem el fitxer afegint un #pragma omp taskloop abans del segon for. obtenim les següent gràfiques i taules:

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	0.42	0.14	0.13	0.17	0.19
Speedup	1.00	2.92	3.12	2.47	2.20
Efficiency	1.00	0.73	0.39	0.21	0.14

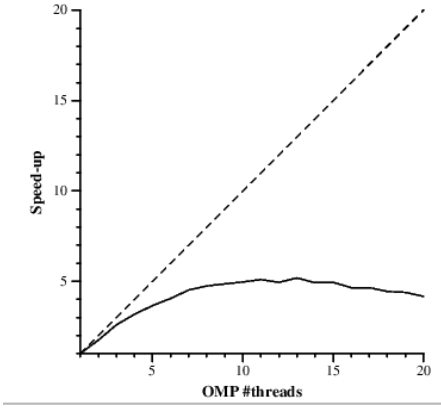
Table 1: Analysis done on Wed Nov 9 07:48:17 PM CET 2022, paco1201



paco1201
Min elapsed execution time
Generated by paco1201 on Wed Nov 9 07:48:01 PM CET 2022

Overview of the Efficiency metrics in parallel fraction, ϕ=99.93%					
Number of processors	1	4	8	12	16
Global efficiency	99.58%	72.83%	38.95%	20.54%	13.71%
Parallelization strategy efficiency	99.58%	75.51%	43.84%	24.08%	16.44%
Load balancing	100.00%	95.60%	95.19%	93.89%	95.36%
In execution efficiency	99.58%	78.99%	46.06%	25.65%	17.24%
Scalability for computation tasks	100.00%	96.45%	88.84%	85.30%	83.37%
IPC scalability	100.00%	98.50%	98.26%	98.03%	97.43%
Instruction scalability	100.00%	99.21%	98.18%	97.19%	96.19%
Frequency scalability	100.00%	98.70%	92.09%	89.54%	88.95%

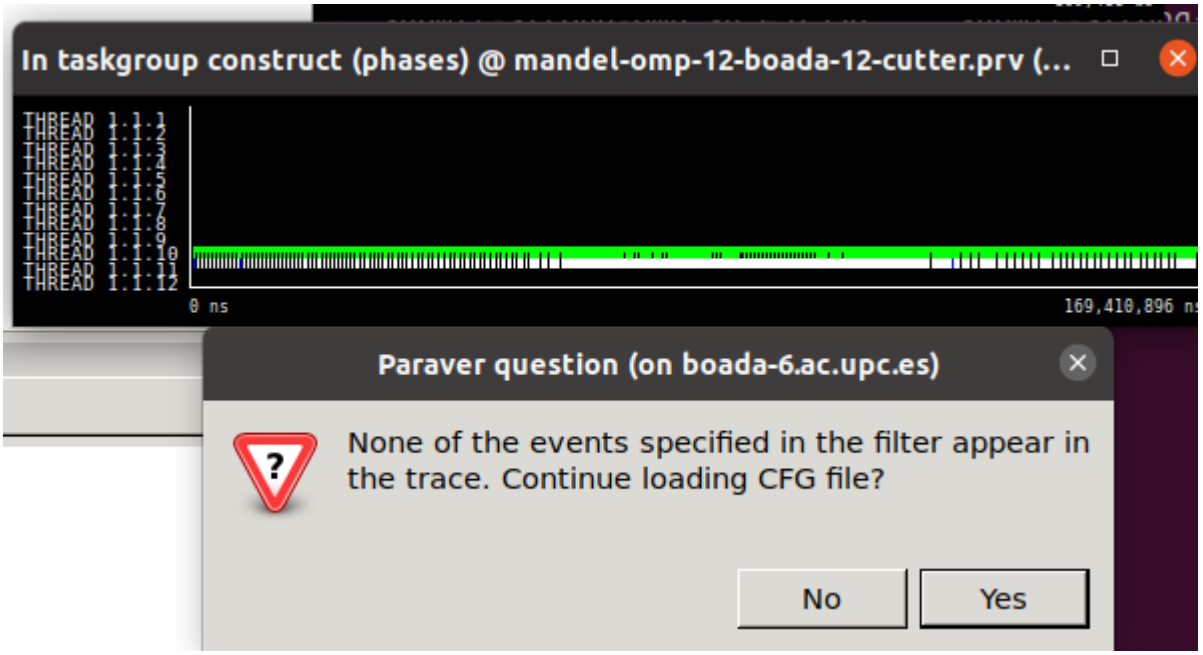
Table 2: Analysis done on Wed Nov 9 07:48:17 PM CET 2022, paco1201



Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	3200.0	12800.0	25600.0	38400.0	51200.0
LB (number of explicit tasks executed)	1.0	0.94	0.84	0.53	0.48
LB (time executing explicit tasks)	1.0	0.96	0.95	0.94	0.95
Time per explicit task (average us)	130.17	33.74	18.31	12.72	9.76
Overhead per explicit task (synch %)	0.06	28.33	113.83	289.71	476.48
Overhead per explicit task (sched %)	0.36	4.11	14.31	25.81	32.0
Number of taskwait/taskgroup (total)	320.0	320.0	320.0	320.0	320.0

