Roofline and Matrix Multiplication PAPI Study

Jose Alves, Rui Brito

Universidade of Minho

Resumo

This paper describes the roofline analysis of two different machines. The roofline consists in a theoretical performance model that provides visual insights on floating point computing performance. This model relates processor performance to off-chip memory traffic, giving a upper bound on performance of a kernel depending on its operational intensity. In this paper is also a study of a matrix multiplication using PAPI(Performance Application Programming Interface). With PAPI we are able to use the low-level hardware counters to identify the most accessed areas of our hierarchy of memomery as well as the operations made by our software. With the results we will present the roofline model of the machine used for the study with the conclusions obtained.

I. Introduction

The importance of a system's characterization is due to the need of understanding the limitations of a system, in order to make the necessary optimizations to obtain the maximum performance of a system. In order to provide a easy-to-understand performance model for floating-point programs, the Roofline was created. The Roofline indentifies an theoretical upper bound on performance on a kernel that relates if a certain opetional intensity is memory-bound, dependant on accesses of memory, or computational-bound, dependant on floating-point operational units. To comprehend the influence of different components and certain architecture characteristics a number of ceilings, both computational and memory, may be added. Since the Roofline model is better understood with a case study, a matrix multiplication problem was proposed to present the theoretical limitations of the algorithm through the model. To specify the problem in the Roofline model, a series of values were obtain from the hardware counters through PAPI(Performance Application Programming Interface). PAPI allow us to reach the low-level hardware counters and obtain information from components accessed by the program.

II. Roofline

In order to calculate the rooflines, we needed the Floating-Point(FP) Performance Peak and the Memory Bandwidth's Peak. The FP Performance Peak is obtained through the CPU information. The number of cores, their clock's frequency and the amount of SIMD(Single Instruction, Multiple Data) they can do simultaneous, provides a good approxiamation of the Performance Peak for Floating-Point operations. To attain the FP Performance Peak we calculate the following formula:

$$\text{GFlop/s}_{\text{max}} = \#_{\text{cores}} \times f_{\text{clock}} \times \#_{\text{SIMD}}$$

MacBook Pro FP Performance Peak:

$$\text{GFlop/s}_{\text{max}} = 2 \times 2.8 \times 8 = 44.8 GFLOPSs$$

HP Pavillion FP Performance Peak:

$$\text{GFlop/s}_{\text{max}} = 4 \times 1.6 \times 8 = 51.2 GFLOPSs$$

The Memory Bandwitdh Peak, being independant of the number of FP Operations per second, is a 45 degree line that demonstrates the maximum FP Operations that can be done for a given Operational Intensity. This peak is given by the memory limitations, their frequency, number of channels and bus bandwidth. To calculate the Memory Bandwidth Peak we resolve the following formula:

$$\mathrm{BW}_{\max} = \#_{\text{channels}} \times mem_{\text{clock}} \times bus_{\text{bandwidth}}$$

MacBook Pro Memory Bandwidth Peak:

$$\mathrm{GFlop/s}_{\max} = 2 \times 1067 \times 64 = 17.072 GBbyte$$

HP Pavillion Memory Bandwidth Peak:

$$\mathrm{GFlop/s}_{\max} = 2 \times 1333 \times 64 = 21.328 GBbyte$$

III. Machines' Characteristics

The specifications of the Machines used to sample the Roofline Model are displayed on Table 1.
The HP Pavillion dv6-2190ep was also used to run the tests the matrix multiplication test case.

The machines Roofline models were calculated through the values explained above. As we can see through Figure 1 and Figure 2 the HP Pavillion Roofline is somewhat higher than the MacBook Pro. This can be explain through the difference in the specifications between both machines.

