ISSN 2047-3338

Dependence of Memory Hierarchy Footprint of Intel's Multicore Processors on Applications by Regression Analysis

Mahmoud ASKARI

Belarusian State University of Information and Radioelectronics, Minsk, Belarus
Department of Computer, Damavand Branch, Islamic Azad University, Damavand, Iran
mahmoudaskari@damavandiau.ac.ir

***Abstract*— The gap between speed of processor and main memory is reduced using parallel systems and multi–core processors are becoming more popular in this direction. Analyzing system performance and the factors affecting on it helps to improve hardware base and operating system managing the calculating processes. The idea of this paper is to investigate the dependence of the miss rate at each level of the memory hierarchy on the type of application. Research has been done on three different models of the Intel architecture including: Core i5–460M Nehalem, Intel Core i5–3317U Ivy–Bridge and Intel Core 2 T5800. Method is based on Stepwise regression analysis on counted events of SPEC CINT2000 benchmarks. The results show that the miss rate in each component of the memory hierarchy is dependent on the type of application and its importance is not dependent on the local of member of memory hierarchy. In this paper, after an introduction, the related works and background of problem are discussed. The experimental methodology and statistical analysis are discussed in chapters 4 and 5. Finally, conclusions and future work are described.**

***Index Terms*— Memory Hierarchy, Intel Processor, SPEC CPU2000 and Regression Analysis**

I. INTRODUCTION

STUDY of factors that affect the efficiency of the computer system memory is important and identifying its bottlenecks may give renewed impetus to the further development of memory hierarchy. This paper evaluates the impact of various applications on miss rate of Translation Look aside Buffer (TLB) and cache levels of the memory hierarchy on three Intel's multi–core processors. Statistics gathered from experiments show that miss rate on the components of the memory hierarchy essentially depends on the type of application. Accordingly, the authors believe that type of application should be considered at the operating system level. That is executing the applications of specific class operating system could be able to control the miss rate based on disabling some of the components of the memory hierarchy, pre–fetching, hyper–threading. This study can be taken into account in developing operating systems and in identify the most important levels of hierarchical memory that speeds up or in the contrary decelerate computing process.

Due to the efforts to increase efficiency and system developers need to study this area, Performance Monitoring Unit (PMU) provides a wide range of countable events. To study the statistics provided by this unit modeling and statistical techniques are exploited.

Intel has initiated a study on multi–core systems since 2005 [1]. The multi–core processors make a hierarchy of memory levels. Nowadays, a vendor places all of these levels directly into the processor, and in few processors one–two levels are placed on the board. Components of the memory hierarchy are included caches and TLB. Examples of such Intel products are Core, Nehalem and Ivy–Bridge. All of them can be seen in two to three level of caches and one to two level of TLBs. Cache memory is a fast memory and TLB translates virtual addresses sent by core's program counters to memory physical addresses. In this paper, processors have two cores and different memory structures, which are described in Background section.

Discussion of graphics processor and memory cache resizing is off in this study. In addition, the pre–fetching and multi–threading technology was disabled in our experiments and has no effect on the miss rates and performance on this study.

II. RELATED WORKS

This section shows the position of this paper in landscape of other related works. Hussam Mousa et al. [2] have used multi–linear regression model and model–tree design to analyze the Cycle per Instruction (CPI) in its various architectural and virtualization events. They have illustrated a path to building a predictive model for workload performance.

ElMoustapha et al. [3] have built a model–tree regression based approach M5 algorithm that account for events from a subset of SPEC suite. They note that the regression tree model gives the best interpretation of the functioning of memory, which corresponds to the intuitive idea of the role of second level cache (L2) in the use of data. Additionally, it can be stated that one disadvantage of decision tree is that it is difficult to design as an optimal tree and its effectiveness is dependent on the type of its design. Rai et al. [4] have suggested regression models by learning the cache L2. The

authors show that the processor Intel Core Duo model obtained from a single processor accurately predicts L2 on a different processor.

Joseph [5] built a regression by factor analysis for the IBM PowerPC processor and showed that there are three most significant factors: the pipeline depth of the command, the reordering buffer and queue micro–processor. In addition, the size of the L2 cache latency and exchanges with it are important.

