Deliverable

Lab 1: Experimental setup and tools

Pablo Vizcaino
Guillem Ramírez Miranda

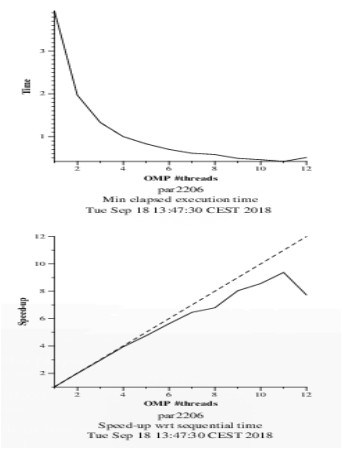
September 2018

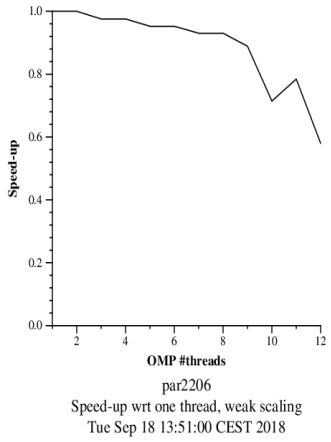
Node architecture and memory

The boada server is organised by nodes. Each node contains a number of sockets which contains cores and each core contains threads.

	boada-1 to boada-4	boada-5	boada-6 to boada-8
Number of sockets per node	2 sockets	2 sockets	2 sockets
Number of cores per socket	6 cores	6 cores	8 cores
Number of threads per core	2 threads	2 threads	1 thread
Maximum core frequency	2,95 Ghz	2,6 Ghz	1,7 Ghz
L1-I cache size (per-core)	32 KB	32 KB	32 KB
L1-D cache size (per-core)	32 KB	32 KB	32 KB
L2 cache size (per-core)	256 KB	256 KB	256 KB
Last-level cache size (per-socket)	12288 KB	15360 KB	20480 KB
Main memory size (per socket)	12 GB	31 GB	16 GB
Main memory size (per node)	24 GB	62 GB	32 GB

Strong vs. weak scalability





When we refer to strong scalability we address to the increase of speedup when we increase the number of resources with a fixed problem size
On the other hand, with weak scalability we measure the performance when the problem size is increased proportional to the number of resources.
In the following figure we can observe that in strong scalability time decreases with more threads and in weak it decreases due to overheads (the ideal situation would be an straight line).

Analysis of task decompositions for 3DFFT

This table shows the increment of parallelism between each version due to the bigger granularity.

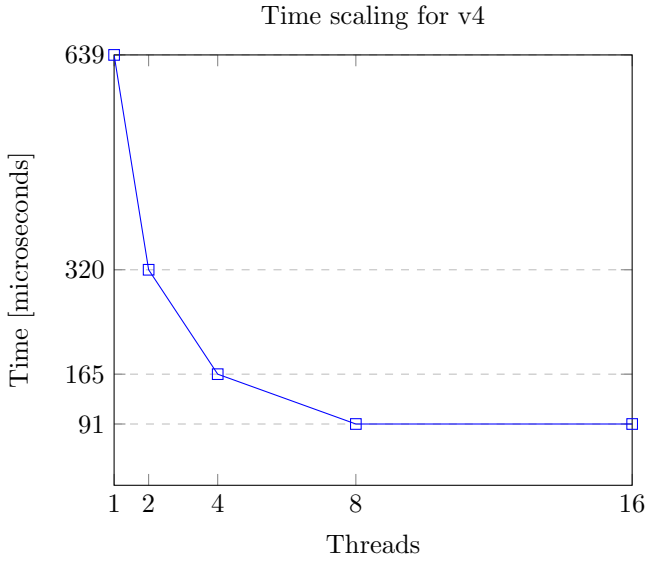
Version	T_1	T_∞	Parallelism
Seq	0,64 s	0.64 s	1
v1	0.64 s	0.64 s	1
v2	0.64 s	0.36 s	1.78
v3	0.64 s	0.15 s	4.27
v4	0.64 s	0.064 s	10.67
v5	0.64 s	0.008 s	80

