PAR2202
Oriol Fonollà
Genís Bosch

PAR Laboratory

Lab 1: Experimental setup and tools

Node architecture and memory

1. Complete the following table with the relevant architectural characteristics of the different node types available in boada:

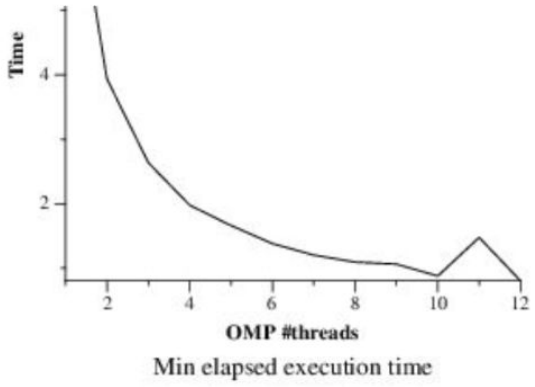
	Boada 1 to 4	Boada 5	Boada 6 to 8
Number of sockets per node	2	2	2
Number of cores per socket	6	6	8
Number of threads per core	2	2	1
Maximum core frequency	2395 MHZ	2600MHZ	1700 MHZ
L1-I cache size (per-core)	32KB	32KB	32KB
L1-D cache size (per-core)	32KB	32KB	32KB
L2 cache size (per-core)	256KB	256KB	256KB
Last-level cache size (per-socket)	12288 KB	15360 KB	20480KB
Main memory size (per socket)	23 GB	31GB	31GB
Main memory size (per node)	12 GB	63GB	16GB

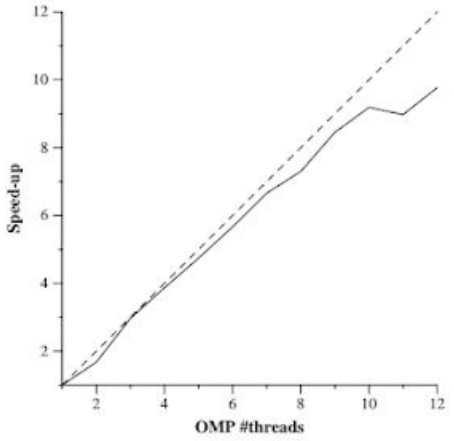
2. Include in the document the architectural diagram for one of the nodes boada-1 to boada-4 as obtained when using the lstopo command.

PAR2202
Oriol Fonollà
Genís Bosch

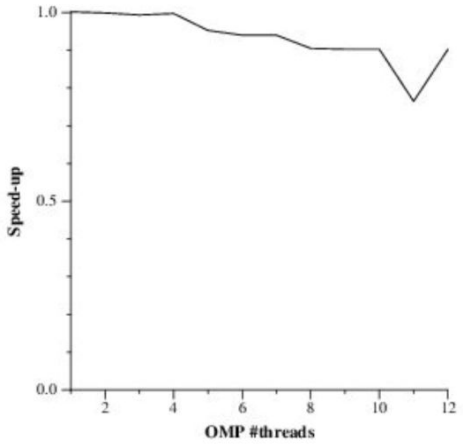
Timing sequential and parallel executions

3. Plot the execution time and speed–up that is obtained when varying the number of threads (strong scalability) and problem size (weak scalability) for pi omp.c on the different node types available in boada. Reason about the results that are obtained.





Strong scalability: El temps d'execució decau a mesura que es fan servir més processadors, ja que es millora el paralel·lisme del programa. El speed up es manté gariabé constant.



Weak scalability: Es poden veure les dependències entre la mida del programa i el número de processadors.

