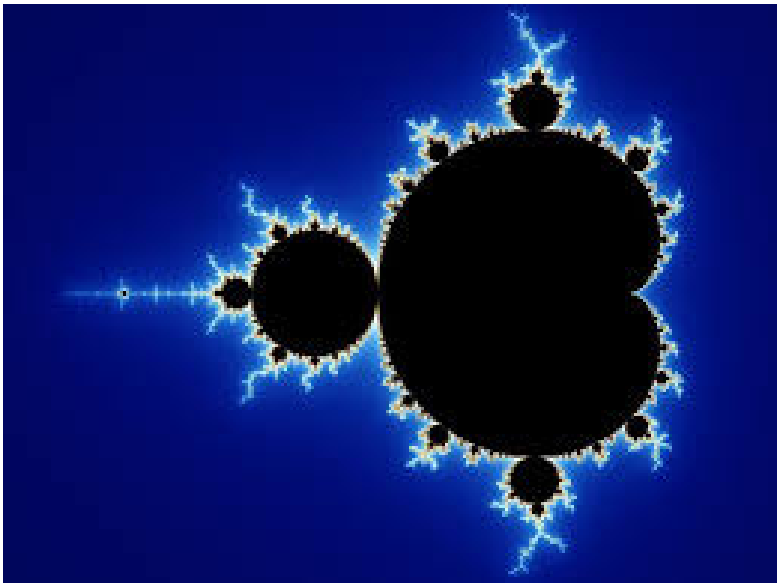
PARALLELISM LABORATORY 4

David Morais
david.morais

Laura van Dinteren
laura.van.dinteren



	Number of threads					
Version	1	4	8	12	16	20
Iterative: Tile (provided, but should be completed	3.05	0.900	0.69	0.70	0.7	0.7
Iterative: Finer grain	3.06	0.77	0.41	0.34	0.26	0.21
Recursive: Leaf (provided, but should be completed)	1.61	1.24	1.33	1.37	1.37	1.37
Recursive: Tree (cutoff 5)	1.6	0.42	0.24	0.18	0.15	0.13

TILE

Model Factor tables

Overview of whole program execution metrics									
Number of Processors	1	2	4	6	8	10	12	14	16
Elapsed time (sec)	3.06	1.72	0.90	0.74	0.69	0.71	0.70	0.71	0.71
Speedup	1.00	1.77	3.39	4.12	4.41	4.30	4.35	4.34	4.34
Efficiency	1.00	0.89	0.85	0.69	0.55	0.43	0.36	0.31	0.27

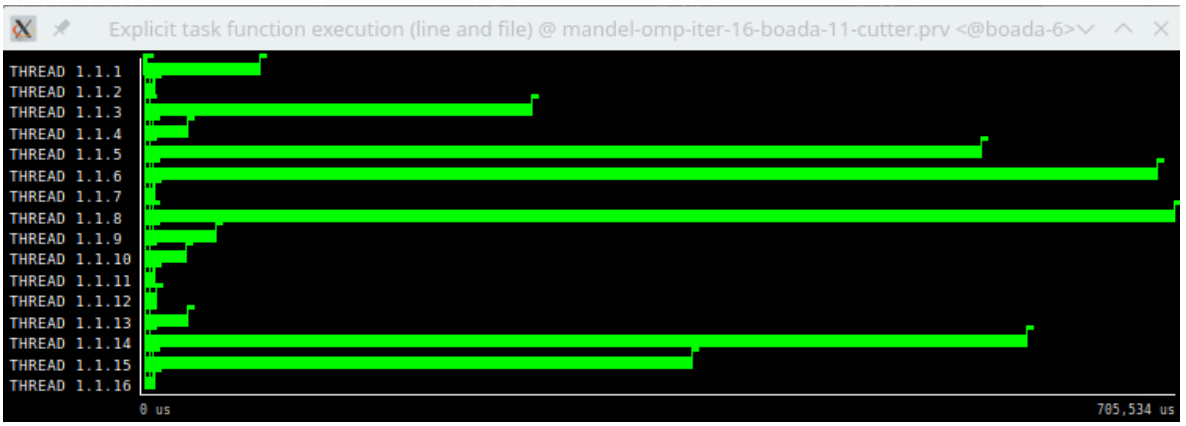
Table 1: Analysis done on Wed Nov 27 04:53:14 PM CET 2024, par4115

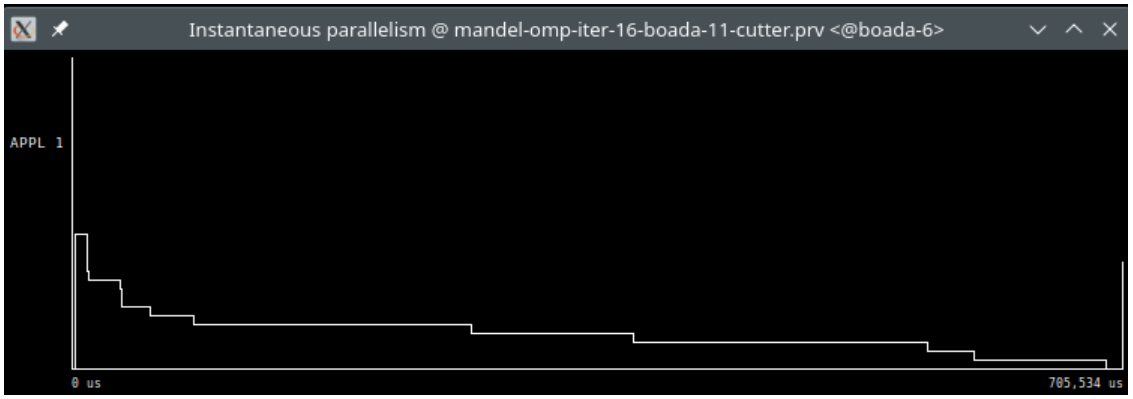
Overview of the Efficiency metrics in parallel fraction, ϕ=99.99%									
Number of Processors	1	2	4	6	8	10	12	14	16
Global efficiency	100.00%	88.73%	84.65%	68.75%	55.15%	42.98%	36.28%	31.01%	27.14%
Parallelization strategy efficiency	100.00%	88.96%	85.04%	73.75%	59.49%	47.75%	40.48%	34.95%	30.52%
Load balancing	100.00%	88.99%	85.10%	73.84%	59.59%	47.82%	40.56%	35.02%	30.62%
In execution efficiency	100.00%	99.97%	99.93%	99.88%	99.83%	99.86%	99.80%	99.80%	99.69%
Scalability for computation tasks	100.00%	99.75%	99.54%	93.21%	92.71%	90.00%	89.62%	88.75%	88.93%
IPC scalability	100.00%	99.99%	99.93%	99.95%	99.94%	99.94%	99.94%	99.92%	99.93%
Instruction scalability	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Frequency scalability	100.00%	99.76%	99.60%	93.26%	92.77%	90.06%	89.66%	88.82%	88.99%

