Multilevel Cache Modeling for Chip-Multiprocessor Systems

Pablo Prieto, Valentin Puente, José-Ángel Gregorio
University of Cantabria, Spain
{prietop,vpuente,monaster}@unican.es

Abstract—This paper presents a simple analytical model for predicting on-chip cache hierarchy effectiveness in chip multiprocessors (CMP) for a state-of-the-art architecture. Given the complexity of this type of systems, we use rough approximations, such as the empirical observation that the re-reference timing pattern follows a power law and the assumption of a simplistic delay model for the cache, in order to provide a useful model for the memory hierarchy responsiveness. This model enables the analytical determination of average access time, which makes design space pruning useful before sweeping the vast design space of this class of systems. The model is also useful for predicting cache hierarchy behavior in future systems. The fidelity of the model has been validated using a state-of-the-art, full-system simulation environment, on a system with up to sixteen out-of-order processors with cache-coherent caches and using a broad spectrum of applications, including complex multithread workloads. This simple model can predict a near-to-optimal, on-chip cache distribution while also estimating how future systems running future applications might behave.

1 Introduction

A simple model that can predict CMP cache hierarchy performance with a reasonable degree of accuracy will be presented. Using this method, it is possible to obtain a first degree approximation for the best proportion of silicon that should be devoted to each level of the memory hierarchy. Needless to say, the model is not precise enough to replace time-consuming, cycle-accurate simulation, but it does permit a preliminary analysis in order to confine the vast design space into a localized region. While an experienced architect guided by intuition could achieve similar conclusions, the model formalizes the process considering real application behavior and technological properties of the implementation.

The observation derived from Chow's classic works [2][3], states that the pattern re-reference interval in any level in the memory hierarchy approximately follows a power law with an exponent equal to -0.5. This is known as the sqrt 2 rule. This property has been used previously with the same objective by Przybyski et. al. [10][11] to determine the most suitable memory hierarchy in single-processor systems. Our objective is to extend those results to much more complex state-of-the-art multi-core architectures. Those architectures introduce a substantial complexity: processors must synchronize their work and last level cache should scale access bandwidth according to processor count. Therefore, the appearance of coherence activity and complex hardware structures in the cache, such as NUCA, could hinder basic observation or invalidate assumptions.

Nevertheless, our results indicate that application re-reference patterns are still dominant over other convoluted side effects introduced by multithread applications running on multi-core architectures. Therefore, using a few computations, we will be able to estimate average access time, with a reasonable margin of error, thus justifying the feasibility of two or three cache levels and the capacity devoted to each level.

Section 2 presents a basic model for the application and cache. Section 3 extends the basic model for a CMP system and explains workload characterization. In Section 4 we use the model for a specific CMP architecture and demonstrate the model validity when the cache size and number of processors are increased. Finally we summarize some conclusions and future work in Section 5.

2 Base Models

2.1 Cache miss rate model

It is well known [2][7][12] that in single-core systems, cache misses can be empirically approximated by a power law function of cache capacity, C:

$$M=m_0 \cdot C^{\alpha} \qquad (1)$$

Where m_0 and α are application-dependent parameters. According to Hartstein et al. [4], m_0 can be expressed as a function of two constants R_0 and K_0:

$$m_0=R_0 \cdot K_0^{-\alpha} \qquad (2)$$

These parameters only depend on the re-reference pattern [4], and can be obtained empirically. This observation is valid if the application is in the cache utility range. If the working size is smaller than cache capacity, i.e., saturating utility, or too big to exploit any local temporality, i.e., low utility, the observation does not apply. Additionally, the power law is only valid if capacity misses dominate. m_0 is cache size independent and is applicable with multiple levels of cache if the size ratio between levels is high [4]. Figure 3 shows the behavior of an application in the last two levels of cache of a system. As can be seen, there is a high correlation with a power law. If conflict misses are very frequent, due for example to very low associativity, (1) is not valid because conflict misses are not correlated with the temporal re-reference pattern.