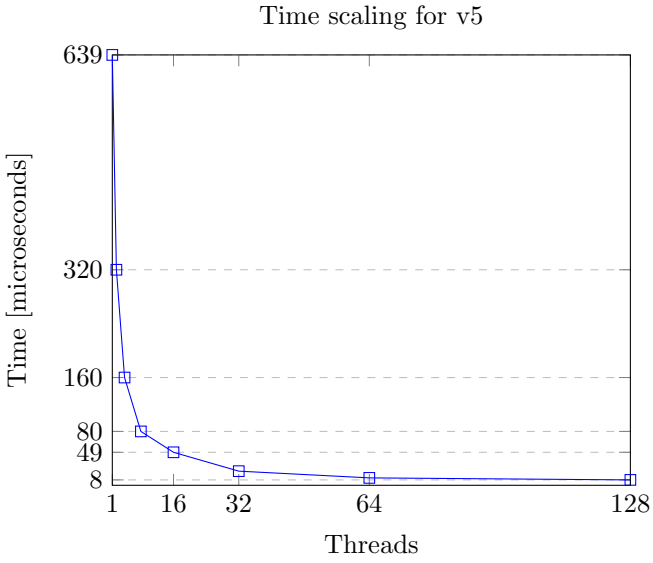
The granularity of V4 is on the 'k' loop, which is the outer one. In the V5 there is more granularity since it is at 'j' level as we can see in this sample of code:

```
    for (k=0; k<N; k++) {
     for (j=0; j<N; j++) {
    tareador_start_task("transpose_zx_planes_loop_j");
       for (i=0; i<N; i++)
       {
         in_fftw[i][j][k][0] = tmp_fftw[k][j][i][0];
         in_fftw[i][j][k][1] = tmp_fftw[k][j][i][1];
       }
     tareador_end_task("transpose_zx_planes_loop_j");

     }
}
```

In this two graphics we can see the impact of the granularity in the scalability. V4's granularity limits the time improvement at 8 threads. Instead, V5 continues improving with more threads.





Understanding the parallel execution of 3DFFT

Version	Φ	S_∞	T_1	T_8	S_8
Initial	0.54	2.15	2.89 s	1.41 s	2.05
Improved Φ	0.89	9.09	2.30 s	0.81 s	3.52
Improved parallel overheads	0.88	8.33	2.20 s	0.80 s	2.75
improved work-distribution overheads	0.87	7.74	2.19 s	0.44 s	4.97

Improved Φ

In the improved version we added the parallelization instrumentation to the init complex grid routine.

Threads	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	90.05	0	8.69	1.17	0.08	0
THREAD 1.1.2	67.87	23.52	8.54	0	0.06	0
THREAD 1.1.3	68.76	23.51	7.66	0	0.07	0
THREAD 1.1.4	65.39	23.54	11.01	0	0.06	0
THREAD 1.1.5	68.46	23.51	7.97	0	0.06	0
THREAD 1.1.6	70.91	23.55	5.48	0	0.06	0
THREAD 1.1.7	70.23	23.55	6.16	0	0.06	0
THREAD 1.1.8	67.46	23.53	8.95	0	0.06	0
Average	71.14125	20.58875	8.0575	0.14625	0.06375	0

Improved parallel overheads

For the sake of reducing overheads we moved the threads creation outside the loops so we don't create them each 'k' iteration.We can observe that the time of the threads not being created is increased and also the running times are lowered.

Threads	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	90.48	0	9.4	0.09	0.03	0
THREAD 1.1.2	66.25	26.19	7.53	0	0.03	0
THREAD 1.1.3	64.26	26.19	9.52	0	0.03	0
THREAD 1.1.4	63.64	26.19	10.13	0	0.03	0
THREAD 1.1.5	68.41	26.22	5.33	0	0.03	0
THREAD 1.1.6	63.9	26.29	9.78	0	0.03	0
THREAD 1.1.7	66.89	26.27	6.81	0	0.03	0
THREAD 1.1.8	69.45	26.38	4.14	0	0.03	0
Average	69.16	22.96625	7.83	0.01125	0.03	0

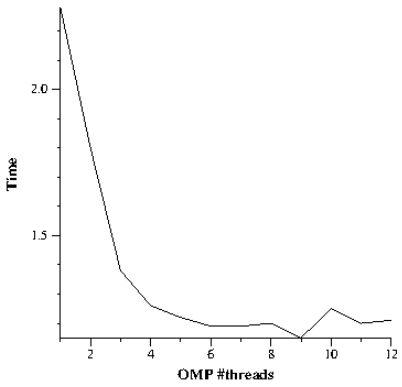
Improved work-distribution overheads

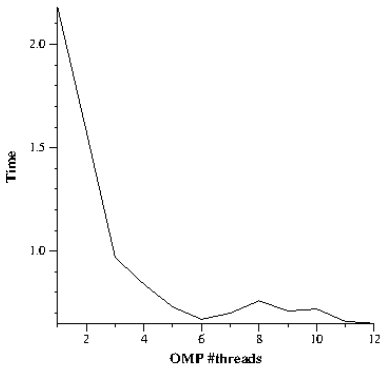
On this final version we moved the work-distribution pragma outside the loops, so the overhead of distributing the work each loop is removed. In the table we can observe that the time the threads were running is lowered and that the value of synchronisation is the same,and also the time not being created has increased.