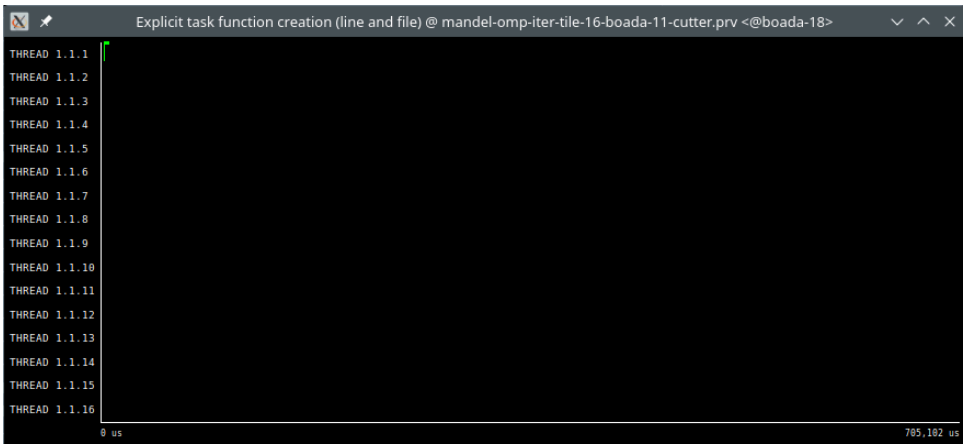
Table 2: Analysis done on Wed Nov 27 04:53:14 PM CET 2024, par4115

Statistics about explicit tasks in parallel fraction									
Number of Processors	1	2	4	6	8	10	12	14	16
Number of explicit tasks executed (total)	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0
LB (number of explicit tasks executed)	1.0	0.94	0.64	0.43	0.47	0.38	0.31	0.27	0.24
LB (time executing explicit tasks)	1.0	0.89	0.85	0.74	0.6	0.48	0.4	0.35	0.3
Time per explicit task (average us)	47802.86	47916.01	48008.19	51242.59	51473.73	53003.08	53193.66	53621.9	53450.86
Overhead per explicit task (synch %)	0.0	12.39	17.52	35.47	67.94	109.36	146.95	186.42	227.93
Overhead per explicit task (sched %)	0.0	0.01	0.01	0.0	0.01	0.01	0.01	0.01	0.01
Number of taskwait/taskgroup (total)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

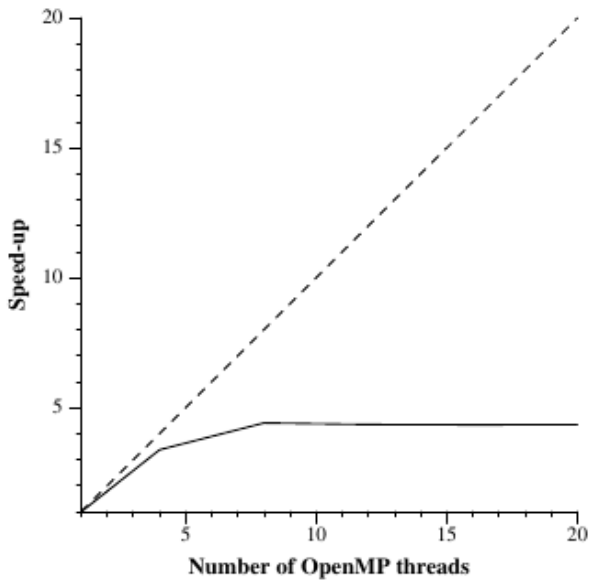
Paraver analysis







Strong scalability



par4115
Speed-up wrt sequential time (mandel funtion only)
Generated by par4115 on Wed Nov 27 05:33:11 PM CET 2024

Analysis

As it is shown in the Model Factor tables, the speed-up of the program for 2,4 and 6 threads is, more or less, what is to be expected. From 8 threads to 16, the speed-up remains the same (we can also see it in the strong scalability graphic) and we can see why in the Paraver task function execution graphic, where it shows a lot of load unbalance, hence why there is a poor instantaneous parallelism shown in the Paraver instantaneous parallelism graphic.

FINER GRAIN

Model Factor tables

Overview of whole program execution metrics									
Number of Processors	1	2	4	6	8	10	12	14	16
Elapsed time (sec)	3.07	1.55	0.98	0.69	0.43	0.36	0.36	0.32	0.27
Speedup	1.00	1.98	3.12	4.47	7.18	8.58	8.42	9.70	11.37
Efficiency	1.00	0.99	0.78	0.75	0.90	0.86	0.70	0.69	0.71

Table 1: Analysis done on Wed Nov 27 04:48:31 PM CET 2024, par4115

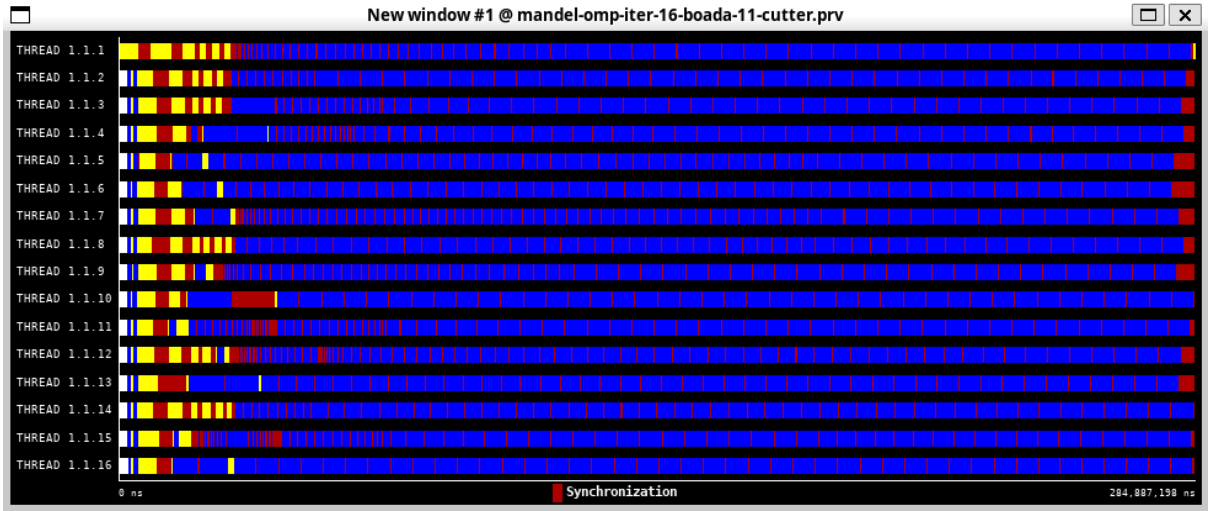
Overview of the Efficiency metrics in parallel fraction, ϕ=99.99%									
Number of Processors	1	2	4	6	8	10	12	14	16
Global efficiency	99.92%	98.94%	78.04%	74.54%	89.78%	85.83%	70.21%	69.29%	71.13%
Parallelization strategy efficiency	99.92%	99.08%	98.73%	97.61%	96.41%	95.46%	95.59%	94.61%	91.54%
Load balancing	100.00%	99.94%	99.82%	99.00%	99.37%	98.40%	98.00%	98.07%	95.83%
In execution efficiency	99.92%	99.14%	98.91%	98.59%	97.03%	97.01%	97.54%	96.47%	95.53%
Scalability for computation tasks	100.00%	99.86%	79.04%	76.36%	93.12%	89.91%	73.45%	73.24%	77.69%
IPC scalability	100.00%	99.83%	99.79%	99.78%	99.83%	99.76%	99.77%	99.77%	99.72%
Instruction scalability	100.00%	100.09%	81.83%	81.83%	100.01%	100.07%	81.81%	81.79%	87.10%
Frequency scalability	100.00%	99.94%	96.79%	93.52%	93.27%	90.07%	89.99%	89.75%	89.45%