2.2 Average Access Time Model

In general, the global average latency of a cache hierarchy can be expressed by the function:

Manuscript submitted: 09-May-2011. Manuscript accepted: 11-Jul-2011. Final manuscript received: 18-Jul-2011

Posted to the IEEE & CSDL on 7/22/2011
DOI 10.1109/L-CA.2011.20

$$L_{globl}=H_1 \cdot L_1+M_1 \cdot H_2 \cdot L_2+M_1 \cdot M_2 \cdot H_3 \cdot L_3 \ldots \quad (3)$$

Where, H_i and M_i are the hit and miss rates of cache level 'i', and L_i is the average access latency of that level. The last term of this expression corresponds to the main memory contribution.

It should be noted that inter-level dependencies such as inclusion, coherence, etc. are not considered in the simplified equation (1), and large variations in previous cache levels affect the re-reference pattern of higher levels [3]. Therefore, we will consider two different parameters for each level of the hierarchy. Substituting (1) into (3), and expressing the hit rate as a function of the miss rate, we obtain:

$$L_{globl}=(1-M_1) \cdot L_1+M_1 \cdot (1-m_{02} C_2^{\alpha 2}) \cdot L_2+M_1 \cdot m_{02} \cdot C_2^{\alpha 2} \cdot (1-m_{03} \cdot C_3^{\alpha 3}) \cdot L_3+\cdots+M_1 \cdot M_2 \cdot M_3 \ldots M_n \cdot L_{mem} \quad (4)$$

Additionally, the access time to a cache bank is dependent on its size, so latency can be expressed as a function of capacity. In a naïve cache bank design, access latency is proportional to the surface occupied by the cache, i.e., linearly proportional to the square root of the capacity. In a sophisticated design, it makes sense to assume that, for a given associativity, access time can be represented as a general power law function of capacity.

$$L_{Physical}=d_0 \cdot C_i^{\gamma} \quad (5)$$

Equation (5) defines access latency as a function of C_i (bank capacity), d_0 (latency factor) and γ (exponent). Using Cacti 6.0[10] with a wide range of realistic configurations, it turns out that the assumption is accurate. For example, Figure 1 shows an eight-way cache bank with 64 bytes per line, serial data and tag access with a 32 *nm* technological node for different capacity sizes. We will assume 3GHz clock frequency.

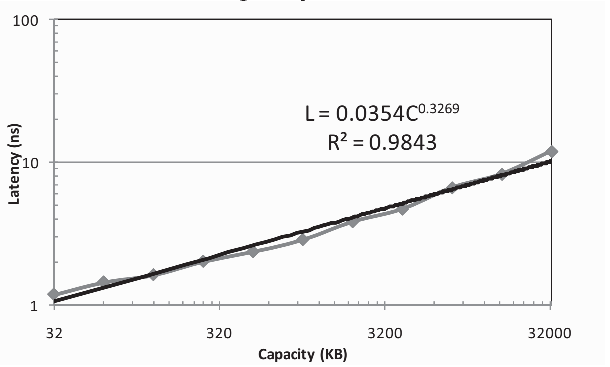

Figure 1. Cache bank latency as a power law of capacity, using 32nm technology.

The hit latency on each level of the hierarchy is not just bank access latency; it also includes a penalty due to misses in previous levels. This miss penalty depends on previous levels' miss latency and therefore we assume:

$$L_i= L_{Physical} +\sigma \cdot L_{i-1} = d_0 \cdot C_i^{\gamma} +\sigma \cdot L_{i-1} \quad (6)$$

Thus, substituting the access latency of each level in (5) into the global latency expression in (4), and considering just three levels of cache in our memory hierarchy, we obtain the following expression:

$$L_{globl} = (1-M_1)L_1 + M_1(1-m_{02}{C_2}^{\propto_2})(d_{02}{C_2}^{\gamma_2}+\sigma L_1) + M_1 m_{02}{C_2}^{\propto_2}(1-m_{03}{C_3}^{\propto_3})(d_{03}{C_3}^{\gamma_3}+\sigma L_2) + M_1 m_{02}{C_2}^{\propto_2} m_{03}{C_3}^{\propto_3}(L_{mem}+\sigma L_3) \quad (7)$$

Where, L_{mem} is the access latency to the main memory, and M_1 and L_1 are miss rate and latency of the first level cache, respectively. As the L1 cache size is mainly determined by the processor clock frequency, we will assume the same value in all cases under study. We also er L_{mem} as constant.