Analysis of task decompositions with Tareador

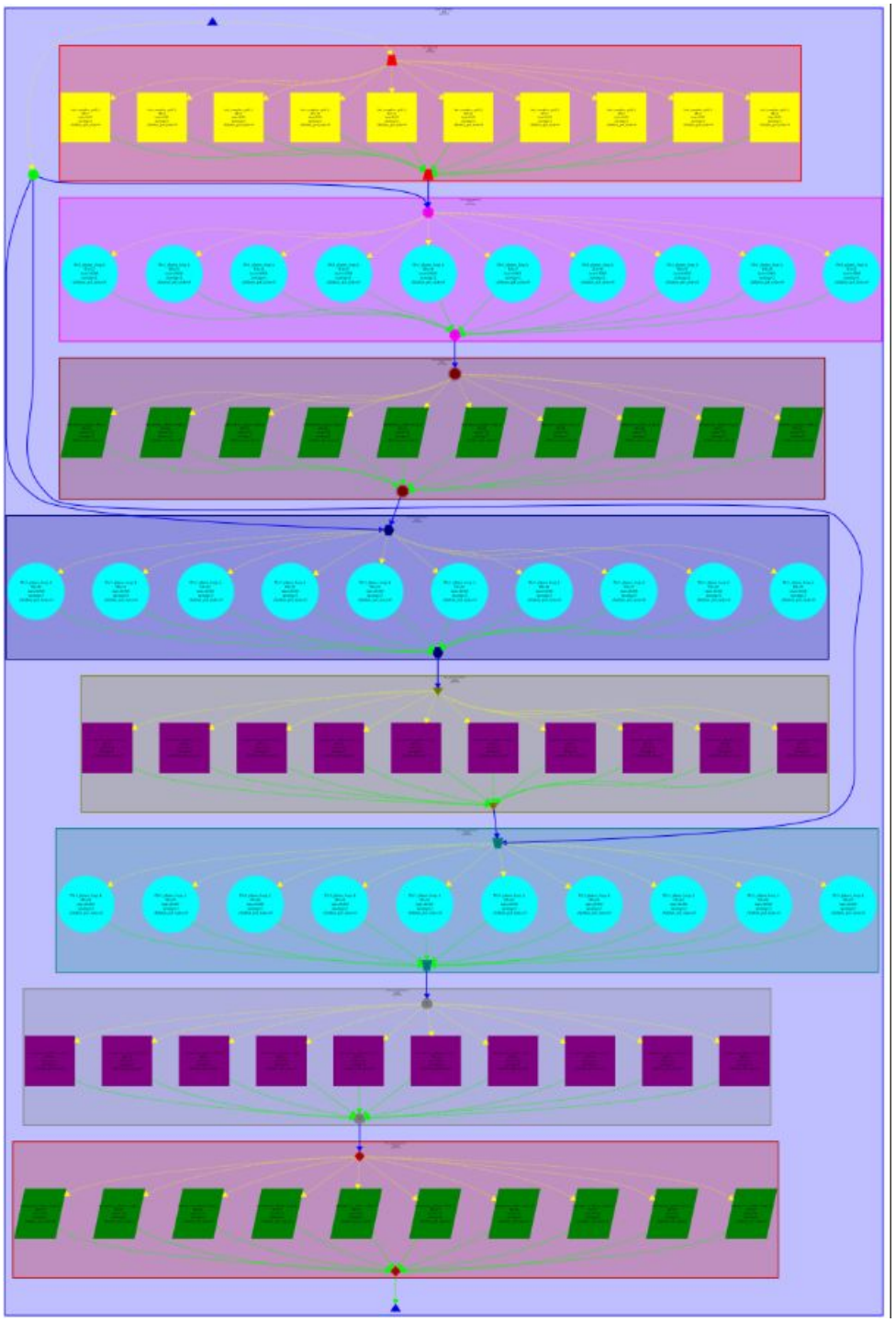
4. Include the relevant(s) part(s) of the code to show the new task definition(s) in v4 of 3dfft seq.c. Capture the final task dependence graph that has been obtained after version v4.