Other studies have been conducted on the workload of special programs. Xu et al. [6] analyzed the cache behavior of a group of important multimedia applications and compared results with SPEC suite applications. They found that multimedia applications actually have better cache behavior than SPEC suite, although these programs have large data and a little data reuse. They have mentioned three reasons for this conclusion: block–partitioning algorithms with small block size to the input data in the cache, within these blocks there is significant data reuse as well as spatial locality, a large number of references which are relatively small and can easily fit in reasonably–sized caches. In addition, further studies can be found in [7], [8] and other related resources.

III. BACKGROUND

This section provides a perspective for understanding the experiments and results analysis of this paper and consists of two parts: the first part explains the regression analysis of the data and the second part shows the models of the memory hierarchy under study.

A. Regression Analysis

Regression analysis is a statistical tool that can show the relationship between a dependent variable to its independent variables in a sample space. In a regression model, can be seen a regression relationship between a response variable and multi predictor variables by Equation 1:

$$y = b_0 + \sum_{i=1}^{n} b_i X_i + a \quad (1)$$

Or it can be written in the form of Equation 2:

$$y = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n + a \quad (2)$$

That y is the response or dependent variable, $b_1, \dots, b_n$ is called effects or regression coefficients and expresses the expected change on variable y in unit change in X_i.The variables $X_1, X_2, \dots, X_n$ are called predictor or independent variables. The variable b_0 is an intercept between response and transverse axis of the coordinate (Y–Axis) although theoretical considerations proposed its value should be zero [9].

The variable a is called residual or error term due to lack of fit in equation. This amount includes all other possible factors that affect the dependent variable y and are not among the independent variables X_i.

The regression model is used to estimate and predict the dependent variables in arbitrary points of a statistical space. Accurately estimate is identifiable with the null hypothesis, so assume there is no significant relationship between the measured parameters considering Equation 3:

$$H_0: b_0 = b_1 = \cdots = b_n = 0 \quad (3)$$

The concept of a null hypothesis is used in two approaches: the significance testing approach of Ronald Fisher and the hypothesis testing approach of Neyman and Pearson. Both methods are considered the certain error rates. If researcher can bring a strong indication from a statistical standpoint to reject the null hypothesis, then the null hypothesis is rejected. The minimum confidence interval for rejecting the null hypothesis is equal to 95%.

Because, usually variables measured in different units so b coefficients in Equation 2 are different in unit. If these units are standard factor units are standard similar to it (beta weights) and Equation 2 can be written as Equation 4:

$$y = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \quad (4)$$

Including variable selection methods in the run a regression is the stepwise regression method. For this purpose in three different ways variables one by one are entered into the model or they will be removed. These methods included Forward selection, backward elimination, and Bidirectional elimination. This method deals with two important parameters: p–inclusion and p–removal. Typically, values are selected to p–inclusion=0.5 and p–removal=0.10. The inclusion factor test the hypothesis about the importance of zero partial correlation factors with the dependent variable. If hypothesis is rejected, then there is a significant factor, if the value of the F–statistic Fischer partial correlation is less than a given probability p–inclusion. Otherwise factor is not included in the set of meaningful and the algorithm proceeds to check the next factor. In this step, pattern continues until the minimum value of F at each step is more than the value defined.

B. Hierarchical Memory Model

In this paper experiments were performed on 3 models of Intel mobile processors: Intel Core i5–460M, Intel Core i5–3317U, and Intel Core 2 T5800.

Intel Core i5–460M micro–architecture Nehalem [10] has two cores and three levels of caches, where L1 and L2 are exclusive and L3 is an inclusive cache with respect to L1 and L2. The L1 cache is divided for instruction and data parts, they are allocated to each core separately, L2 cache is also allocated to each core, instructions and data are stored in L2 cache together. L3 cache is shared between the cores. TLB design is performed in hardware mode on this processor. It has two levels, the first level of the buffer allocated for each core, and then it is divided for instruction and data. Instruction TLB is divided into two modes: 4Kbyte pages size and 2 (or 4) Mbyte pages size. 4Kbyte mode has 4–way set associative structure and 64 entries line in cache. 4Mbyte mode has fully associative structure with 7 entries line.

