PARAL·LELISME

Primera Entrega

Autors:
Sergi Soriano
Mingjian Chen

Grup: 21

October 6, 2014
Facultat d'Informàtica de Barcelona

Node architecture and memory

1. Draw and briefly describe the architecture of the computer in which you are doing this lab session (number of sockets, cores per socket, threads per core, cache hierarchy size and sharing, and amount of main memory).

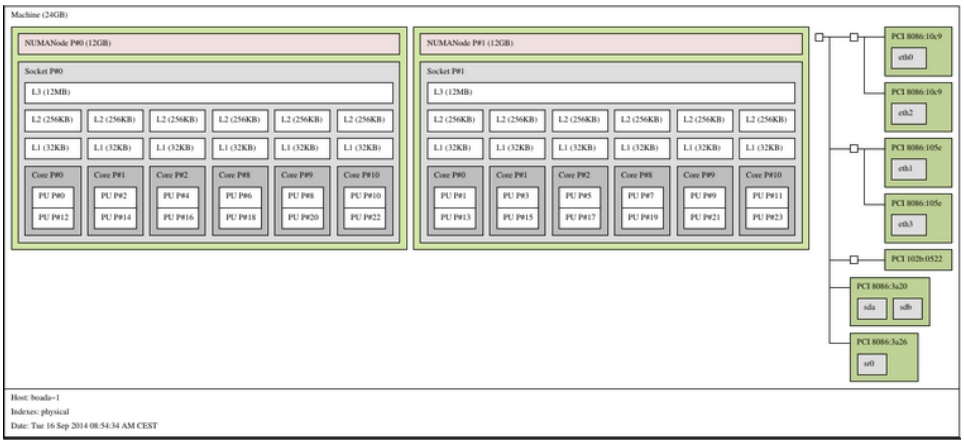


Figure 1: La arquitectura de computador

- **Nombre de sockets: 2**
- **Core per sockets: 6**
- **Thread(s) per core: 2**
- **3 cache level: L1, L2 i L3**
  - **L1d: 32K**
  - **L1i: 32K**
  - **L2 : 256K**
  - **L3 : 12288K (sharing)**
- **Memoria principal (23.49 GBytes)**

Timing sequential and parallel executions

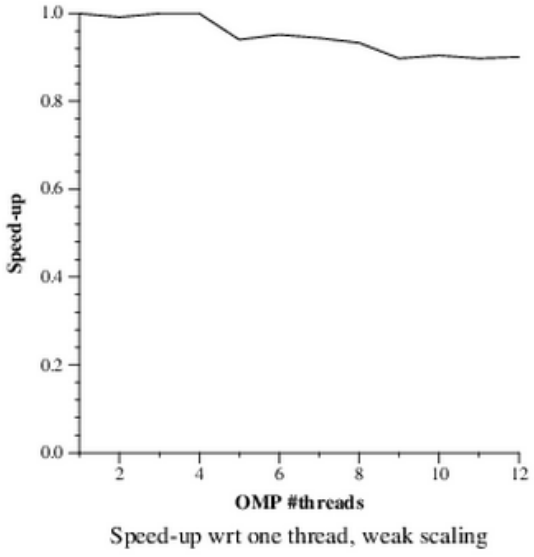
2. Indicate the library header where the structure struct timeval is declared and which are its fields.

   Es definida a `sys/time.h`**.**
   Els seus camps són: tv_sec, tv_usec

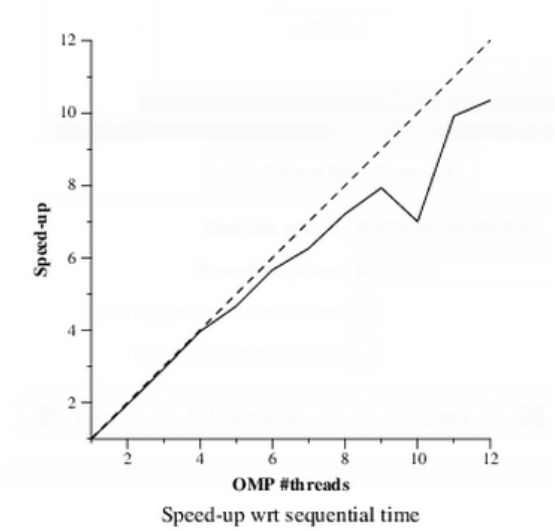
3. Plot the execution time when varying the number of threads (strong scalability) and problem size (weak scalability) for `pi_omp.c`. Reason about how the scalability of the program. Tracing sequential and parallel executions.