3 CMP Model

3.1 Memory Hierarchy

We will use a state-of-the-art CMP architecture, which mimics the aggressiveness of commercial systems such as Intel Nehalem or IBM Power7. In order to do so, the chip will include P processors, first level (L1) and second level private caches (L2) and a shared NUCA [6] third level (L3) distributed across B banks connected using a point-to-point network. Cache coherence is maintained using a distributed directory in L3 and forcing lower levels inclusion.

We simplistically ignore contention or coherency misses in the mesh network to estimate L3 access latency. Therefore, using the topological average distance, router pipeline length and packet sizes for data responses and request, we can easily compute average network latency (N_0). Thus, the latency of L3 can be represented by:

$$L_{Physical}=d_0 \, (C_i/B)^{\gamma}+N_0 \quad (8)$$

As C_2 refers to the second level cache as a whole, we can express it as the sum of all private L2 caches of each processor ($C_{2p}P$) (where P is the number of processors). As we wish to determine the amount of cache devoted to each level, we assume a fixed area C devoted to the last two levels of cache. Thus, the capacity of each of these two levels can be expressed as a fraction of C:

$$C_{2p}=x \cdot C$$
$$C_3=C-C_{2p} \cdot P=(1-x \cdot P) \cdot C \quad (9)$$

We denote capacity ratio as x. Substituting expression (9) into (7):

$$L_{globl} = (1-M_1)L_1 + M_1(1- m_{02}(xCP)^{\propto_2})(d_{02}(xCP)^{\gamma_2}+\sigma L_1) + M_1 m_{02}(xCP)^{\propto_2} \left(1- m_{03}((1-xP)C)^{\propto_3}\right)\left(d_{03}\left(\left(\frac{1-xP}{B}\right)C\right)^{\gamma_3} + N_0 + \sigma L_2\right) + M_1 m_{02}(xCP)^{\propto_2} m_{03}\left((1-xP)C\right)^{\propto_3}(L_{mem}+\sigma L_3)$$

(10)

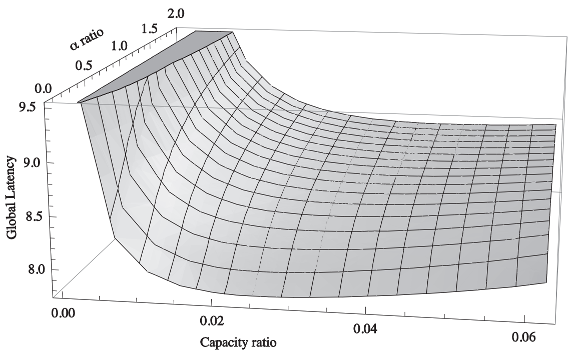

Figure 2. Global Latency behavior versus C_{2p}/C and α_2/α_3.

With this expression, we can determine the average access time as a function of the area devoted to each one of the L2 and L3 levels, i.e., the term x. If we choose a fixed total capacity of 16MB (C=16MB), eight processors (P=8), and sixteen NUCA banks, Figure 2 represents total memory latency as a function of both the capacity ratio and the ratio of decay constant in the re-reference pattern of L2 and L3. Where the y axis represents the ratio between the re-reference pattern at L2, $\propto_2$, and at L3, $\propto_3$. The shape of the function shows that there is a minimum in the latency. In other words, (10) is useful to determine the optimum capacity ratio in which the average latency perceived by the processor is minimal.

3.2 Workload Characterization

Both factor m_0 and exponent α in (1) are workload dependent and must be obtained for each level of the cache hierarchy.

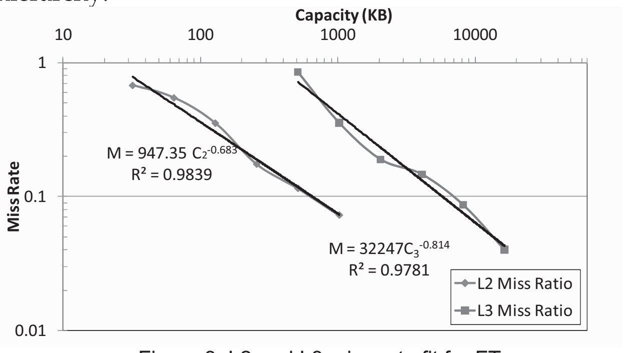

Figure 3. L2 and L3 miss rate fit for FT.

We obtained these parameters by experimentation employing a full system simulator based on Simics [8] extended only with the memory simulator, *ruby*, part of the GEMS timing structure [9]. We characterize L2 cache varying its size while maintaining a fixed large L3 cache size (64MB). Subsequently, for L3 re-reference pattern characterization, L2 is kept fixed at 64KB. Accurate memory timing has been considered: coherence activity and network contention have been taken into account. We have chosen three different types of workloads with different sharing degrees. All applications will be in the cache-size dependent zone. On one hand, for multiprogrammed benchmarks, which consists of eight copies of a single program running on the system, there is not shared data. On the other hand, there are multithreaded workloads, chosen among numerical applications of the NAS parallel benchmark (OpenMP implementation version 3.2.1) [5], and transactional workloads from the *Wisconsin Commercial Workload suite* [1].

TABLE 1. MAIN PARAMETERS OF WORKLOADS UNDER STUDY.

Workload	m_{02}	α_2	R^2	m_{03}	α_3	R^2	α_2/α_3
BZIP2	38.014	**-0.394**	0.95	22605	**-0.745**	0.98	0.533
GCC	553.79	**-0.715**	0.99	2.3E5	**-0.994**	0.98	0.719
OLTP	43.858	**-0.395**	0.97	7022.3	**-0.744**	0.98	0.53
APACHE	33.506	**-0.356**	0.96	3550.5	**-0.678**	0.99	0.525
JBB	47.212	**-0.418**	0.99	228.64	**-0.467**	0.99	0.895
IS	3.4246	**-0.15**	0.94	4E6	**-0.967**	0.99	0.155
FT	947.35	**-0.683**	0.98	3.2E4	**-0.814**	0.98	0.839

The miss rate of each level for different cache sizes is approximated by a power function and the parameters m_0 and α are determined. Figure 3 shows the fit for the FT application. A summary of these parameters for each workload can be seen in Table 1 where the meaning of the parameters is the same as those in (1). We include the correlation of each fit, which in most cases is close to 1.0.

4 MODEL APPLICATION & VALIDATION

4.1 Optimal L2/L3 capacity ratio

In order to validate the efficacy of the model, we will perform detailed micro-architectural simulations with each application and capacity ratio. The system configuration chosen has eight 4-way, out-of-order processors with an instruction window of 128 entries.

Both L2 and L3 caches share chip area, so increasing L2 cache size will decrease the area dedicated to L3 cache. The aggregate capacity of both levels of cache is fixed at 16MB. The interconnection network is a 4x4 Torus network with one L3 cache bank connected to each switch. Figure 4 shows the average access latency for all applications with each L2/C capacity ratio. The figure includes experimental and theoretical results derived from the application of the model. Each point has been averaged over seven applications using the arithmetic mean. The model and time-consuming simulation are in agreement in terms of the optimal capacity rate, which is 1/8, i.e., 256KB for each private L2. In most cases, the model is optimistic, due to contention unawareness and latency prediction is up to one cycle less than is observed with the experimental setup. Note that although the differences in latency are not significant, the optimal system is a 12% faster than the worst-case configuration. Other complex NUCA allocations could be considered if we model N_0 in (8) accordingly. Similar experimental results are obtained with a reasonable associativity degree for L2 and L3. Very low associativity could invalidate (1) because conflict misses will dominate over capacity misses.

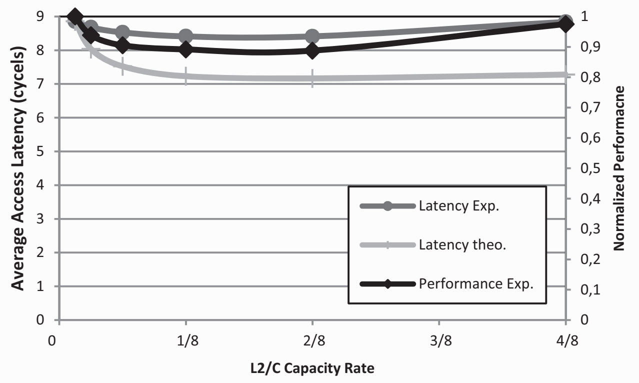

Figure 4. Theoretical and simulated applications. 8-processor system with 16MB devoted to both L2 and L3 caches.

4.2 Optimal Levels of Cache

Substituting values from Table 1 into (10) we can predict the cache behavior of the system under study for each application. Ignoring the second term in (10) we could predict a 2-level hierarchy with the last level shared cache. Moreover, ignoring the third term in (10) we could model a private last level cache. Finally, for each of the parameterized applications we determine the optimal x when 3 levels are considered. The model, according to the results shown in Figure 5, indicates that having a third level cache is the best option.

Although, a single second level is not performance compelling, the model prediction in terms of shared versus private is coherent: with applications with high sharing degrees, such as *oltp* or *apache*, shared cache performs better than private caches. With low sharing degree, such as numerical applications, private caches are a better alternative. Experimental results, using the previously depicted configuration, are also provided. In all applications except IS, the ordering is predicted correctly. When private slices size is 2MB, IS reach saturation utility, which is not captured by the model.

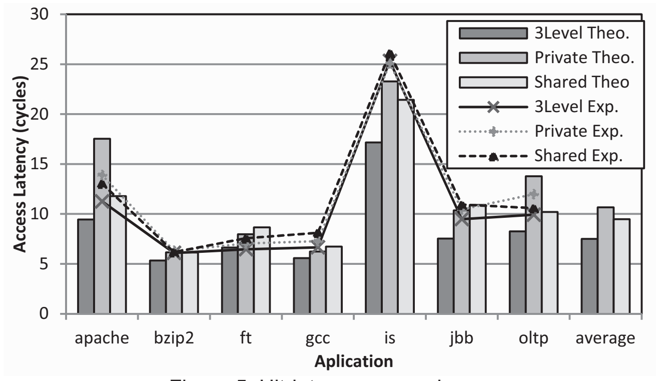

Figure 5. Hit-latency comparison.

4.3 Next Generation Systems and applications

The model can also be used for systems with larger numbers of cores, cache size or future applications. We will apply the model in a system with 16 processors and 128MB of capacity devoted to L2 and L3 caches.

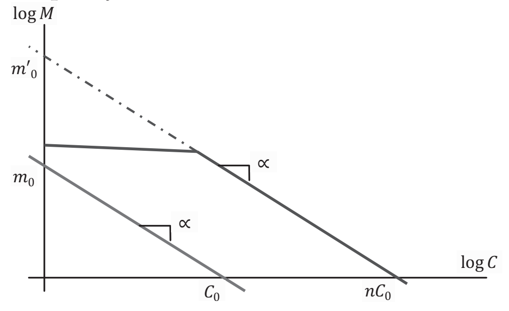

Figure 6. m_0 dependencies in linear region.