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_k");
    for (j = 0; j < N; j++)
      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_k");
  }
}
```

PAR2202
Oriol Fonollà
Genís Bosch



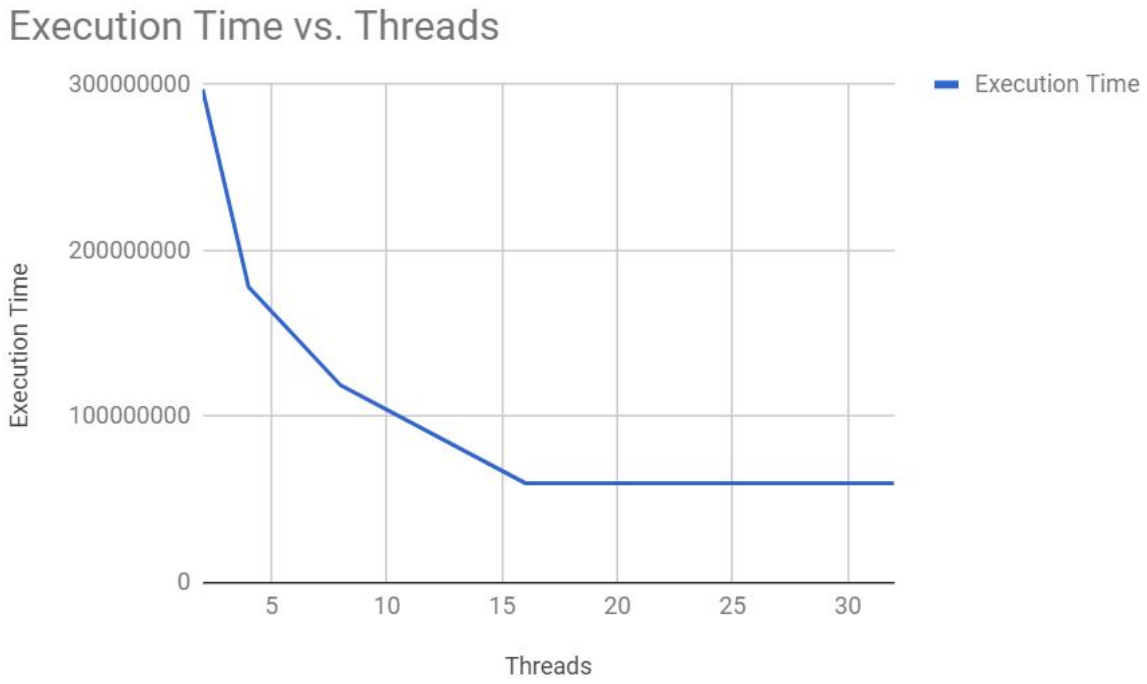
PAR2202
Oriol Fonollà
Genís Bosch

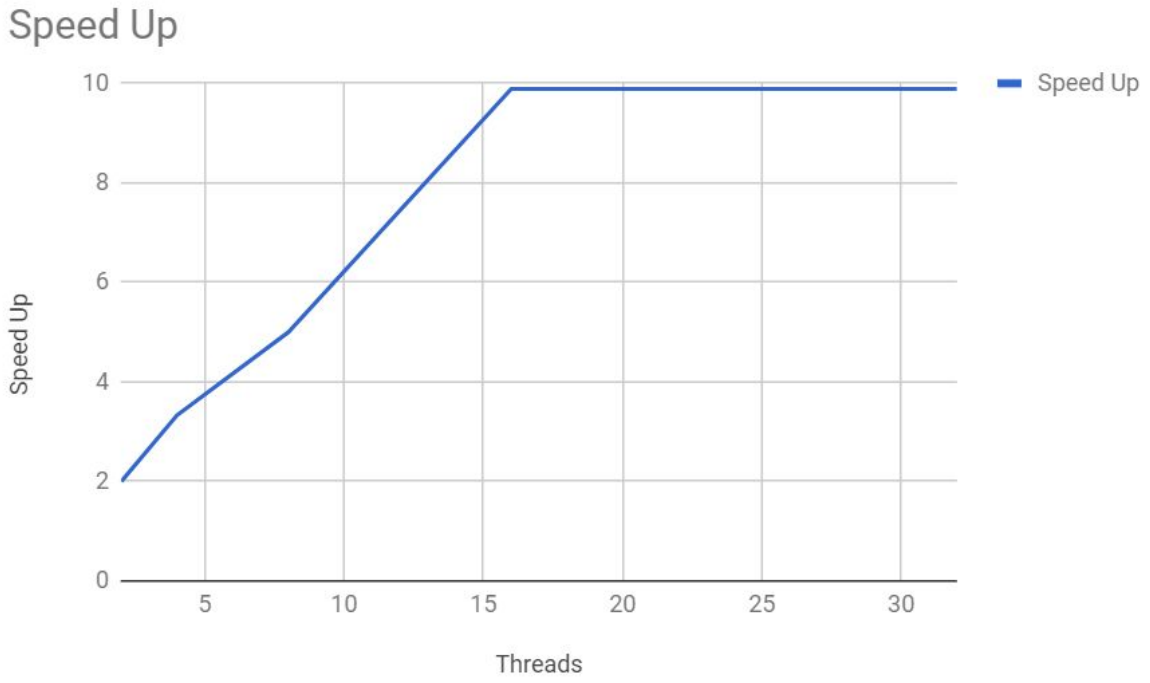
5. Complete the following table for the initial and different versions generated for 3dfft seq.c, briefly commenting the evolution of the metrics with the different versions.

Version	T1	T∞	Parallelism
seq	593772001	593705001	1.00
v1	593772001	593705001	1.00
v2	593772001	315437001	1.88
v3	593772001	108937001	5.45
v4	593772001	60012001	9.89

Amb aquesta taula podem veure que, amb les millores afegides, cada versió és més paral·lelitzable que l'anterior per tant el seu temps "amb infinits" threads és cada cop menor.

6. With the results from the parallel simulation with 2, 4, 8, 16 and 32 processors, draw the execution time and speedup plots for version v4 with respect to the sequential execution (that you can estimate from the simulation of the initial task decomposition that we provided in 3dfft seq.c, using just 1 processor). Briefly comment the scalability behaviour shown on these two plots.



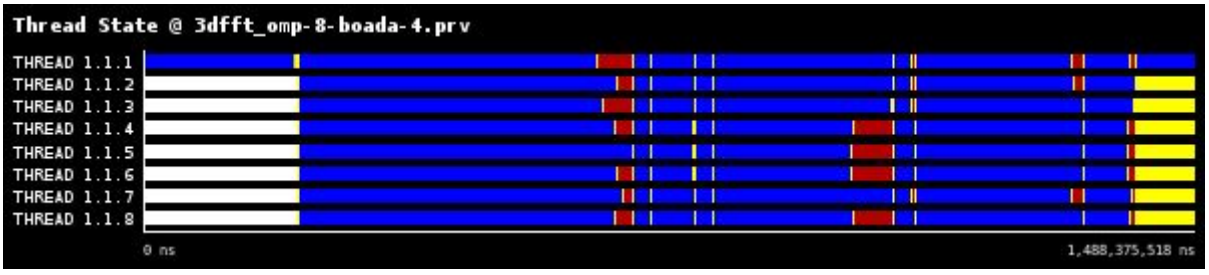


Amb aquests dos gràfics podem veure que el programa té un temps d'execució més ràpid a mesura que s'augmenten el nombre de threads fins que aquest número arriba a 16, punt en el que ja ha arribat al nivell màxim de speed up i encara que augmentem el nombre de threads, el temps d'execució i per tant el speed up es manté constant.

Tracing the execution of parallel programs

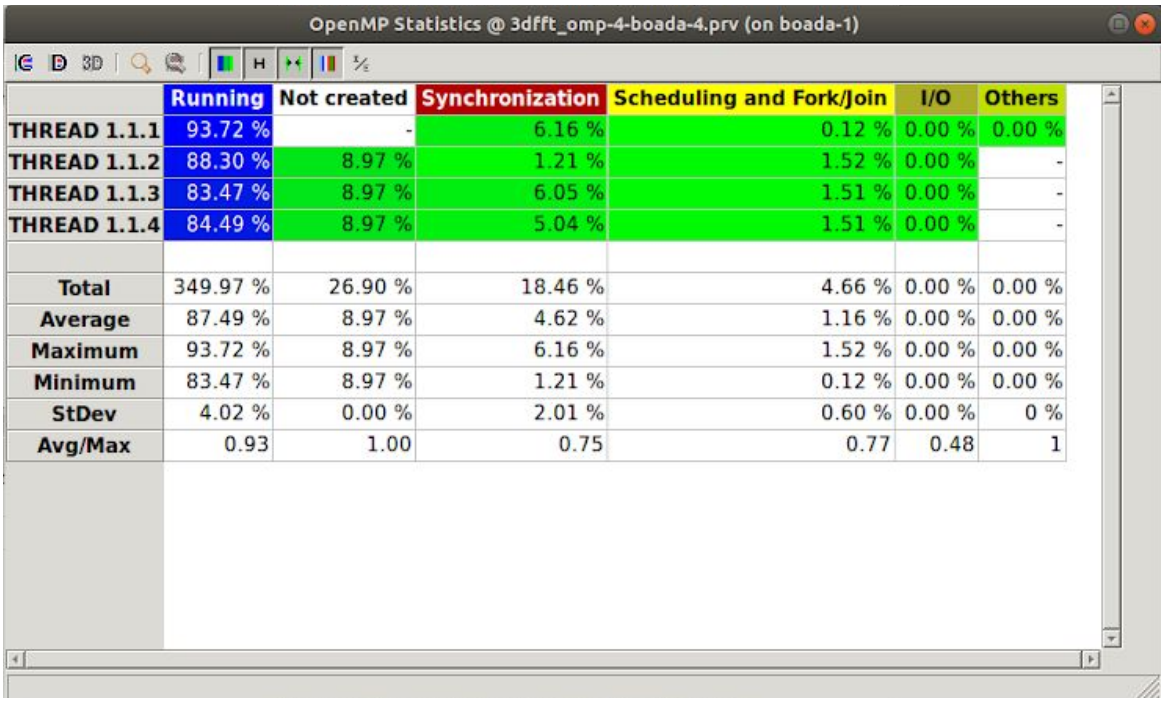
7. From the analysis with Paraver that you have done for the complete parallelization of 3dfft omp.c, explain how have you computed the value for φ, the parallel fraction of the application. Please, include any Paraver timeline that may help to understand how you have performed the computation of φ.