Data TLB is divided into two modes: 4Kbyte pages size and 2 (or 4) Mbyte pages size. 4Kbyte mode has 4–way set associative structure and 64 entries line in cache. 4Mbyte mode has 4–way set associative structure with 32 entries line. The second level TLB (STLB) allocated for each core separately. If during the execution multi–threading is enabling, the STLB will be shared between two thread of each core. In this case, each core of 460M is capable of executing two

threads simultaneously of a total 4 threads. Figure 1 shows a diagram of an Intel i5–460M.

In general, Intel Core i5–3317U micro–architecture Ivy Bridge [11] has a structure similar to 460M, but at least has two differences: 3317U lacks the large page mode on ITLB, instead there has two part on STLB: the 4Kbyte page mode shared between cores with 512 entries and 4–way set associative structure and 1Mbyte page mode separately for each core with 64byte line size and 4–way set associative structure. Figure 2 shows a diagram of an Intel i5–3317U.

Intel Core 2 T5800 micro–architecture Core [13] has not L3 cache at all. It possesses L1I for instruction and L1D for data similar to 460M and also a L2 cache with 2 Mbyte shared between the cores. TLB is partitioned into two levels: first one has included ITLB for instruction and DTLB for data. Second level of TLB has DTLB separately for each core. Figure 3 shows a diagram of an Intel Core 2 T5800.

The i5–460M and T5800 have Hard Disk Device for persistent storage of data, but 3317U has Solid State Device (SSD). 3317U has 32Gbyte physical memory. The 460M has 8Gbyte and T5800 has 3Gbyte of it.

Memory hierarchies are respected in all the three systems. If the address is not found in the first level of TLB, then second level of TLB will be to search and if it is not found again, then memory controller addresses to Random Access Memory (RAM). If the address is not found then there is a miss situation with a complex and time–consuming search for the virtual page that contains the desired address.

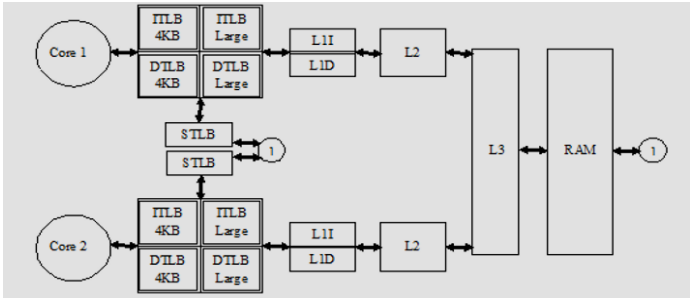

Fig. 1. A diagram of the memory hierarchy of i5–460M

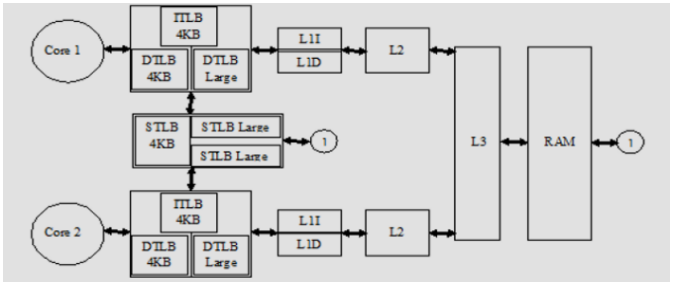

Fig. 2. A diagram of the memory hierarchy of i5–3317U

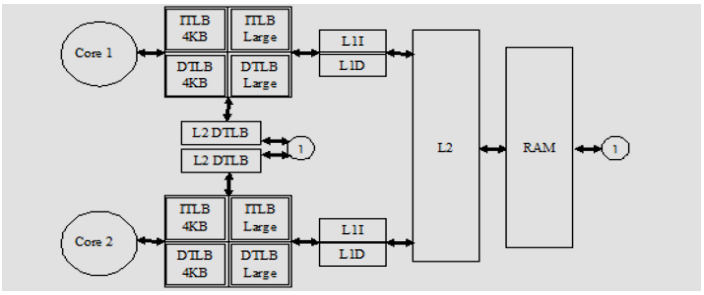

Fig. 3. A diagram of the memory hierarchy of Core 2 T5800

IV. Experimental Methodology

Experiments performed are based on 64–bit Intel environment which use features of Performance Monitoring Unit (PMU) [12] to measure various events using Intel Vtune 2013. PMU is a hardware part that builds inside a processor in all modern systems to count the performance parameters like instruction cycles, cache hits, cache misses and etc.