Table 3: Analysis done on Wed Nov 9 07:48:17 PM CET 2022, paco1201

On podem veure que millorem els resultats anteriors del task però no molt. Ens passa com abans que l'escalabilitat no acaba de ser bona, ja que tot i pujar més ràpid, hi ha un punt on el speedup comença a baixar com abans. Cal destacar que la ditribució de càrrega millora molt respecte l'anterior tal com altres aspectes, que podem veure en les taules 2 i 3. Podem dir que la granularitat és millor.
Ara anem a analitzar amb el paraver:



amb això podem veure que tenim un taskgroup implicit dins del taskloop, no un taskwait (el missatge d'error es intentant carregar una opció de taskwait). Veiem que no ens beneficia en absolut, per tant anem a provar amb la clàusula nogroup per evitar el taskgroup implicit, ja creiem que no son necessaries.

Llavors un cop afegit el nogroup just darrera de la instrucció, obtenim aquests resultats:

Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	0.42	0.12	0.11	0.13	0.18
Speedup	1.00	3.60	3.90	3.11	2.26
Efficiency	1.00	0.90	0.49	0.26	0.14

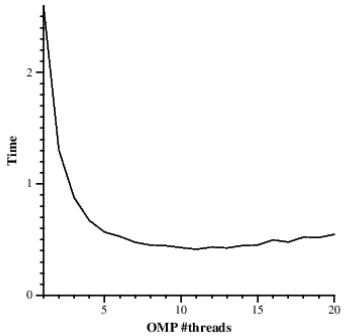
Table 1: Analysis done on Wed Nov 9 09:03:14 PM CET 2022, paco1201

Overview of the Efficiency metrics in parallel fraction, ϕ=99.88%					
Number of processors	1	4	8	12	16
Global efficiency	99.65%	89.72%	48.70%	25.87%	14.09%
Parallelization strategy efficiency	99.65%	93.48%	55.14%	30.45%	16.95%
Load balancing	100.00%	98.90%	98.51%	96.03%	95.77%
In execution efficiency	99.65%	94.53%	55.97%	31.71%	17.70%
Scalability for computation tasks	100.00%	95.97%	88.33%	84.94%	83.13%
IPC scalability	100.00%	98.85%	98.09%	97.86%	97.11%
Instruction scalability	100.00%	99.21%	98.18%	97.16%	96.21%
Frequency scalability	100.00%	97.86%	91.73%	89.34%	88.98%

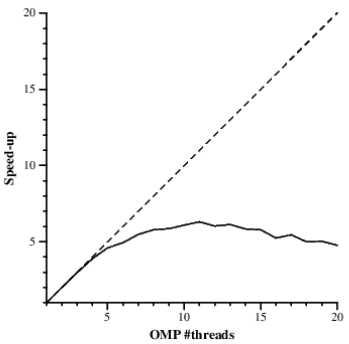
Table 2: Analysis done on Wed Nov 9 09:03:14 PM CET 2022, paco1201

Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	3200.0	12800.0	25600.0	38400.0	51200.0
LB (number of explicit tasks executed)	1.0	0.52	0.73	0.74	0.92
LB (time executing explicit tasks)	1.0	0.99	0.98	0.96	0.96
Time per explicit task (average us)	129.98	33.85	18.39	12.75	9.77
Overhead per explicit task (synch %)	0.0	4.55	70.12	208.02	458.07
Overhead per explicit task (sched %)	0.35	2.42	11.3	20.53	32.31
Number of taskwait/taskgroup (total)	0.0	0.0	0.0	0.0	0.0

Table 3: Analysis done on Wed Nov 9 09:03:14 PM CET 2022, paco1201



paco1201
Min elapsed execution time
Generated by paco1201 on Wed Nov 9 08:58:03 PM CET 2022



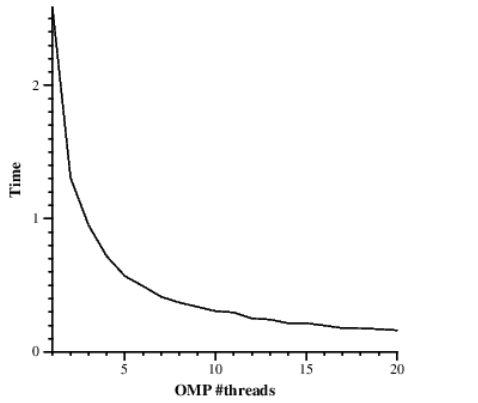
Podem veure que ens han millorat una mica els resultats, tot i que tenim el mateix problema que abans, ja que el speedup a partir d'un punt no puja més. Veiem també que els overheads han disminuït i l'eficiència ha millorat.

En un resum, podem veure que és millor aquesta opció, tot i que podem veure que l'estratègia de punts no és la millor que tenim.

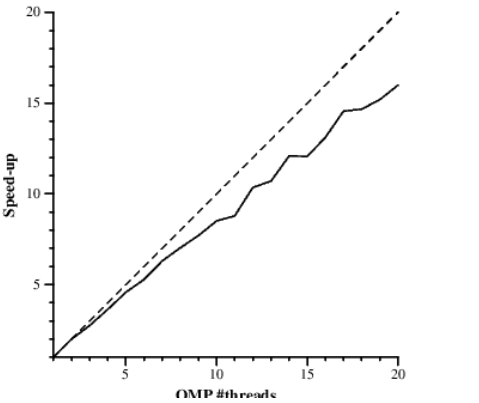
Ara mirarem l'estratègia **row**.

Degut a que ens demanen que ens fixem en l'experiència per aquest apartat utilitzarem

Posem la instrucció just abans del primer for, amb el firstprivate(row,col) i el no group i ens han millorat molt els resultats:



paco1201
Min elapsed execution time
Generated by paco1201 on Wed Nov 9 08:20:37 PM CET 2022



Overview of whole program execution metrics					
Number of processors	1	4	8	12	16
Elapsed time (sec)	0.42	0.12	0.06	0.04	0.03

Overview of the Efficiency metrics in parallel fraction, ϕ=99.87%					
Number of processors	1	4	8	12	16
Global efficiency	99.97%	89.00%	84.87%	81.18%	79.58%
Parallelization strategy efficiency	99.97%	90.47%	91.95%	90.02%	88.23%
Load balancing	100.00%	90.52%	92.27%	90.38%	89.16%
In execution efficiency	99.97%	99.94%	99.66%	99.60%	98.97%
Scalability for computation tasks	100.00%	98.38%	92.30%	90.18%	90.19%
IPC scalability	100.00%	99.55%	99.51%	99.52%	99.62%
Instruction scalability	100.00%	100.00%	99.99%	99.99%	99.98%
Frequency scalability	100.00%	98.83%	92.76%	90.62%	90.56%

Table 2: Analysis done on Wed Nov 9 08:20:44 PM CET 2022, paco1201

Statistics about explicit tasks in parallel fraction					
Number of processors	1	4	8	12	16
Number of explicit tasks executed (total)	10.0	40.0	80.0	120.0	160.0
LB (number of explicit tasks executed)	1.0	0.59	0.32	0.19	0.17
LB (time executing explicit tasks)	1.0	0.91	0.92	0.9	0.89
Time per explicit task (average us)	41484.44	10541.22	5616.68	3832.78	2873.95
Overhead per explicit task (synch %)	0.0	10.51	8.69	10.98	13.18
Overhead per explicit task (sched %)	0.02	0.02	0.05	0.09	0.13
Number of taskwait/taskgroup (total)	0.0	0.0	0.0	0.0	0.0

Table 3: Analysis done on Wed Nov 9 08:20:44 PM CET 2022, paco1201

Podem veure la millora tan sols mirar les gràfiques, hem arreglat l'escalabilitat i el speedup fent que sigui molt més apropiat. Podem veure que la distribució de càrrega ha baixat però tot i així l'eficiència del programa ha pujat com podem veure en la segona taula del modelfactor. Aquesta seria la millor opció. També veiem que hi ha molts menys overheads cosa que també ajuda a aquesta millora.

Per tant, podem veure que la millor opció seria fer el taskloop sense el taskgroup implícit ja que no ens aporta res, algunes altres proves que podriem fer seria fer una modificació amb un taskwait per provar-ho o un taskgroup posat de diferent forma, podria aportar resultats diferents però no crec que millors.

Histogram task instantiation @ mandel-omp-12-boada-11-cutter.prv (on ...

	102 (mandel-omp.c, mandel-omp)
THREAD 1.1.1	1
THREAD 1.1.2	-
THREAD 1.1.3	-
THREAD 1.1.4	-
THREAD 1.1.5	-
THREAD 1.1.6	-
THREAD 1.1.7	-
THREAD 1.1.8	-
THREAD 1.1.9	-
THREAD 1.1.10	-
THREAD 1.1.11	-
THREAD 1.1.12	-
Total	1
Average	1
Maximum	1
Minimum	1
StDev	0
Avg/Max	1

Histogram task execution @ mandel-omp-12-boada-11-cutter.pr

	102 (mandel-omp.c, mandel-omp)
THREAD 1.1.1	53
THREAD 1.1.2	5
THREAD 1.1.3	5
THREAD 1.1.4	6
THREAD 1.1.5	7
THREAD 1.1.6	7
THREAD 1.1.7	8
THREAD 1.1.8	6
THREAD 1.1.9	4
THREAD 1.1.10	6
THREAD 1.1.11	7
THREAD 1.1.12	6
Total	120
Average	10
Maximum	53
Minimum	4
StDev	13.01
Avg/Max	0.19

Aquí veiem que hi ha un thread el 1.1.1 que crea la tasca, i llavors s'executen amb una càrrega balancejada tots menys aquest que l'ha creat.