El valor de φ l'hem obtingut dividint la suma de tota la part paral·lelitzable entre la suma del temps d'execució del programa amb un processador.



"Versió 4, 8 threads paraver timeline"

8. Show and comment the profile of the % of time spent in the different OpenMP states for the complete parallelization of 3dfft omp.c on 4 threads.



	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	93.72 %	-	6.16 %	0.12 %	0.00 %	0.00 %
THREAD 1.1.2	88.30 %	8.97 %	1.21 %	1.52 %	0.00 %	-
THREAD 1.1.3	83.47 %	8.97 %	6.05 %	1.51 %	0.00 %	-
THREAD 1.1.4	84.49 %	8.97 %	5.04 %	1.51 %	0.00 %	-
Total	349.97 %	26.90 %	18.46 %	4.66 %	0.00 %	0.00 %
Average	87.49 %	8.97 %	4.62 %	1.16 %	0.00 %	0.00 %
Maximum	93.72 %	8.97 %	6.16 %	1.52 %	0.00 %	0.00 %
Minimum	83.47 %	8.97 %	1.21 %	0.12 %	0.00 %	0.00 %
StDev	4.02 %	0.00 %	2.01 %	0.60 %	0.00 %	0 %
Avg/Max	0.93	1.00	0.75	0.77	0.48	1

Veiem que al estar el programa completament paral·lelitzat hi ha molt poc % de temps en que els threads 2, 3 i 4 estan sense fer res, només aproximadament un 9% del que dura la execució del programa.