Intel Vtune amplifier is an application that works on 32 and 64 bit x86 based systems to count the events related to system performance [13]. Windows 7 is the operating system (OS) used on 460M and 3317U. Also, On the T5800 is installed OS XP Pack 3. All of these operating systems can run parallel programs on all CPU cores. Programs that have been used to run and to count their events by Intel Vtune are SPEC CPU2000 benchmarks.

The Standard Performance Evaluation Corporation (SPEC) is a corporation to make a standardized set of relevant benchmarks to use to evaluate system performance [SPEC]. In this paper, all 12 benchmarks of package CINT2000 (fix point operations) are used: 164.gzip, 175.vpr, 176.gcc, 181.mcf, 186.crafty, 197.parser, 252.eon, 253.perlbmk, 254.gap, 255.vortex, 256.bzip2, and 300.twolf. Measurements have been performed based on 2 user mode, so that every time on each core runs a copy of the benchmark in parallel with other core. Also, multi–threading was disabled in experiments and each copy of benchmarks is performed on each core separately. During the experiment, pre–fetching was off and it has no any effect on the results of TLBs and caches. For the reliability of the results of the experiments, each 12 application is performed at least 50 times.

Each execution consists of three random repetitions of the application to calculated average parameters and in total 1800 times is performed for each system. Therefore, a total of 5400 runs have been performed to calculate the available values for each system consists Cycle Per Instruction (CPI), miss rate for Instruction TLB (ITLB), Data TLB (DTLB), Second TLB (STLB), Instruction of first level cache (L1I), L1D, L2, and L3 miss rate.

To calculate the CPI and Miss rates have been used main Equations (5) – (12) based on Table I.

On different processors PMU may use other equivalent events instead of listed in Table I and some equations (5)–(12) are in the other forms. For further explanation refer to the Intel documentation and other related references [14].

V. Statistical Analysis

Tables II, III and IV summarize the results of experiments as coefficients of regression model that discussed in part III, on 460M, 3317U and T5800 respectively. The information in the tables is the result of statistical analysis on MATLAB. In each row of tables, the numbers are related respectively to the following benchmarks: *1.Gzip, 2.Vpr, 3.Gcc, 4.Mcf, 5.Crafty, 6.Parser, 7.Eon, 8.Perlbmk, 9.Gap, 10.Vortex, 11.Bzip2, and 12.*Twolf. In addition, each number in each column indicates importance of that member of memory hierarchy of its benchmark base on P (P-value). For example, the second row of Table II, have the information of *1.Gzip* that L3 has higher importance (1), next STLB (2), DTLB (3), L2 (4) and ITLB (5).

TABLE I
Events Used for 460M, 3317U and T5800.

Event	Name	Explanation
CPU_CLK_UNHALTED .THREAD	*CCT*	Total cycles of execution for the program under test.
INST_RETIRED.ANY	*IRA*	Number of instructions that retired execution.
ITLB_MISS_RETIRED	*IMR*	Number of retired instructions that miss on ITLB.
DTLB_MISSES.ANY	*DMA*	Number of data requests that miss on DTLB.
DTLB_LOAD_MISSES .STLB_HIT	*DLS*	Number of miss on DTLB that Hit on STLB.
L1I_MISSES	*LIM*	Number of the miss on Instruction L1.
L1D_REPL	*LDR*	Number of miss on Data L1 when L1 Data cache line allocated.
L2_LINES_IN.SELF . ANY	*LLS*	Count the L2 line allocated to miss on L2.
MEM_LOAD_RETIRED .L3_MISS	*MLL*	Count the retired loads that miss the L3 cache.

$$CPI = CCT/IRA \quad (5)$$
$$ITLB\ miss\ rate = IMR/IRA \quad (6)$$
$$DTLB\ miss\ rate = DMA/IRA \quad (7)$$
$$STLB\ miss\ rate = 1 - (DLS/IRA) \quad (8)$$
$$L1I\ miss\ rate = LIM/IRA \quad (9)$$
$$L1D\ miss\ rate = LDR/IRA \quad (10)$$
$$L2\ miss\ rate = LLS/IRA \quad (11)$$
$$L3\ miss\ rate = MLL/IRA \quad (12)$$

L1D has not significant value and it shown with symbol *X*. In statistical significance testing, the P–value is a significant one that discussed in part III. A low P–value (< 0.05) indicates that you can reject the null hypothesis.