IV. PAPI Case Study

The PAPI case study made in this project, was to analyse the performance of a **matrix multiplication** algorithm,

$$MatrixA * MatrixB = MatrixC \qquad (1)$$

wich contains a triple nested loop with the indexes i,j and k(line,column and position). Our implementation will explore the index order **i,j,k** of the triple nested loop.
The algorithm of matrix multiplication is presented here, in order to better understand the problem at hand.

```
for (i = 0; i < size; i++) {
    for (j = 0; j < size; j++) {
        for(k = 0; k < size; k++) {
            acc += matrixA[i][k] * matrixB[k][j];
            }
            matrixC[i][j] = acc;
            acc = 0;
        }
    }
```

As we can observe, the algorithm obtains the result of each element of matrix C through the multiplication of the line from matrix A with the corresponding column in the matrix B. As defined in the program, matrices are an array of arrays, so when is required a new element of the matrix, it fetch a line of contiguous elements from the main memory, bringing a line from the matrix. Obviously this is prejudicial to matrix B since it needs the elements of a column and not from a line, to fix this problem we expirement running the test for both the naive implementation, and one that calculates the transpose matix B and runs the algorithm throught it.
To note that he implementation produced to calculate the matrix multiplication was made in C and compiled with Optimization level 1 (-o).

IV.1 PAPI Specifications

To measure the implementation's perfomance, hardware counters were used. To gather the information of these counters, as stated above, PAPI was used. Although able to obtain the low-level hardware counters results, is still needed to identify wich to gather as well as make sure they are available in the system. To obtain