   Amb el weak scaling podem veure que amb els mateixos threads i augmentant les iteracions, el speed-up que podem conseguir podem dir que és més o menys el mateix.

   En canvi amb el strong scaling, augmentant els threads i una única entrada de iteracions, podem veure que el speed-up que aconseguim a mesura que augmenta les iteracions es més alt, ja que aprofita millor els threads. En el cas del strong scaling veiem que als 7 threads es produeix un overhead.



(a) Weak scalability



(b) Strong scalability

Tracing sequential and parallel executions

4. From the instrumented version of pi seq.c, and using the appropriate **Paraver** configuration file, obtain the value of the parallel fraction for this program when executed with 100.000.000 iterations.

   El percentatge de la fracció que s'executa en paralel és del 96.63%.

5. From the instrumented version of **pi omp.c**, and using the appropriate Paraver configuration file, show a profile of the % of time spent in the different OpenMP states **ONLY** during the execution of the parallel region (not considering the sequential part before and after) when using 8 threads and for 100.000.000 iterations.

	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	92.21 %	5.59 %	1.91 %	0.27 %	0.01 %	0.00 %
THREAD 1.1.2	93.36 %	6.55 %	0.08 %	-	0.01 %	-
THREAD 1.1.3	93.19 %	6.55 %	0.25 %	-	0.00 %	-
THREAD 1.1.4	93.23 %	6.55 %	0.21 %	-	0.00 %	-
THREAD 1.1.5	91.13 %	6.55 %	2.31 %	-	0.00 %	-
THREAD 1.1.6	93.19 %	6.56 %	0.25 %	-	0.00 %	-
THREAD 1.1.7	91.05 %	6.55 %	2.39 %	-	0.00 %	-
THREAD 1.1.8	93.27 %	6.56 %	0.17 %	-	0.00 %	-
Total	740.64 %	51.47 %	7.57 %	0.27 %	0.05 %	0.00 %
Average	92.58 %	6.43 %	0.95 %	0.27 %	0.01 %	0.00 %
Maximum	93.36 %	6.56 %	2.39 %	0.27 %	0.01 %	0.00 %
Minimum	91.05 %	5.59 %	0.08 %	0.27 %	0.00 %	0.00 %
StDev	0.92 %	0.32 %	0.98 %	0 %	0.00 %	0 %
Avg/Max	0.99	0.98	0.40	1	0.41	1

Figure 3: % Time spent in different states

6. From the instrumented versions of **pi_seq.c** and **pi_omp.c** (executed with the same number of iterations), and using the appropriate Paraver configuration files, obtain the average number of instructions executed and MIPS per thread during the execution of the parallel region (when using 8 threads). Are the numbers obtained coherent?

	Running	Not created	Synchronization	Scheduling and Fork/Join	I/O	Others
THREAD 1.1.1	2,757.76	24.35	771.97	237.69	80.66	24.35
THREAD 1.1.2	2,609.47	2,609.57	24.28	-	115.82	-
THREAD 1.1.3	2,614.67	2,614.78	1,373.91	-	292.36	-
THREAD 1.1.4	2,613.80	2,613.90	1,367.74	-	305.30	-
THREAD 1.1.5	2,661.31	2,661.52	745.52	-	307.92	-
THREAD 1.1.6	2,614.74	2,614.84	1,380.60	-	301.41	-
THREAD 1.1.7	2,663.95	2,664.26	721.10	-	326.81	-
THREAD 1.1.8	2,612.87	2,612.98	1,351.16	-	327.98	-
Total	21,148.56	18,416.21	7,736.29	237.69	2,058.27	24.35
Average	2,643.57	2,302.03	967.04	237.69	257.28	24.35
Maximum	2,757.76	2,664.26	1,380.60	237.69	327.98	24.35
Minimum	2,609.47	24.35	24.28	237.69	80.66	24.35
StDev	47.99	861.14	458.41	0	92.93	0
Avg/Max	0.96	0.86	0.70	1	0.78	1

Figure 4: pi_omp

	Running	Not created	I/O	Others
THREAD 1.1.1	3,555.89	23.17	92.16	23.17
Total	3,555.89	23.17	92.16	23.17
Average	3,555.89	23.17	92.16	23.17
Maximum	3,555.89	23.17	92.16	23.17
Minimum	3,555.89	23.17	92.16	23.17
StDev	0	0	0	0
Avg/Max	1	1	1	1

Figure 5: pi_seq

No són coherents. Com bé ens va comentar el professor de laboratori, ve donat per un problema en el contador hardware de la màquina.

Visualizing the task graph and data dependences

7. Include the source code for function **dot_product** in which you show the `Tareador` instrumentation that has been added to study the potential parallelism in the code. This instrumentation has to appropriately define tasks and filter the analysis of variable(s) that cause the dependence(s).

```
//dot_product.c
void dot_product (long N, double A[N], double B[N],
    double *acc){
  double prod;
  tareador_disable_object(&*acc);
  *acc=0.0;
  int i;
  for (i=0; i<N; i++) {
    tareador_start_task("loop_dot");
    prod = my_func(A[i], B[i]);
    *acc += prod;
    tareador_end_task("loop_dot");
  }
  tareador_enable_object(&*acc);
}
```

8. Capture the task dependence graph and execution timeline (for 8 processors) for that task decomposition.

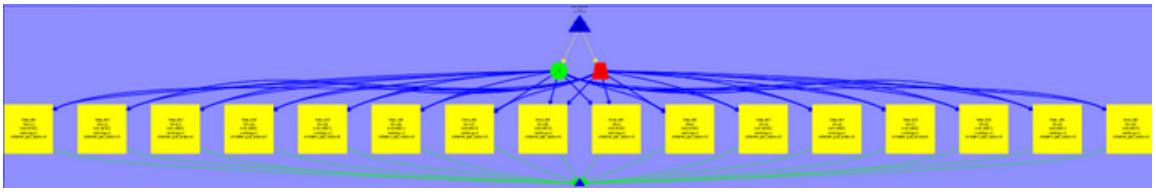


Figure 6: Figure task dependence grahp `dot_product.c`

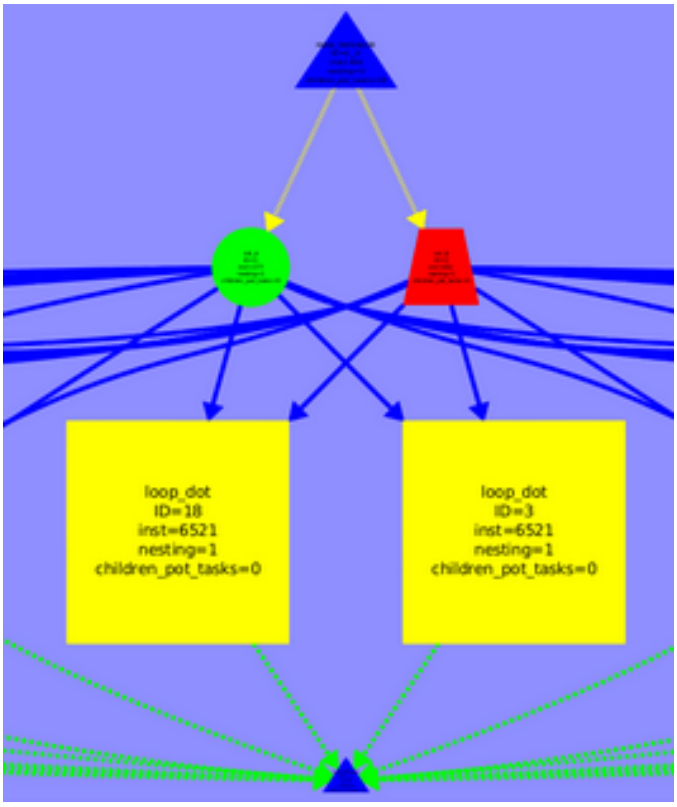


Figure 7: Figure task dependence grahp ampliado `dot_product.c`

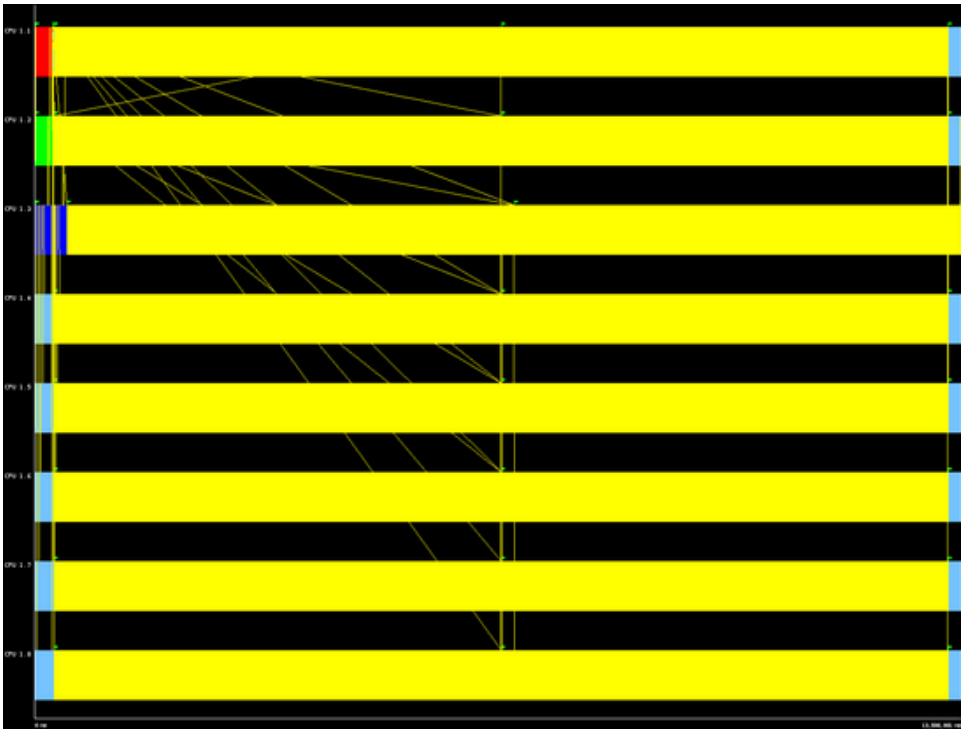


Figure 8: Figure timeline `dot_product.c`

Analysis of task decomposition

9. Complete the following table for the initial and different versions generated for `3dfft_seq.c`.

Version	T_1	T_∞	**Parallelism**
sqe	593762	593747	1.000
v1	593762	593747	1.000
v2	593762	315188	1.884
v3	593762	108336	5.481
v4	593762	59412	9.994

10. 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).

Processors	**Time (ns)**	**Speed-up**
1	593,762,001	1
2	297,297,001	1.997
4	156,077,001	3.804
8	86,558,001	6.860
16	59,412,001	9.994
32	59,412,001	9.994

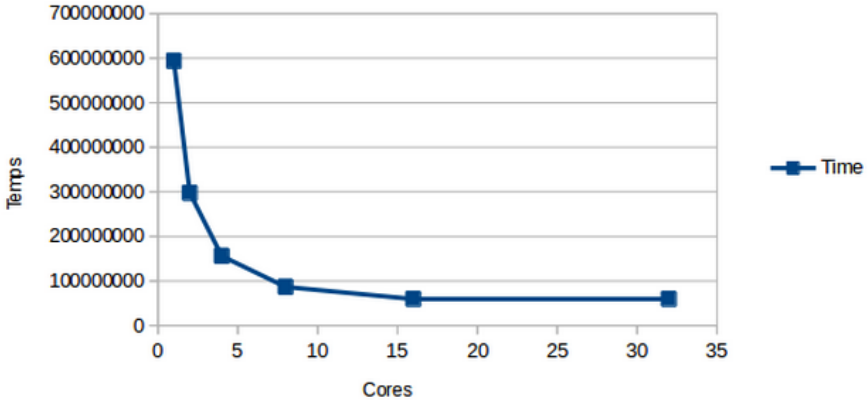


Figure 9: Figure execution timeline en funció del nombre de cores

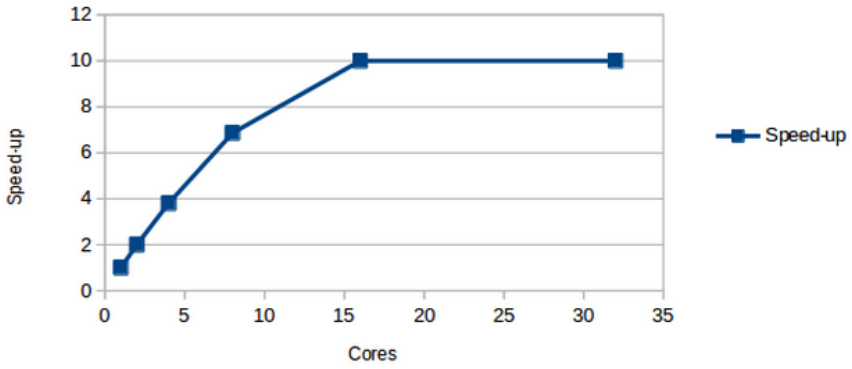


Figure 10: Figure Speed-up en funció del nombre de cores