If P–value is less than 0.05, the null hypothesis is rejected with 95 percent probability. Test is called significant if it is less than 0.05 and if it is less than 0.01 is called very significant.

According to the P–value in Tables II, III and IV system state from the standpoint of miss rates are summarized in Tables V and VI that respectively show the same factors and closely factors. Theses tables have three rows. In each row, the two systems are compared with regard to the factors in Tables II, III and IV. This comparison allows better details can be seen. For example, benchmarks Gcc and Vortex have same value for ITLB (2 for Gcc on 460M and also 2 for 3317U).

According to Tables V and VI and considering the nature of each of the benchmarks, the following results can be seen:

Benchmarks Gzip and Bzip2 are in Last Level of Cache (LLC). Gzip and Bzip2 both is data compression program.
Benchmarks Vpr and Twolf are in LLC and also Twolf can be seen in L1D. Vpr and Twolf is design program: Vpr is Integrated Circuit Computer–Aided Design program and Twolf is Computer Aided Design.
Benchmark Vortex can be seen in three locations of Tables V and VI: ITLB, DTLB and LLC. Vortex is a single–user object–oriented database transaction benchmark that is run three times: each time a different mix of database inserts, deletes and lookups is used to simulate different database usage patterns.

TABLE II
Results of Factor Analysis for 460M.

	ITLB	DTLB	STLB	L1I	L1D	L2	L3
1	5	3	2	6	X	4	1
P	0.00753	2.02e-11	7.18e-12	0.04312	P > 0.05	0.00161	3.58e-16
2	X	3	X	X	4	2	1
P	P > 0.05	6.34e-05	P > 0.05	P > 0.05	0.000506	3.08e-12	4.64e-29
3	2	X	4	X	X	1	3
P	1.17e-06	P > 0.05	3.76e-05	P > 0.05	P > 0.05	1.60e-11	1.22e-06
4	5	2	3	X	4	X	1
P	0.01577	4.45e-05	0.00121	P > 0.05	0.012805	0.99245	8.68e-16
5	X	X	X	X	X	1	2
P	P > 0.05	P > 0.05	P > 0.05	P > 0.05	P > 0.05	1.67e-17	4.97e-11
6	X	X	4	X	3	2	1
P	P > 0.05	P > 0.05	0.00015	P > 0.05	2.28e-06	4.23e-13	6.30e-24
7	X	X	1	3	4	X	2
P	P > 0.05	P > 0.05	1.32e-07	0.00281	0.003161	P > 0.05	8.45e-05
8	1	X	X	4	X	2	3
P	3.62e-06	P > 0.05	P > 0.05	0.0316	P > 0.05	7.78e-05	0.02077
9	4	5	X	2	3	X	1
P	0.00947	0.02363	P > 0.05	8.07e-08	0.000429	P > 0.05	1.67e-15
A	3	4	X	X	X	2	1
P	6.18e-05	0.00170	P > 0.05	P > 0.05	P > 0.05	4.78e-08	3.19e-28
B	X	X	2	X	X	X	1
P	P > 0.05	P > 0.05	2.51e-06	P > 0.05	P > 0.05	P > 0.05	1.04e-24
C	X	X	X	4	3	2	1
P	P > 0.05	P > 0.05	P > 0.05	0.01695	0.001709	1.48e-10	1.09e-44

Benchmark Mcf can be seen in DTLB. Mcf is a program used for single–depot vehicle scheduling in public mass transportation.

Benchmark Parser can be seen in DTLB and STLB. Parser is a word processing benchmark.

Benchmark Eon can be seen in ITLB and STLB. Eon is a computer visualization that sends a number of 3D lines into a 3D polygonal model.

Benchmark Gcc can be seen in ITLB. Gcc is a C language optimizing compiler.

TABLE III
RESULTS OF FACTOR ANALYSIS FOR 3317U.