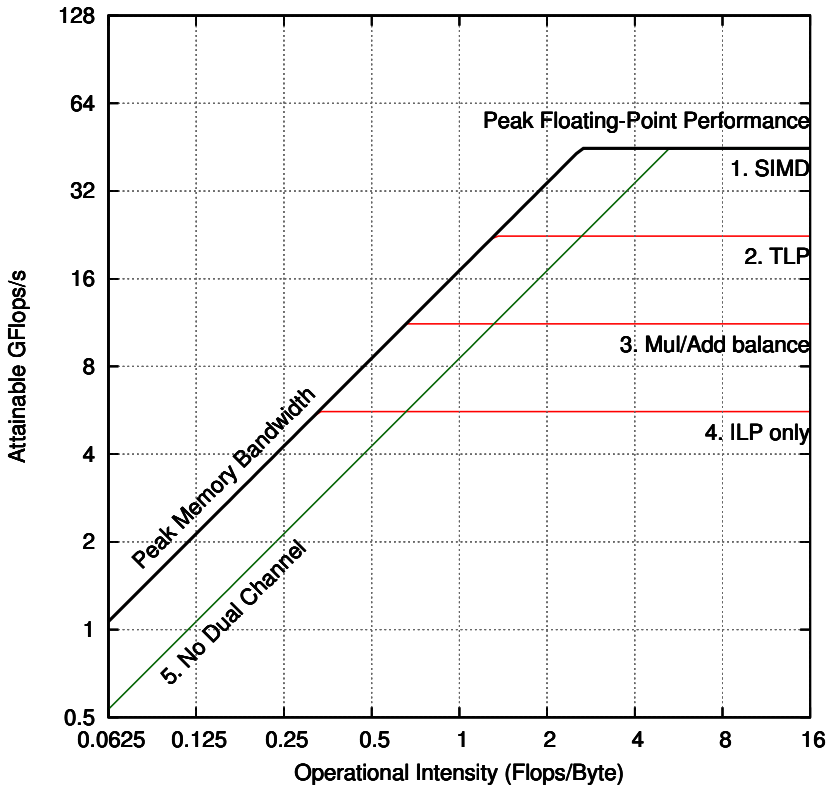

Figura 1: *Mackbook Pro late 2008*

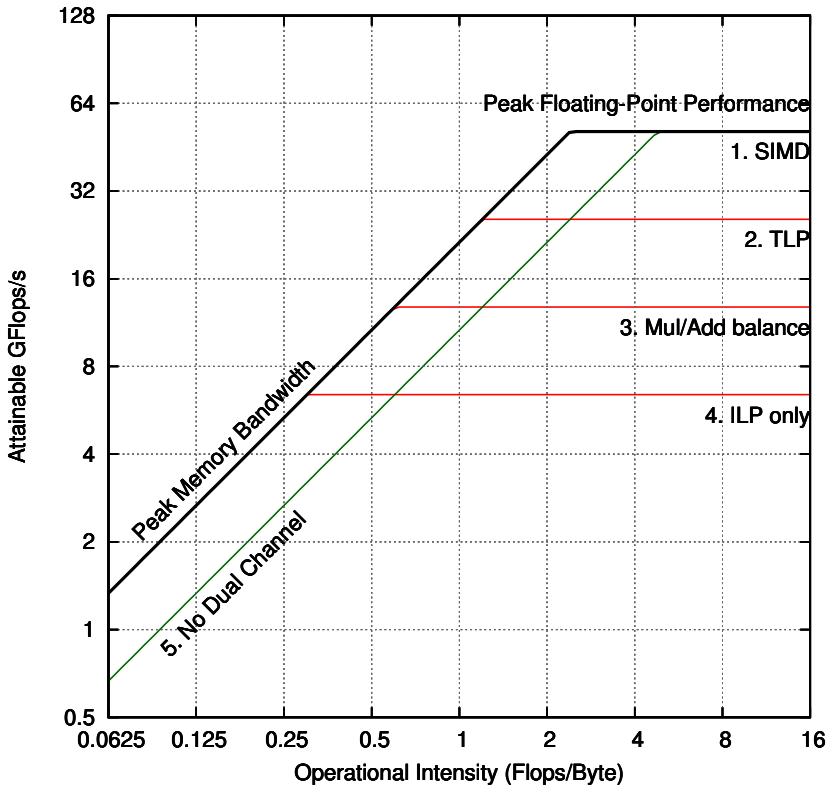

Figura 2: *HP Pavillion dv6-2190ep*

	MacBook Pro late 2008	Pavillion dv6-2190ep
Manufacter:	Apple	HP
Processor		
Manufacturer:	Intel	Intel
Arch:	Core	Nehalem
Model:	Core 2 Duo T9600	i7-720QM
Cores:	2	4
Clock Frequency:	2.80 GHz	1.60 GHz
FP Performance's Peak:	44.8 GFlops/s	51.2 GFlops/s
Cache		
Level:	1	1
Size:	32KB + 32KB	32KB + 32KB
Line Size:	64 B	64 B
Associative:	8-way	4/8-way
Memory Access Bandwidth:	40 GB/s	22 GB/s
Level:	2	2
Size:	6 MB	256 KB
Line Size:	64 B	64 B
Associative:	24-way	8-way
Level:	-	3
Size:	-	6 MB
Line Size:	-	64 B
Associative:	-	12-way
RAM		
Type:	SDRAM DDR3 PC3-8500	SDRAM DDR3 PC3-10600
Frequency:	1067 MHz	1333 MHz
Size:	4 GB	4 GB
Num. Channels:	2	2
Latency:	13.13 ns	13.5 ns

Tabela 1: *Machines' specifications*

the needed information, the following counters were used:

PAPI _TOT_CYC Total number of cycles;

PAPI _TOT_INS Instructions completed;

PAPI _LD_INS number of load instructions;

PAPI _SR_INS number of store instructions;

PAPI _FP_OPS Floating point operations;

PAPI _FP_INS Floating point instructions;

PAPI _L1_DCA L1 data cache accesses;

PAPI _L1_DCM L1 data cache misses;

PAPI _L2_DCA L2 data cache accesses;

PAPI _L2_DCM L2 data cache misses;

PAPI _L3_DCA L3 data cache accesses;

Four tests, as we can see in Table 2, were chosen to run in the two different version(naive and transpose). Each test was run four times, with the best execution time being select as long as the range was no larger than five per cent from the other three. To measure the influency of accessing each layer of the memory hierarchy each test fits in a different level(L1, L2, L3 and RAM).

