INF-4101C

Practical session (2h)

Evaluation of cache memory performances

Objectives:

1. To be able to estimate the performances of cache memory;
2. To understand the cache memory parameters (size, line, associativity);
3. To understand the programming style consequences on the efficiency the cache memory;

Exercises:

Handling of tools of evaluation of the cache memory performances

The measures of the cache memory performances will be realized by means of the freeware Valgrind and its tool cachegrind (http: // valgrind.org/). Notice that to be able to use it, the program must be compiled with the option -g of gcc compiler. To call cachegrind, it is necessary to write the following commande in the prompt :

valgrind – tool=cachegrind ./ nom_programme < options of the program >

After the execution, a part of the results is printed on the screen: estimated statistics of every level of the cache memory (Figure 1). At the same time, a file will be created with the name cachegrind.out.3442 (3442 is the PID of the analyzed process). In this file, you will find the parameters of the cache memory considered for simulation as well as the collected events (Figure 2).

```
==3442== I   refs:      325,695,314
==3442== I1  misses:          1,084
==3442== L2i misses:          1,071
==3442== I1  miss rate:        0.00%
==3442== L2i miss rate:        0.00%
==3442==
==3442== D   refs:      169,916,905  (120,559,278 rd   + 49,357,627 wr)
==3442== D1  misses:        625,845  (   313,529 rd   +    312,316 wr)
==3442== L2d misses:         11,348  (     1,548 rd   +      9,800 wr)
==3442== D1  miss rate:         0.3% (       0.2%     +        0.6%  )
==3442== L2d miss rate:         0.0% (       0.0%     +        0.0%  )
==3442==
==3442== L2 refs:           626,929  (   314,613 rd   +    312,316 wr)
==3442== L2 misses:          12,419  (     2,619 rd   +      9,800 wr)
==3442== L2 miss rate:          0.0% (       0.0%     +        0.0%  )
```

Figure 1: Estimated statistics example

```
desc: I1 cache:         32768 B, 64 B, 8-way associative
desc: D1 cache:         32768 B, 64 B, 8-way associative
desc: L2 cache:         6291456 B, 64 B, 24-way associative
cmd: ./test2 im2.pgm 3
events: Ir I1mr I2mr Dr D1mr D2mr Dw D1mw D2mw
```

Figure 2: Example of cachegrind.out file

Exercise 1:

Write a program in c programming language which realizes a dynamic allocation of two matrices, matrix A and matrix B each of the size at least 1000 x 1000 elements (int type) and are allocated as 1-D array. This program is going to:

- Initialize the matrix A with a zero and to copy out the elements of the matrix A in B. You have to estimate three versions of this program:
- **Version 1**: read the matrices elements column by column, **without** use of any optimized functions (memcpy or the other);
- **Version 2:** read the matrices elements line by line, without use of any optimized functions (memcpy or the other);
- **Version 3:** Use optimized functions i.e. memcpy, calloc, etc to realize the objectives of the program.

Tasks :

1. Identify the parameters of the system cache memory of your PC and represent if graphically.
2. For every version of the program, make the evaluation of the performances of cache memory, analyze it and compare it.

Exercise 2

For the version 2 and 3 of the program of the preceding exercise, initialize the matrix A with random integer numbers. The matrix B will contain mean values of 3x3 neighborhood for each element.
We will procced to changes of the parameters of the level D1 of the cache memory and analyze the consequences on the performances (expressed in the total number of "misses "). For each of both versions, fill a table of the results (Tab. 1), create also the graphs (axis x: size of memory D1, axis: total data misses). What can we conclude?

Attention, for these measures, automate the tests, that is, for example, create script " shell " who allows launching sets of tests with various parameters, and at the same time pre-process the results. The objective is to facilitate their exploitation during the writing of the report. You can be inspired an available example.

Use the following command to vary D1 level parameters:

valgrind –-tool=cachegrind –-D1=mem_size, associativity, line_size ./nom_programme <options>

Tab. 1 : Measured Cache performances example (number of misses)

Input Data Size	D1 cache size				
Block line 16 B	1k	2k	4k	8k	16k
1-way assoc.					
2-way assoc.					
4-way assoc.					
8-way assoc.					

You can compare the obtained result with Fig.3. Can you observe the same phenomena?

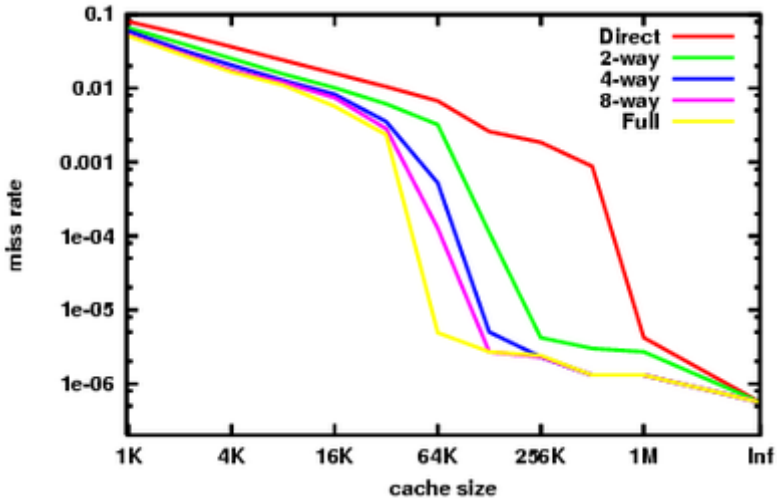

Figure 1 cache performance seen on the Integer portion of the SPEC CPU2000 benchmarks [1]

References:

[1] Hill and Cantin, *"Cache performance of SPEC CPU2000". Cs.wisc.edu. http://research.cs.wisc.edu/multifacet/misc/spec2000cache-data/ and https://en.wikipedia.org/wiki/CPU_cache*