	ITLB	DTLB	STLB	L1I	L1D	L2	L3
1	X	X	X	X	3	2	1
P	P > 0.05	P > 0.05	P > 0.05	P > 0.05	3.60e-05	1.67e-14	1.17e-24
2	X	1	X	X	X	2	3
P	P > 0.05	2.70e-17	P > 0.05	P > 0.05	P > 0.05	8.78e-08	0.00832
3	2	X	X	X	1	X	X
P	0.04098	P > 0.05	P > 0.05	P > 0.05	4.36e-07	P > 0.05	P > 0.05
4	X	1	X	X	X	X	X
P	P > 0.05	0.00117	P > 0.05	P > 0.05	P > 0.05	P > 0.05	P > 0.05
5	X	2	X	X	1	X	X
P	P > 0.05	0.02246	P > 0.05	P > 0.05	0.00015	P > 0.05	P > 0.05
6	X	2	3	X	1	X	X
P	P > 0.05	0.01288	0.01742	P > 0.05	3.02e-08	P > 0.05	P > 0.05
7	1	X	2	X	X	3	X
P	2.51e-10	P > 0.05	9.41e-06	P > 0.05	P > 0.05	0.00014	P > 0.05
8	X	2	1	X	3	X	X
P	P > 0.05	0.00665	0.00047	P > 0.05	0.02273	P > 0.05	P > 0.05
9	X	X	2	X	X	1	X
P	P > 0.05	P > 0.05	176.53	P > 0.05	P > 0.05	6.4413	P > 0.05
A	3	1	4	X	X	2	X
P	0.00109	4.62e-08	0.03355	P > 0.05	P > 0.05	2.28e-05	P > 0.05
B	3	X	X	X	2	X	1
P	0.03373	P > 0.05	P > 0.05	P > 0.05	0.028793	P > 0.05	4.48e-19
C	4	1	3	X	X	X	2
P	0.03362	2.36e-16	1.66e-05	P > 0.05	P > 0.05	P > 0.05	3.34e-10

VI. CONCLUSION

In this paper, dependence of memory hierarchy footprints between three different models of Intel multi–core architecture was analyzed and compared. These architectures included Intel Core i5–460M, Intel Core i5–3317U, Intel Core 2 T5800. These systems were used to evaluate by SPEC CINT2000 benchmarks and for counting events to calculate miss rates was used Intel VTune2013. Statistical analysis using stepwise regression techniques was done using MATLAB software. In this analysis, the pre–fetching and multi–threading have been inactive.

TABLE IV
RESULTS OF FACTOR ANALYSIS FOR T5800.

	ITLB	DTLB	STLB	L1I	L1D	L2	L3
1	X	X	X	X	1	2	X
P	P > 0.05	P > 0.05	P > 0.05	P > 0.05	0.036379	0.00289	P > 0.05
2	X	X	X	X	1	X	X
P	P > 0.05	0.34071	P > 0.05	0.97934	0.005733	0.5548	P > 0.05
3	X	X	X	X	X	1	X
P	P > 0.05	0.29284	P > 0.05	0.16596	0.68724	0.00070	P > 0.05
4	X	1	X	X	X	X	X
P	P > 0.05	0.34027	P > 0.05	0.75403	0.60824	0.04820	P > 0.05
5	X	X	X	X	X	1	X
P	P > 0.05	0.18593	P > 0.05	0.1363	0.20942	0.00351	P > 0.05
6	X	2	X	X	X	1	X
P	P > 0.05	0.00054	P > 0.05	P > 0.05	0.41512	2.78e-06	P > 0.05
7	2	1	X	X	X	X	X
P	1.25e-05	0.00010	P > 0.05	0.54903	0.90178	0.38629	P > 0.05
8	X	X	X	X	X	1	X
P	P > 0.05	0.59604	P > 0.05	0.5474	0.4318	0.00103	P > 0.05
9	X	1	X	X	X	X	X
P	P > 0.05	9.64e-05	P > 0.05	P > 0.05	0.62236	0.52606	P > 0.05
A	X	2	X	X	X	1	X
P	P > 0.05	0.00023	P > 0.05	0.53304	0.125	0.00145	P > 0.05
B	X	X	X	X	X	1	X
P	P > 0.05	0.80631	P > 0.05	P > 0.05	0.97087	0.00093	P > 0.05
C	X	X	X	X	2	1	X
P	P > 0.05	0.33305	P > 0.05	0.30074	0.018262	0.00794	P > 0.05

TABLE V
BENCHMARKS WITH EQUAL IMPORTANCE.