Table 2: Analysis done on Wed Nov 27 04:48:31 PM CET 2024, par4115

Statistics about explicit tasks in parallel fraction									
Number of Processors	1	2	4	6	8	10	12	14	16
Number of explicit tasks executed (total)	8448.0	8448.0	8448.0	8448.0	8448.0	8448.0	8448.0	8448.0	8448.0
LB (number of explicit tasks executed)	1.0	0.95	0.63	0.43	0.45	0.37	0.31	0.26	0.29
LB (time executing explicit tasks)	1.0	1.0	1.0	0.99	0.99	0.99	0.98	0.99	0.97
Time per explicit task (average us)	362.98	365.16	462.14	479.91	395.1	410.74	502.38	505.08	481.64
Overhead per explicit task (synch %)	0.0	0.56	0.71	1.46	2.12	2.57	2.47	3.14	5.02
Overhead per explicit task (sched %)	0.07	0.33	0.51	0.87	1.34	1.78	1.74	2.08	3.3
Number of taskwait/taskgroup (total)	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0	64.0

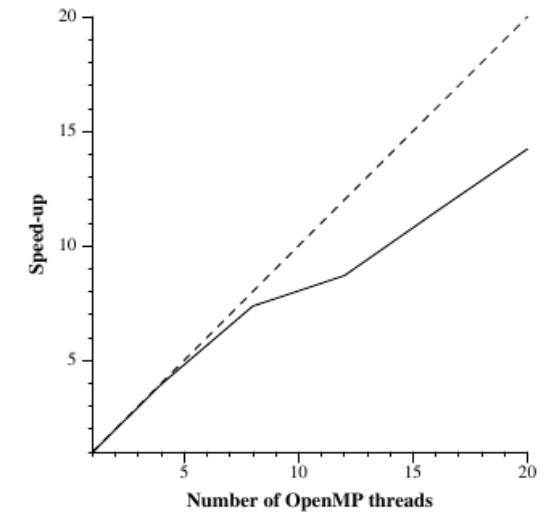
Table 3: Analysis done on Wed Nov 27 04:48:31 PM CET 2024, par4115

Paraver analysis



Blue run - Red sync - Yellow scheduling fork/join - White not created

Strong scalability



Analysis

As it is shown in the Model Factor tables and also in the strong scalability graphic; the more threads, bigger is the speed-up, even though we can see a significant decrease in speed-up between 8 and 12 threads. The efficiency for 16 threads is quite good: 70% and the load balancing is also good: 95% for 16 threads. The instantaneous parallelism remains the same during most of the execution, as it is shown in the Paraver instantaneous parallelism graphic, and this is because of the good load balancing: almost all the tasks are from the same size and can be executed in parallel (shown in the last Paraver graphic), so all threads can be executing a task at the same time. There is, but, a slight overhead when we increase the threads due to synchronization.

LEAF

Model Factor tables

Overview of whole program execution metrics									
Number of Processors	1	2	4	6	8	10	12	14	16
Elapsed time (sec)	1.62	1.24	1.24	1.33	1.33	1.37	1.37	1.37	1.37
Speedup	1.00	1.30	1.30	1.22	1.21	1.18	1.18	1.18	1.18
Efficiency	1.00	0.65	0.32	0.20	0.15	0.12	0.10	0.08	0.07

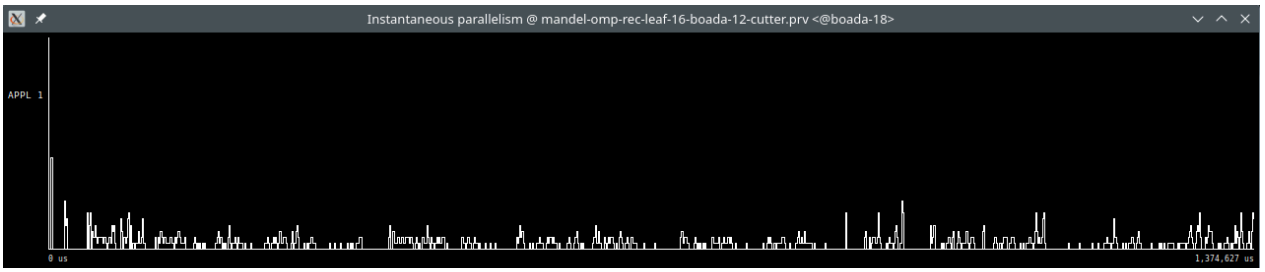
Table 1: Analysis done on Wed Nov 27 03:14:51 PM CET 2024, par4115

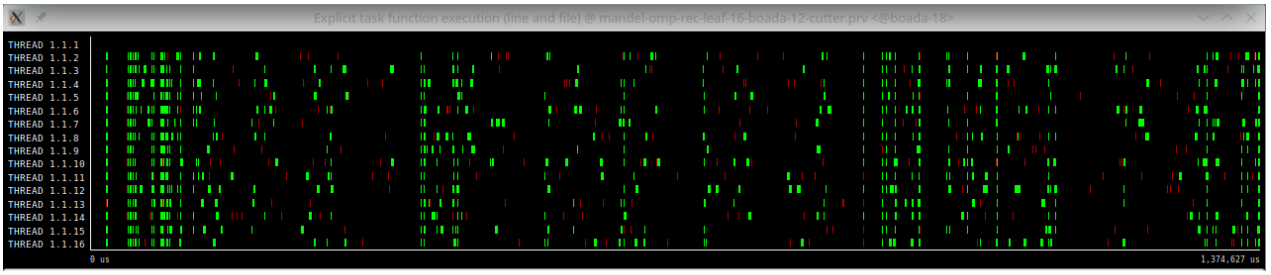
Overview of the Efficiency metrics in parallel fraction, ϕ=99.98%									
Number of Processors	1	2	4	6	8	10	12	14	16
Global efficiency	99.94%	65.05%	31.55%	20.30%	14.74%	11.77%	9.82%	8.41%	7.35%
Parallelization strategy efficiency	99.94%	65.23%	32.52%	21.78%	16.28%	13.09%	10.94%	9.40%	8.25%
Load balancing	100.00%	65.46%	32.64%	21.87%	16.34%	13.14%	10.98%	9.43%	8.28%
In execution efficiency	99.94%	99.66%	99.63%	99.60%	99.64%	99.63%	99.60%	99.59%	99.56%
Scalability for computation tasks	100.00%	99.72%	97.04%	93.17%	90.54%	89.93%	89.75%	89.51%	89.19%
IPC scalability	100.00%	99.66%	99.59%	99.54%	99.56%	99.49%	99.56%	99.53%	99.50%
Instruction scalability	100.00%	100.06%	100.06%	100.07%	100.06%	100.06%	100.06%	100.06%	100.06%
Frequency scalability	100.00%	99.99%	97.38%	93.54%	90.88%	90.34%	90.10%	89.88%	89.58%

Table 2: Analysis done on Wed Dec 4 03:16:41 PM CET 2024, par4115

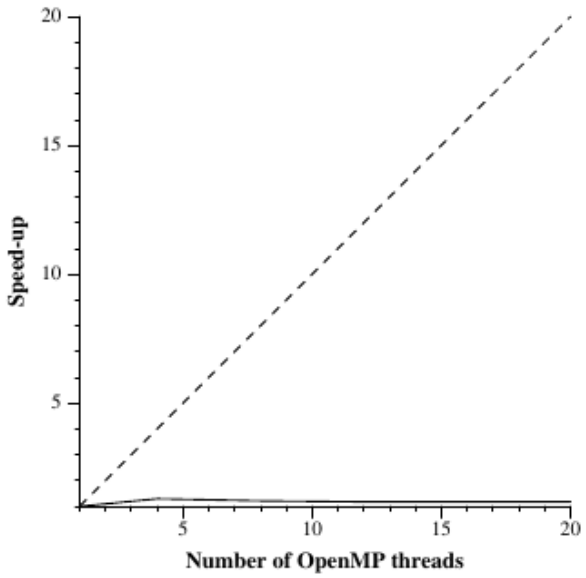
Statistics about explicit tasks in parallel fraction									
Number of Processors	1	2	4	6	8	10	12	14	16
Number of explicit tasks executed (total)	3013.0	3013.0	3013.0	3013.0	3013.0	3013.0	3013.0	3013.0	3013.0
LB (number of explicit tasks executed)	1.0	0.56	0.67	0.85	0.86	0.87	0.84	0.78	0.83
LB (time executing explicit tasks)	1.0	0.5	0.68	0.86	0.79	0.61	0.77	0.63	0.5
Time per explicit task (average us)	125.96	127.27	128.9	135.98	137.54	140.55	140.59	140.62	140.69
Overhead per explicit task (synch %)	0.0	224.15	888.0	1517.13	2211.89	2812.88	3454.9	4101.64	4747.13
Overhead per explicit task (sched %)	0.24	0.8	1.04	0.97	0.94	0.94	0.94	0.95	0.96
Number of taskwait/taskgroup (total)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Paraver analysis:





Strong scalability



par4115
Speed-up wrt sequential time (mandel funtion only)
Generated by par4115 on Wed Nov 27 03:14:59 PM CET 2024

Analysis

As we can see in the Paraver images, there are a lot of tasks being created but the instantaneous parallelism isn't as high as we might expect. That is because each task does not do a lot of work, being the cost of creating each task higher than what each task has to do. It is also seen in the model factor tables: there is a very poor speed-up for 16 threads: 1.18, the efficiency for 16 threads is also very poor: 7%. The strong scalability graphic backs this theory, showing that parallelizing this program and adding more threads to the execution does not improve the execution time as good as it could. So, we can conclude that the leaf strategy is not a good strategy for parallelizing this code.

TREE

MAX_DEPTH = 4

Model Factor tables

Overview of whole program execution metrics									
Number of Processors	1	2	4	6	8	10	12	14	16
Elapsed time (sec)	1.61	0.82	0.43	0.31	0.26	0.22	0.20	0.17	0.17
Speedup	1.00	1.96	3.71	5.16	6.20	7.16	8.24	9.35	9.73
Efficiency	1.00	0.98	0.93	0.86	0.78	0.72	0.69	0.67	0.61

Table 1: Analysis done on Mon Dec 9 06:10:28 PM CET 2024, par4115

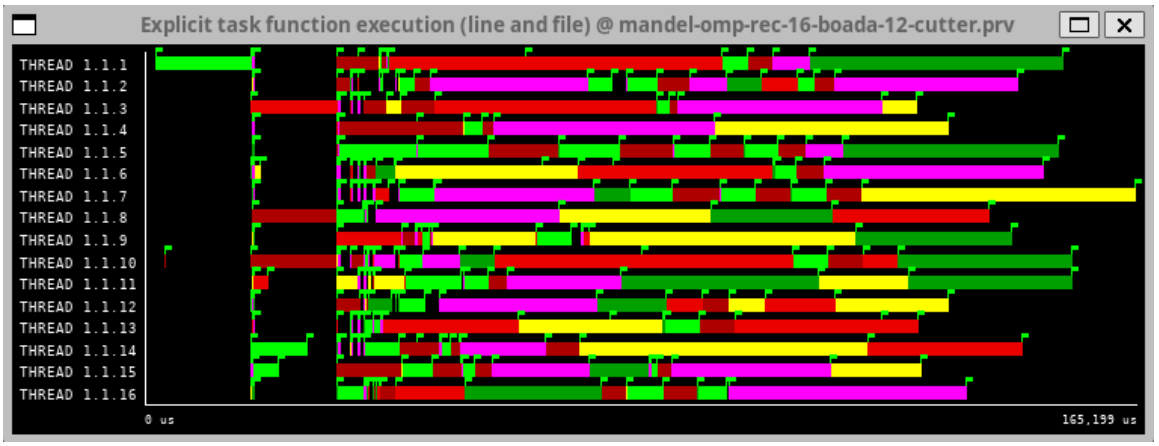
Overview of the Efficiency metrics in parallel fraction, ϕ=99.98%									
Number of Processors	1	2	4	6	8	10	12	14	16
Global efficiency	99.98%	98.12%	92.84%	86.03%	77.59%	71.67%	68.77%	66.90%	60.93%
Parallelization strategy efficiency	99.98%	98.40%	93.24%	92.17%	84.57%	79.53%	76.51%	74.57%	68.31%
Load balancing	100.00%	99.71%	98.64%	95.56%	94.88%	85.91%	87.83%	87.92%	82.23%
In execution efficiency	99.98%	98.69%	94.52%	96.45%	89.13%	92.58%	87.11%	84.81%	83.07%
Scalability for computation tasks	100.00%	99.71%	99.57%	93.34%	91.75%	90.11%	89.88%	89.72%	89.19%
IPC scalability	100.00%	99.95%	99.93%	99.91%	99.90%	99.89%	99.89%	99.85%	99.82%
Instruction scalability	100.00%	100.02%	100.02%	100.02%	100.02%	100.02%	100.01%	100.01%	100.01%
Frequency scalability	100.00%	99.74%	99.63%	93.40%	91.82%	90.20%	89.97%	89.84%	89.33%

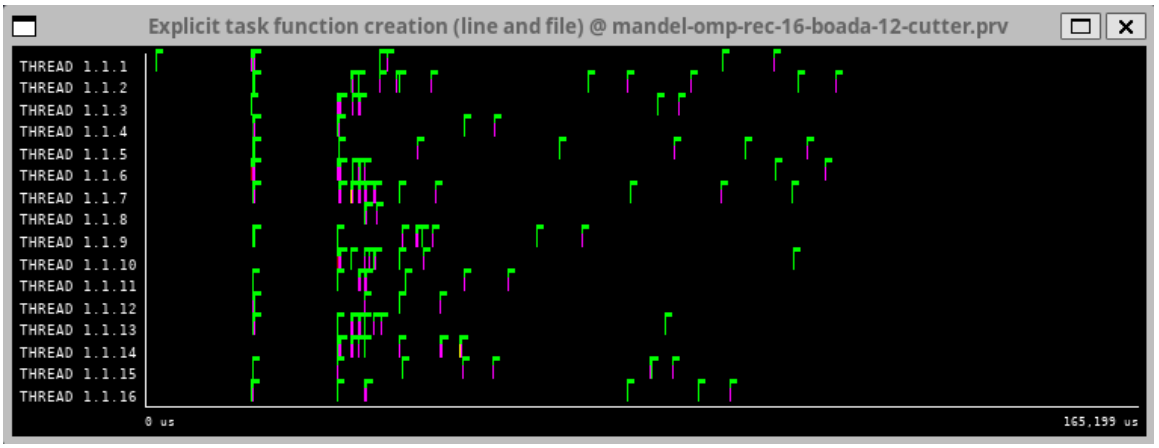
Table 2: Analysis done on Mon Dec 9 06:10:28 PM CET 2024, par4115

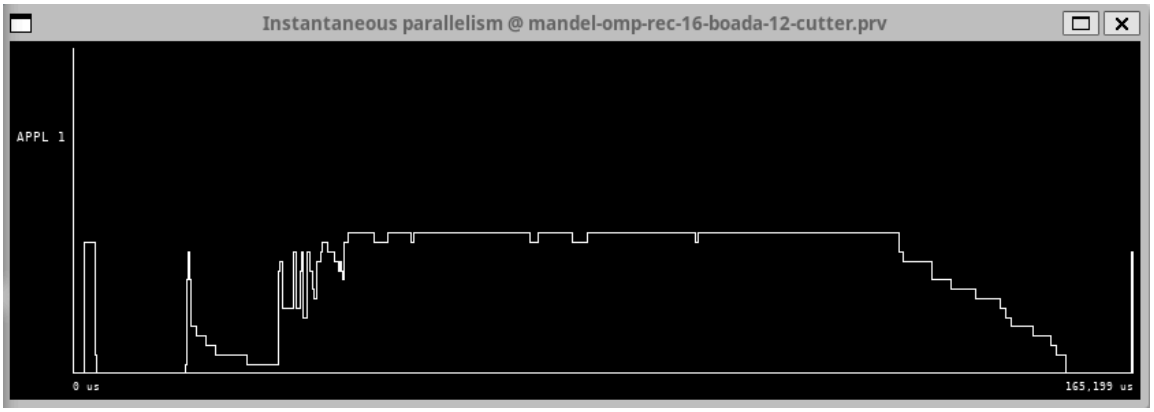
Statistics about explicit tasks in parallel fraction									
Number of Processors	1	2	4	6	8	10	12	14	16
Number of explicit tasks executed (total)	818.0	818.0	818.0	818.0	818.0	818.0	818.0	818.0	818.0
LB (number of explicit tasks executed)	1.0	0.72	0.57	0.56	0.65	0.49	0.59	0.55	0.49
LB (time executing explicit tasks)	1.0	1.0	0.99	0.96	0.96	0.88	0.89	0.88	0.83
Time per explicit task (average us)	1963.36	1969.01	1970.56	2117.72	2152.78	2218.92	2207.98	2184.53	2212.86
Overhead per explicit task (synch %)	0.01	1.58	7.07	8.1	17.69	24.66	29.39	33.14	44.58
Overhead per explicit task (sched %)	0.01	0.02	0.03	0.03	0.04	0.05	0.07	0.14	0.14
Number of taskwait/taskgroup (total)	169.0	169.0	169.0	169.0	169.0	169.0	169.0	169.0	169.0

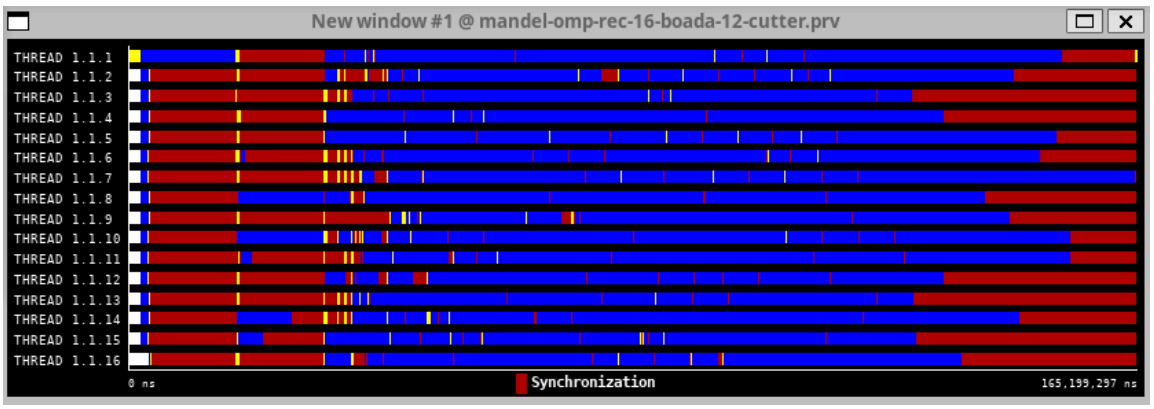
Table 3: Analysis done on Mon Dec 9 06:10:28 PM CET 2024, par4115

Paraver analysis

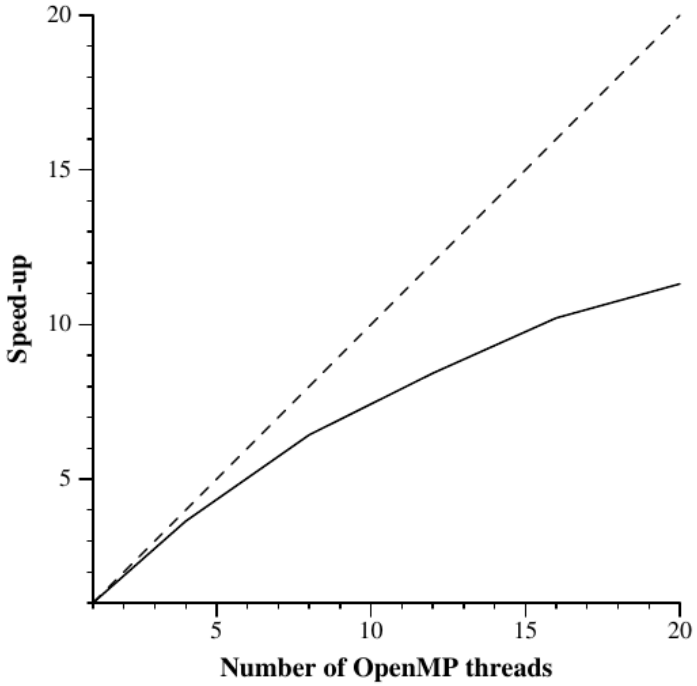








Strong scalability



par4115
Speed-up wrt sequential time (mandel funtion only)
Generated by par4115 on Mon Dec 9 06:39:31 PM CET 2024

MAX_DEPTH = 5

Modelfactor tables

Overview of whole program execution metrics									
Number of Processors	1	2	4	6	8	10	12	14	16
Elapsed time (sec)	1.61	0.82	0.44	0.32	0.25	0.21	0.19	0.17	0.15
Speedup	1.00	1.97	3.66	5.11	6.41	7.61	8.65	9.58	10.80
Efficiency	1.00	0.99	0.91	0.85	0.80	0.76	0.72	0.68	0.67

Table 1: Analysis done on Mon Dec 9 06:13:57 PM CET 2024, par4115

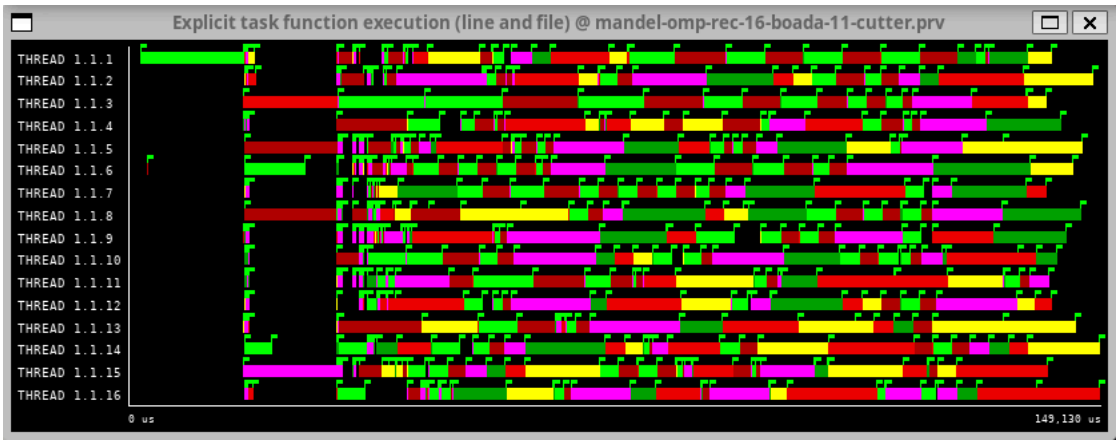
Overview of the Efficiency metrics in parallel fraction, ϕ=99.97%									
Number of Processors	1	2	4	6	8	10	12	14	16
Global efficiency	99.93%	98.50%	91.44%	85.13%	80.15%	76.12%	72.14%	68.52%	67.57%
Parallelization strategy efficiency	99.93%	98.57%	94.53%	91.10%	87.46%	84.53%	80.22%	76.47%	75.56%
Load balancing	100.00%	98.85%	98.27%	94.39%	93.75%	92.19%	91.96%	83.93%	89.09%
In execution efficiency	99.93%	99.72%	96.19%	96.52%	93.29%	91.69%	87.24%	91.11%	84.82%
Scalability for computation tasks	100.00%	99.93%	96.73%	93.45%	91.64%	90.05%	89.92%	89.60%	89.42%
IPC scalability	100.00%	99.94%	99.90%	99.89%	99.88%	99.80%	99.81%	99.74%	99.74%
Instruction scalability	100.00%	100.05%	100.05%	100.05%	100.05%	100.05%	100.05%	100.04%	100.04%
Frequency scalability	100.00%	99.94%	96.78%	93.51%	91.71%	90.18%	90.05%	89.79%	89.61%