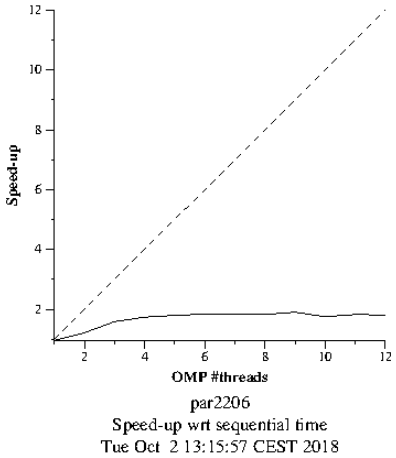
Threads	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	93.81	0	6.03	0.13	0.03	0
THREAD 1.1.2	67.06	24.01	8.89	0	0.03	0
THREAD 1.1.3	66.84	24.01	9.12	0	0.03	0
THREAD 1.1.4	69.23	24.01	6.72	0	0.03	0
THREAD 1.1.5	68.79	24.02	7.17	0	0.03	0
THREAD 1.1.6	70.43	24.02	5.52	0	0.03	0
THREAD 1.1.7	68.09	24.02	7.86	0	0.03	0
THREAD 1.1.8	67.63	24.01	8.33	0	0.03	0
Average	71.485	21.0125	7.455	0.01625	0.03	0

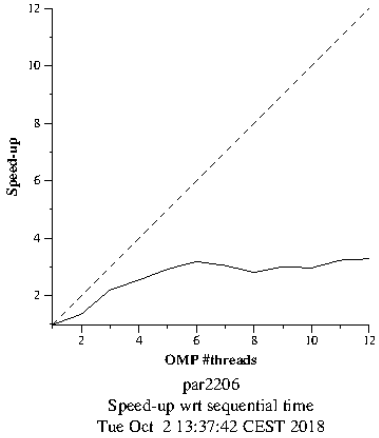
Strong scallability plots

You can take a look at strong scallability plots to confirm that with each optimisation we obtain better results.



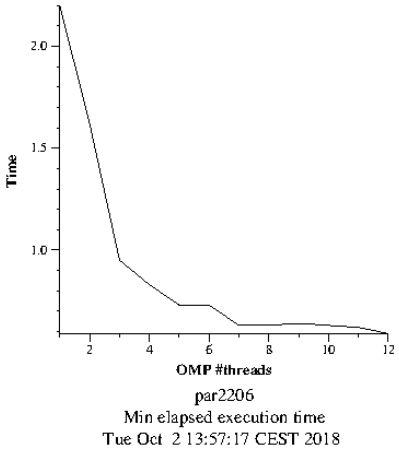


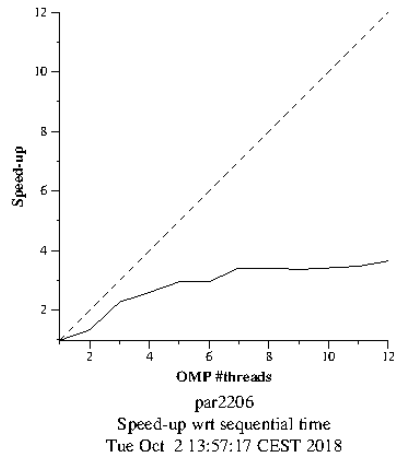




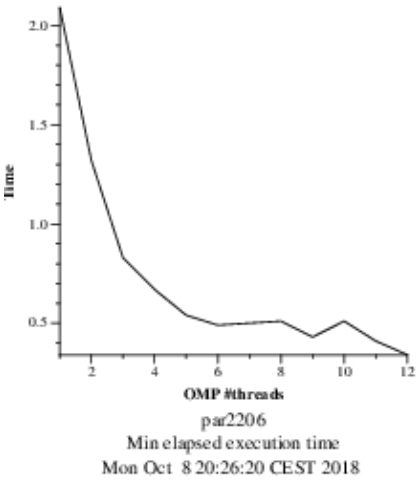
Initial version

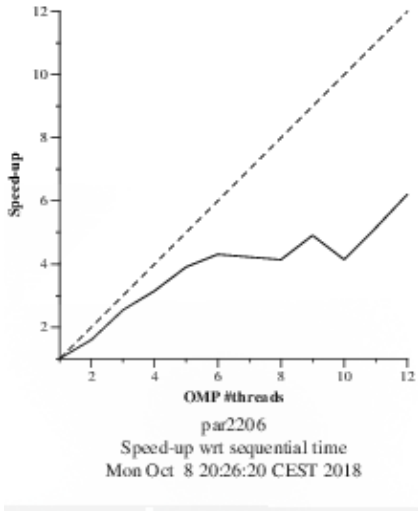
Improved Φ





Improved parallel overheads





Improved work-distribution overheads