Similar	ITLB	DTLB	LLC (L2 or L3)
460M, 3317U	Gcc, Vortex		Vpr, Gzip, Bzip2
460M, T5800			Vortex, Bzip2
3317U, T5800		Mcf, Parser	Bzip2

TABLE VI
BENCHMARKS WITH NEARLY IDENTICAL IMPORTANCE.

Similar	ITLB	DTLB	L1D	LLC (L2 or L3)
460M, 3317U		Mcf		Twolf
460M, T5800			Twolf	Gzip
3317U, T5800	Eon	Vortex		Gzip

The results indicate that miss rate on different levels of memory hierarchy is dependent on classes of applications. In this direction, compression programs including Gzip and Bzip2 are located in LLC. Design programs including Vpr and Twolf are located in LLC and Twolf is also in L1D.

Database program included Vortex is run three times and it can be seen in ITLB, DTLB and LLC. Scheduling program included Mcf is located in DTLB. Parser is a word processing and it is located in DTLB and STLB. Eon as computer visualization class is in ITLB and STLB. Finally, Gcc as a C compiler is in ITLB level. The authors, based on these results have suggested that operating system designers attention to this classification and in the execution time of each class of programs through disable insignificant levels, reduce the search time and the miss penalty on the less important levels of memory hierarchy and therefore the system will increase performance.

As future work, it can be seen the impact of pre–fetching and multi–threading techniques on these experiments and analysis.

REFERENCES

[1] S. Akhter, and J. Roberts, *Multi–Core Programming.* USA: Inter Press, 2006.

[2] H. Mousa, K. Doshi, T. Sherwood, and E. Vall, "VrtProf: Vertical Profiling for System Virtualization," *Hawaii International Conference on System Science*, pp. 1–10, 2010.

[3] E. Vall, J. Woodlee, C. Yount, K. Doshi, and S. Abraham, "On the Comparison of Regression Algorithms for Computer Architecture Performance Analysis of Software Applications," *International Symposium on Performance Analysis of Systems and Software*, pp. 179-190, 2008.

[4] J. K. Rai, A. Negi, R. Wankar, and K. D. Nayak, "Characterizing L2 cache behavior of programs on multi–core processors: Regression models and their transferability," *International Journal of Computer Information Systems and Industrial Management Applications*, pp. 212–221, 2010.

[5] P. J. Joseph, K. Vaswani, and M. J. Thazhuthaveetil, "Construction and Use of Linear Regression Models for Processor Performance Analysis," *Twelfth International Symposium on High–Performance Computer Architecture*, pp. 99–108, 2006.

[6] Z. Xu, S. Sohoni, R. Min, and Y. Hu, "An Analysis of Cache Performance of Multimedia Applications," in *IEEE Transactions on Computers*, pp. 20–38, 2004.

[7] R. R. Rao, D. Blaauw, D. Sylvester, and A. Devgan, "Modeling and analysis of parametric yield under power and performance constraints," in *IEEE Design and Test of Computer*, pp. 376–385, 2005.

[8] G. Palermo, C. Sivano, and V. Zaccaria, "A Variability–Aware Robust Design Space Exploration Methodology for On–Chip Multiprocessors Subject to Application–Specific Constraints," in *ACM Transactions on Embedded Computing Systems*, pp. 1–29, 2012.

[9] A.K. Sharma, *Correlations and Regression*, India: New Delhi, Discovery Publishing House, 2005.

[10] J. Casazza, *First the Tick, Now the Tock: Intel Microarchitecture (Nehalem).* USA, Intel Corporation, 2009.

[11] T. Shanley, *x86 Instruction Set Architecture*, USA, MindShare Press, 2009.

[12] Y. Choi., "Design and Experience: Using the Intel® Itanium® 2 Processor Performance Monitoring Unit to Implement Feedback Optimizations," in *Proceedings of the 34th Annual International Symposium on Microarchitecture*., USA: Texas, pp. 182–191, 2001.

[13] J. Reinders, *Intel VTune Performance Analyzer Essentials*, USA, Intel Press, 2005.

[14] (2013). *Intel 64 and IA–32 Architectures Optimization Reference Manual.*[Online].Available: http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf