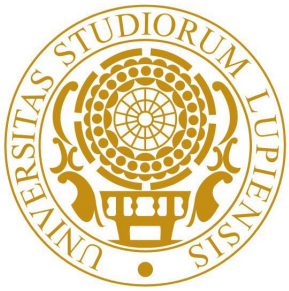
UNIVERSITÀ DEL SALENTO



Facoltà di Ingegneria

Corso di Laurea Magistrale in Computer Engineering

High Performance Computing Project

Cache dimension estimation

Professor: **Giovanni Aloisio**

Students: **Capoccia Leonardo,**
Basile Davide,
Corvaglia Salvatore

Academic year 2018/2019

Abstract

During the first project, we have built a diagram for the variation of the peak performances with respect to the variation of the array size used in the script. We recognized that the performance variation is due to the memory hierarchy inside modern computers' architectures.
The second project is the creation of the roofline model used to understand how the performance of a given program can be improved acting on memory access, code refactoring and the usage of different data structures.

Chapter 1

Cache size estimation

During the first project, we used a benchmark program called "Schonauer Triad Benchmark" to estimate the size of the cache levels in the computer architecture under scrutiny.

1.1 Schonauer Triad Benchmark

The code of the Schonauer Triad Benchmark is very simple and it is structured as three different *for* loops, one of which in the *main* routine. At each iteration, the *simulation* routine is called. In this routine, there are two nested *for* loops (the outer loop with the index going from 1 to R, the inner going from 1 to N). In the outer loop, a *dummy* routine is called to prevent the compiler from optimizing the code removing the inner loop from the outer: this is because the inner loop is not dependent from the outer *for* loop index. The loops run for a total number of times which is almost constant at each iteration, making it possible to calculate the peak performances. The ideal case is a case in which the peak performances are constant, but this is not a real case considering the cache hierarchy existing in the actual computers' architecture.
Each time the *simulation* routine is called the time elapsed inside the function is recorded and printed into a file alongside the **MFLOPS** calculated as:

$$\textbf{MFLOPS} = \frac{2 \times R \times N}{10^6}$$

The multiplicative factor is given from the operations done in inner simulation loop, where there is a sum and a product. The values are then saved inside a *CSV* file that can be used to print the diagram.

Algorithm 1 Operations done in the inner simulation loop

```
1: A[i] = B[i] + C[i]*D[i];
```

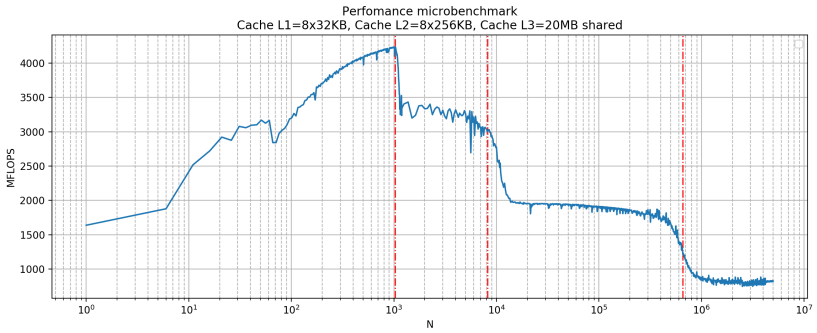


Figure 1.1: Performances of Intel Xeon E5-2670@2.6GHz (Sandy Bridge). Red lines represent different cache sizes, respectively L1, L2 and L3 as reported in the datasheet.

1.2 Printing of the results and size estimation

The plot is obtained from the *CSV* file produced by the program as output. From the graph is possible to see four different plateau, which represent the cache levels. In the architecture used there are present 3 different levels of cache (L1, L2 and L3). The size of each cache level can be estimated using a simple formula:

$$L_i = \frac{N_i * 4 * 8}{1024}[KB]$$

where N_i is the number where a fall in the graph begins and it represents the size of each of the 4 array instantiated and 8 is the size in bytes of each item in the array (each item is a real number). L_i is the size of the i-th cache level.
Let's see in our graph the caches' sizes:

$$L_1 = \frac{1035 * 4 * 8}{1024} = 32.34[KB]$$

$$L_2 = \frac{7992 * 4 * 8}{1024} = 243.5[KB]$$

$$L_3 = \frac{565895 * 4 * 8}{1024} = 17.68[MB]$$

1.3 Final taught and conclusion

We can learn different things about the underlying computer architecture by studying the Fig. 1.2:

1. When N is very small, the performances are low because the processors' pipes are not full, which translates to low efficiency and consequently to a loss in the performances.

2. After the point in which the pipes are full, the level 1 cache starts to be filled with the arrays, which brings the highest performance possible. This is true until the L1 cache is completely saturated.

3. The L1 cache is completely full, so the L2 cache starts to be used, which is less performant and slower than the L1, but bigger in terms of size. This phase is represented in the plateau with an average peak performance of about 3200 MFLOPS.

4. When the L2 cache is full, the L3 starts to be used. This is the last level of cache inside the architecture in use, but it is shared among all the processors.

5. After the last phase, the arrays are too big to fit in the caches, so the slower RAM is used, which bring an additional loss of performances. The performance reached is constant for bigger values of N.

One last important thing to be carried at our attention is that the L3 cache is shared, so the value found by calculation is reasonable considering multiprogramming problems, which are problems related to multiple programs running at the same time on the same (or different) processor.

As a conclusion, we can state that for heavy optimization of a program we need to optimize the cache hit rate of the program being optimized which brings the data to the processor faster delivering high performances. To do this we need to have insights about the underlying computer architecture.