

Lab1 deliverable

Daniel Sattler and Cristina Raluca Vijulie

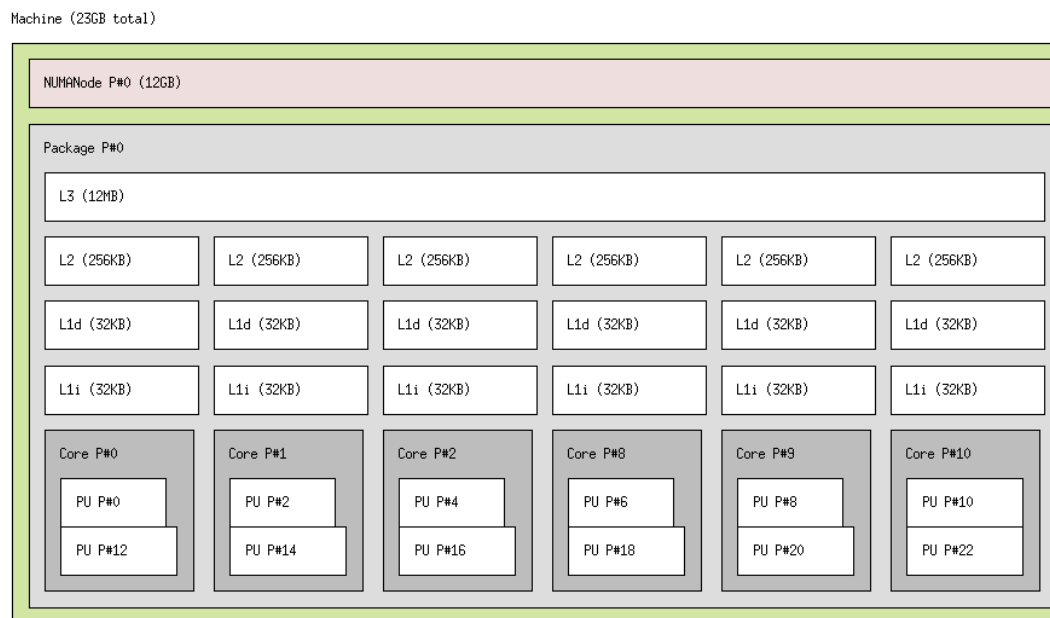
par1207

October 12, 2017

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).

- (a) sockets: 2
- (b) cores/socket: 6
- (c) threads: 2



Timing sequential and parallel executions

2. Describe what do you need to add to your program to measure the elapsed execution time between a pair of points in the program, clearly indicating the library header file

that needs to be included, the library functions that need to be invoked, the data structure and its fields.

To measure the time we need to get a timestamp with the time of the day when the execution starts and subtract that value to the time of day when the program ends. To do so we use the following:

- The library:

```
#include <sys/time.h>
```

- The function:

```
int gettimeofday(struct timeval *, void *);
```

- The data structure **timeval**, which includes the following fields:

```
time_t      tv_sec      seconds
suseconds_t tv_usec     microseconds
```

3. Plot the speedup obtained when varying the number of threads (strong scalability) and problemsize (weak scalability) for pi omp.c. Reason about how the scalability of the program.

Visualizing the task graph and data dependences

4. 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).

5. Capture the task dependence graph for that task decomposition and the execution timelines (for 8 processors) that allow you to understand the potential parallelism attainable. Briefly comment the relevant information that is reported by the tools.

Analysis of task decompositions

6. 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	T_1	T_∞	Parallelism
seq			
v1			
v2			
v3			
v4			

7. 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.

Tracing sequential and parallel executions

8. 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, showing the steps you followed to obtain it. Clearly indicate which Paraver configuration file(s) did you use.

9. 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 when using 8 threads and for 100.000.000 iterations. Clearly indicate which Paraver configuration file(s) did you use and your own conclusions from that profile.