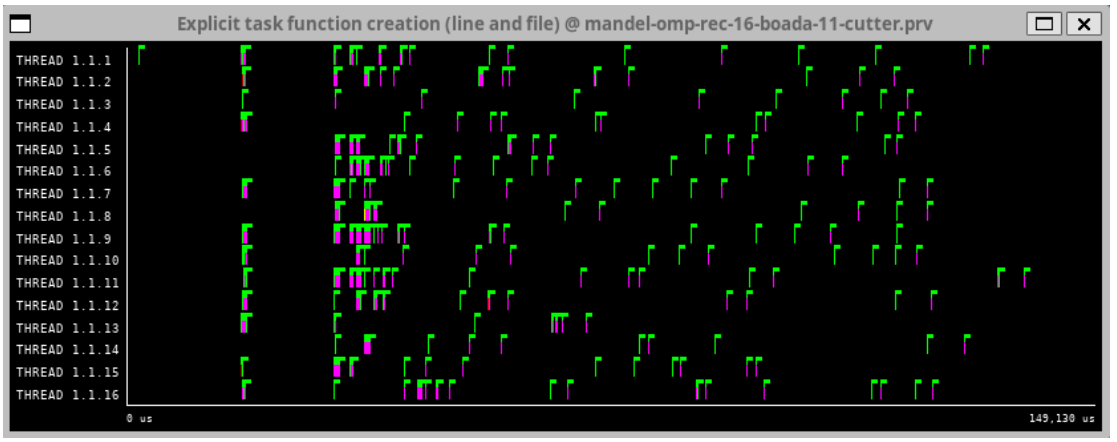
Table 2: Analysis done on Mon Dec 9 06:13:57 PM CET 2024, par4115

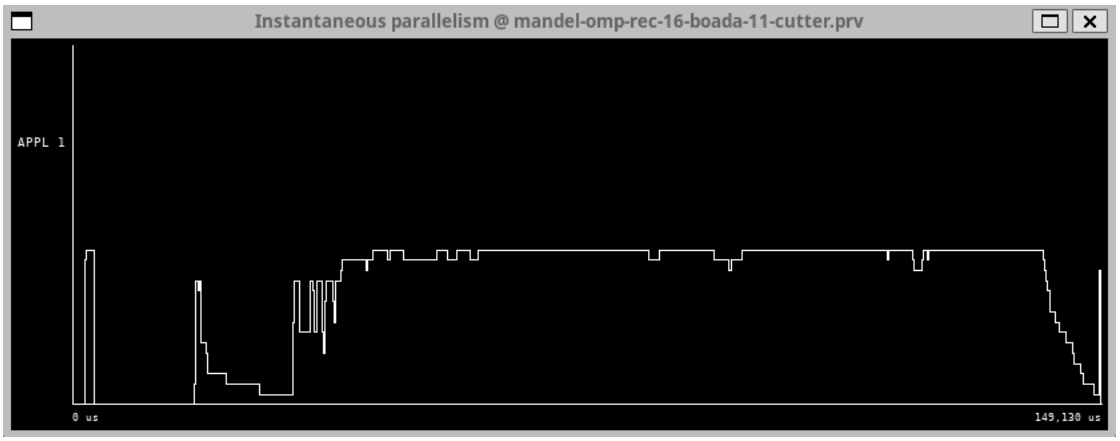
Statistics about explicit tasks in parallel fraction									
Number of Processors	1	2	4	6	8	10	12	14	16
Number of explicit tasks executed (total)	2358.0	2358.0	2358.0	2358.0	2358.0	2358.0	2358.0	2358.0	2358.0
LB (number of explicit tasks executed)	1.0	0.74	0.67	0.57	0.46	0.42	0.38	0.44	0.59
LB (time executing explicit tasks)	1.0	0.99	0.98	0.95	0.94	0.92	0.89	0.83	0.91
Time per explicit task (average us)	681.56	682.22	704.56	734.57	747.8	756.41	765.56	774.42	769.56
Overhead per explicit task (synch %)	0.03	1.37	5.62	9.34	13.73	17.56	23.39	28.95	30.26
Overhead per explicit task (sched %)	0.04	0.05	0.06	0.07	0.1	0.17	0.19	0.26	0.35
Number of taskwait/taskgroup (total)	481.0	481.0	481.0	481.0	481.0	481.0	481.0	481.0	481.0

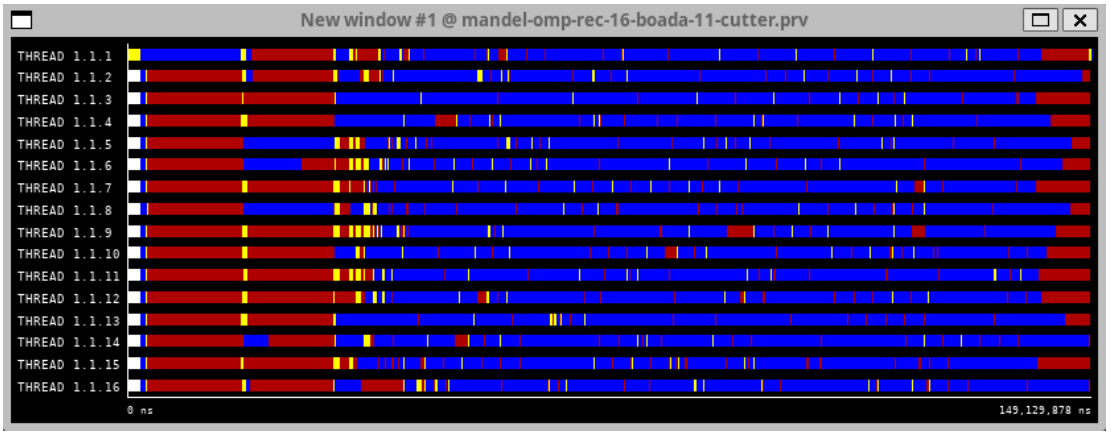
Table 3: Analysis done on Mon Dec 9 06:13:57 PM CET 2024, par4115

Paraver analysis

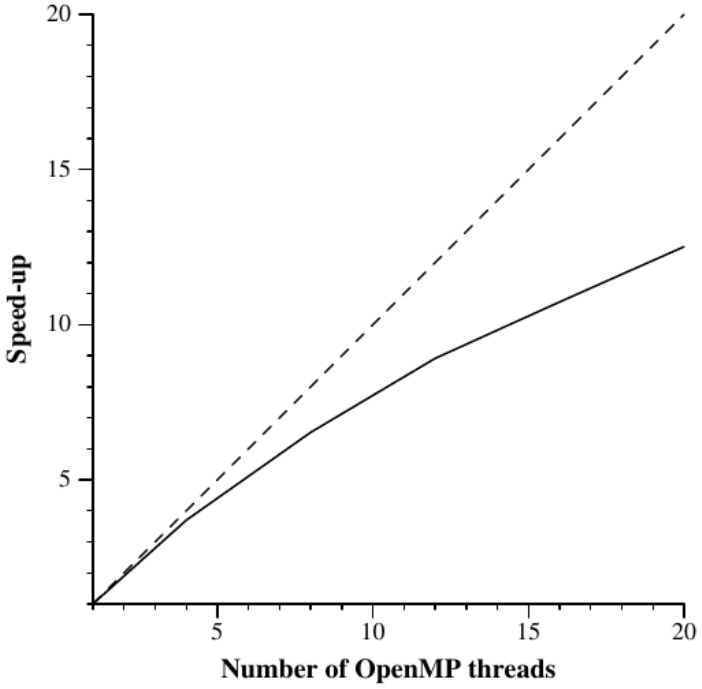








Strong scalability



par4115
Speed-up wrt sequential time (mandel funtion only)
Generated by par4115 on Mon Dec 9 06:45:12 PM CET 2024

MAX_DEPTH = 6

Modelfactor tables

Overview of whole program execution metrics									
Number of Processors	1	2	4	6	8	10	12	14	16
Elapsed time (sec)	1.62	0.82	0.43	0.32	0.25	0.21	0.18	0.16	0.15
Speedup	1.00	1.97	3.78	5.13	6.50	7.66	8.78	9.81	10.90
Efficiency	1.00	0.99	0.94	0.86	0.81	0.77	0.73	0.70	0.68

Table 1: Analysis done on Mon Dec 9 06:16:28 PM CET 2024, par4115

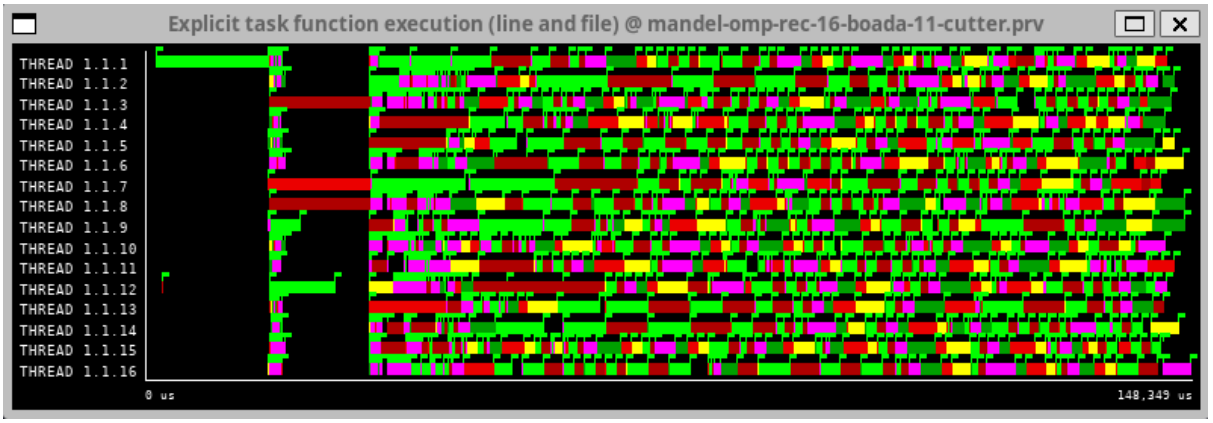
Overview of the Efficiency metrics in parallel fraction, ϕ=99.98%									
Number of Processors	1	2	4	6	8	10	12	14	16
Global efficiency	99.82%	98.38%	94.36%	85.43%	81.13%	76.63%	73.14%	70.05%	68.09%
Parallelization strategy efficiency	99.82%	98.51%	94.80%	91.40%	88.47%	85.26%	81.67%	78.25%	76.52%
Load balancing	100.00%	99.26%	99.08%	95.82%	92.51%	93.60%	85.66%	83.54%	89.07%
In execution efficiency	99.82%	99.24%	95.68%	95.39%	95.63%	91.09%	95.35%	93.67%	85.91%
Scalability for computation tasks	100.00%	99.87%	99.54%	93.46%	91.70%	89.88%	89.56%	89.52%	88.98%
IPC scalability	100.00%	99.89%	99.80%	99.86%	99.71%	99.57%	99.49%	99.64%	99.42%
Instruction scalability	100.00%	100.14%	100.14%	100.14%	100.14%	100.14%	100.13%	100.13%	100.13%
Frequency scalability	100.00%	99.84%	99.60%	93.47%	91.84%	90.14%	89.89%	89.72%	89.38%

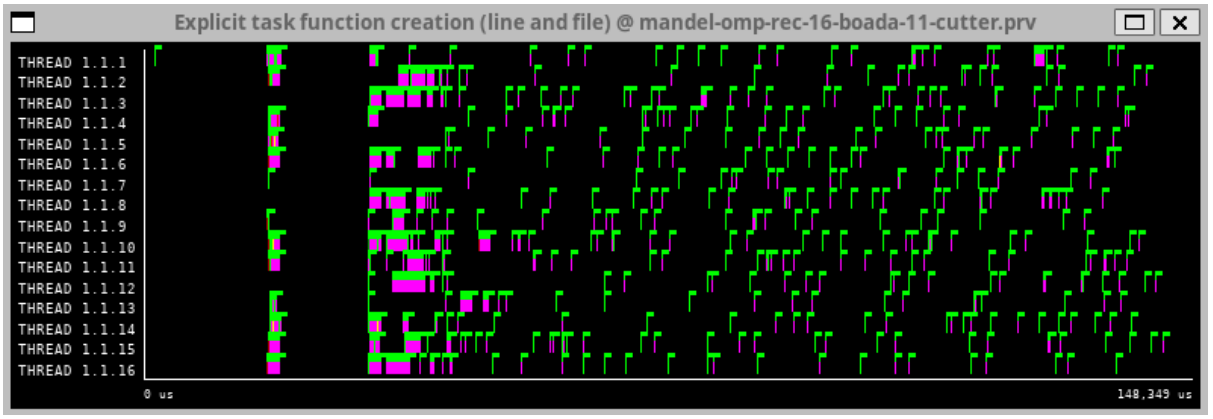
Table 2: Analysis done on Mon Dec 9 06:16:28 PM CET 2024, par4115