Memory Level	Size of
L1	32 KB
L2	256 KB
L3	6 MB
RAM	7.68 MB

Tabela 2: *Test cases*

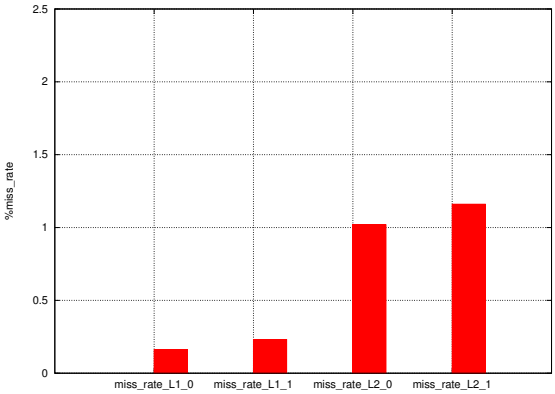

Figura 3: *Percentage of Cache Misses*

V. Results

To measure the data cache misses the counters PAPI_L1_DCM and PAPI_L2_DCM were used.

The graphic shows an increase of percentage of misses with the optimized version, they are misleading. The graph shows that although the percentage of misses increased, the total of misses didn't because the number of accesses also droped.

Usage of both levels of cache was estimated with specific counters. PAPI_L1_DCA and PAPI_L2_DCA provided the number of data accesses to the caches.

Before the results were out, it was expected a decrease of cache accesses from version one to version two of the algorithm.

As we can see in graph, the number of access to cache drops significantly from the first version to the second version while running with the L1 Cache Test. Though in the second test, the L2 Cache, the number of accesses slightly increased.

Referências

[1] `Roofline: An insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures`
Samuel Webb Williams, Andre Waterman, David A. Patterson
23th November 2012

[2] `http://ark.intel.com/products/35563/Intel-Core2-Duo-Processor-T9600-6M-Cache-2_80-GHz-1066-MHz-FSB`
Intel® Core ™ 2 Duo Processor T9600 (6M Cache, 2.80 GHz, 1066 MHz FSB)
©Intel Corporation
23th November 2012

[3] `http://ark.intel.com/products/43122/Intel-Core-i7-720QM-Processor-6M-Cache-1_60-GHz`
Intel® Core ™ i7-720QM Processor(6M Cache, 1.60 GHz)
©Intel Corporation
23th November 2012

[4] `http://www.crucial.com/store/ListParts.aspx?model=MacBook%20Pro`

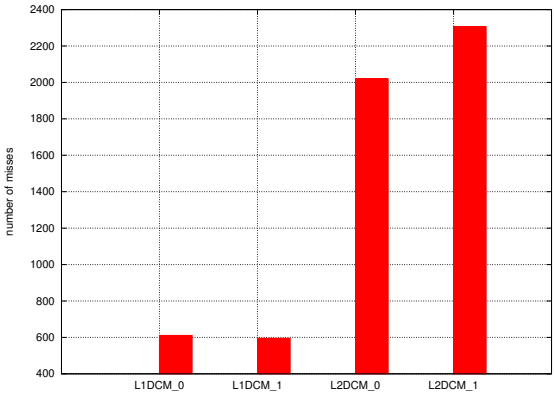

Figura 4: *Number of Cache Misses*

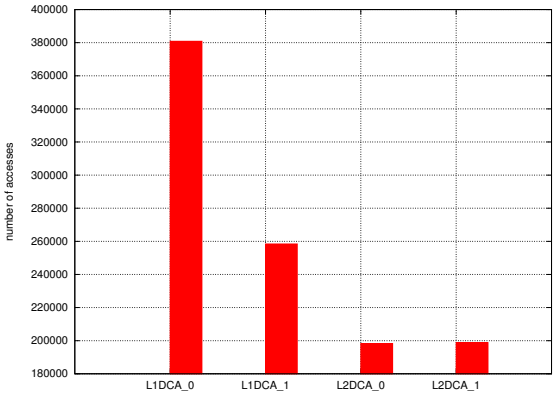

Figura 5: *Number of Memory Accesses*

%20%28Early%202008%20and
%20Late%202008%29
Computer memory upgrades for Apple MacBook Pro (Early 2008 and Late 2008) Laptop/Notebook from Crucial.com
© Micron Technology
24th November 2012

[5] http://www.crucial.com/store/listparts.aspx?model=Pavilion%20dv6-2190ep&Cat=RAM
Computer memory upgrades for HP - Compaq Pavilion dv6-2190ep Laptop/Notebook from Crucial.com
© Micron Technology
24th November 2012