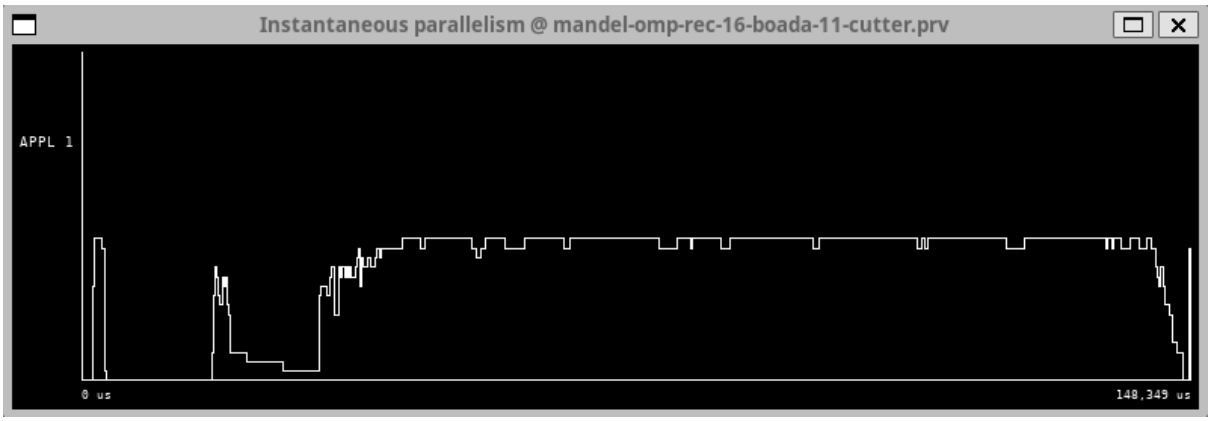
Statistics about explicit tasks in parallel fraction									
Number of Processors	1	2	4	6	8	10	12	14	16
Number of explicit tasks executed (total)	6810.0	6810.0	6810.0	6810.0	6810.0	6810.0	6810.0	6810.0	6810.0
LB (number of explicit tasks executed)	1.0	0.77	0.71	0.46	0.57	0.57	0.58	0.57	0.41
LB (time executing explicit tasks)	1.0	0.99	0.99	0.96	0.94	0.94	0.86	0.84	0.9
Time per explicit task (average us)	236.32	236.98	237.74	255.06	261.72	264.98	266.44	267.4	269.32
Overhead per explicit task (synch %)	0.07	1.33	5.16	8.98	12.23	16.07	20.43	25.43	27.88
Overhead per explicit task (sched %)	0.11	0.14	0.19	0.16	0.26	0.5	0.81	1.0	1.26
Number of taskwait/taskgroup (total)	1397.0	1397.0	1397.0	1397.0	1397.0	1397.0	1397.0	1397.0	1397.0

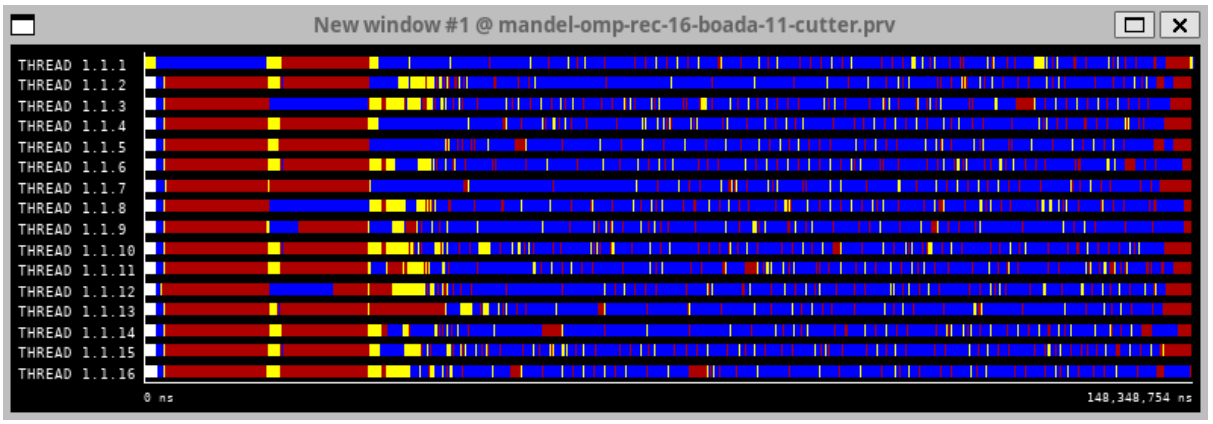
Table 3: Analysis done on Mon Dec 9 06:16:28 PM CET 2024, par4115

Paraver analysis

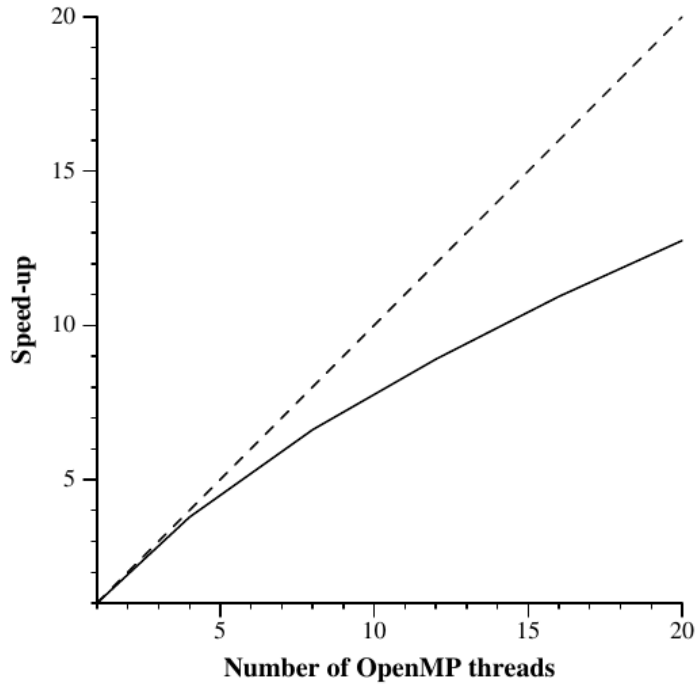








Strong scalability



par4115
Speed-up wrt sequential time (mandel funtion only)
Generated by par4115 on Sun Dec 8 07:09:08 PM CET 2024

Analysis

For this program, we analyzed which could be the best cut-off. The program does 7 levels of recursion, so it had to be either 7 or below 7. When we executed the program with a cut-off of 7 (it would be the same with no cut-off at all), the elapsed time was 0.15 seconds. With 4,5 and 6 levels of recursion the elapsed time was more or less the same as with 7 levels: 0.15 seconds, and below 4 levels the elapsed time increased. So we analyzed deeper the latest to decide which was best. For 4 levels of recursion, the program created 818 explicit tasks; for 5 levels of recursion the program created 2358 tasks and for 6 levels of recursion the program created 6810 tasks. The strong scalability graphic improves enough to choose 5 levels of recursion over 4 levels of recursion (the speed-up for 4 threads is 10.7 with strong scaling and for 5 threads is 12.55) but it doesn't improve enough to choose 6 levels of recursion over 5 (keeping in mind the number of tasks created). Furthermore, if we analyze the sync graphics we can see that, at first, most of the tasks spend a lot of time syncing and that is because when doing the first part of the code (the loops), there is a taskgroup that waits for all the tasks created.

TILE AND FINE GRAIN COMPARISON

The Model Factor tables from the fine grain program show a much better performance compared to the tile program. The efficiency is better, the speed-up is increasing continuously (the tile one for 8 threads and above the speed-up remains the same) the reason for this is due to the amount of generated tasks. The load balance is much better being the tasks created of approximately the same size, contrary to the tile program that the balance of the tasks varies a lot. All of this is reflected in the Paraver and strong scalability graphics. Overall, the fine grain strategy is a better strategy in all aspects to parallelize this program.

LEAF AND TREE COMPARISON

(all the analysis will be with a cut-off of 5 of the tree recursion strategy)

Looking at the Model Factor tables from the leaf and tree program, we can soon realize that the tree program has better performance overall. The speed-up from the leaf strategy is really low, being the speed-up from the tree strategy almost 10 times better (for 16 threads-> 10.8/1.18) and the efficiency is also almost 10 times better (0.67/0.07). The Paraver graphics show a better and more constant parallelization. The strong scaling graphic reflects it all, showing a really poor scaling for the leaf strategy. In conclusion, for this program, a tree recursion strategy is better than a leaf recursion strategy because the tasks in the leaf are so small that the overhead from synchronization is higher than the work of the task itself, making the strategy a bad choice.