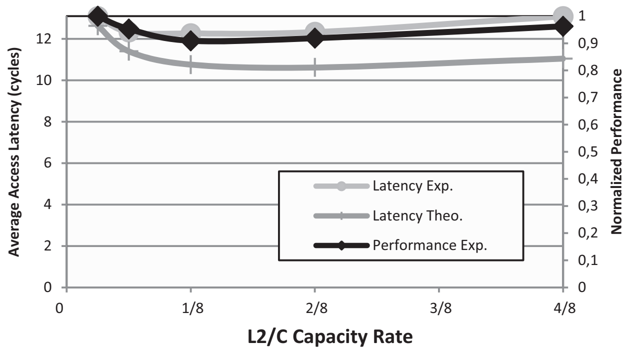

Figure 7. Theoretical and simulated applications. 16 processor system with 128MB devoted to L2 and L3 cache.

It is not possible to characterize future applications but it might be feasible to scale up if the properties are known for current applications. It is expected that applications executed in future systems will scale in requirements, so we assume the system will be working in the cache dependent region. According to Figure 6, it can be observed that if the linear region is displaced by a factor of n, m_0 is displaced by a factor of $n^{-\propto}$. According to (2), K_0 is multiplied by a factor n (in our proposal, n=128/16=8). To model the parameters $\propto_2$ and $\propto_3$, we averaged the values in Table 1 (0.45 and 0.80, respectively) and scaling m_{02} and m_{03} from 16MB to 128MB.

In order to validate the model prediction, five new applications with linear cache behavior have been simulated in a system with 128MB of cache. The results have been compared with the result using the parameters of future applications in (10). Figure 7 shows the comparison of the two average access latencies, also including the average of normalized performance. Again the model is in agreement with the experimental data, although now experimental optimal capacity ratio is 1/16 (512KB for each private L2) whereas theoretically, it is closer to 1/8 (1MB for each private L2).

5 Conclusions

We have examined the behavior of caches in a CMP with three levels of cache, the last one being shared. We have provided a very simple analytical model of the memory access latency behavior of multiprogrammed and multithreaded applications, with results that approximately coincide with simulation ones. In spite of the simplicity of the model, it is useful to reduce the computational effort required by time consuming simulations with a few computations.

6 Acknowledgements

This work has been supported by the Spanish Ministry of Science and Innovation, under contracts TIN2010-18159, and by the HiPEAC2 European Network of Excellence. The authors would like to thank the reviewers for their valuable comments.

References

[1] A.R. Alameldeen et al. “Evaluating Non-deterministic Multi-threaded Commercial Workloads”, Proc. 5th Workshop on Computer Architecture, pp. 30-38, Feb. 2002.
[2] C.K. Chow. “On Optimization of Storage Hierarchies”, IBM Journal of R&D, vol. 18, pp. 194-204, 1974.
[3] C.K. Chow. “Determination of Cache’s Capacity and its Matching Storage Hierarchy”, IEEE Transaction on Computers, vol. c-25, pp. 157-164, 1976.
[4] A. Hartstein, V. Srinivasan, T.R. Puzak, P.G. Emma. “On the Nature of Cache Miss Behavior: Is It SQRT(2)?”, Journal of Instruction-Level Parallelism 10, pp. 1-22, 2008.
[5] H. Jin, M. Frumkin, J. Yan. “The OpenMP Implementation of NAS Parallel Benchmarks and its Performance”, NAS Technical Report NAS-99-011, NASA Ames Research Center, Moffett Field, CA, 1999.
[6] C. Kim, D. Burguer and S.W. Keckler. “Nonuniform Cache Architecture for Wire-delay dominated on-chip caches”, IEEE Micro, vol. 23, no. 6, pp. 99-107, 2003.
[7] M.H. MacDougall. “Instruction-level Program and Processor Modeling”, Computer, vol. 17, pp. 14-24, 1984.
[8] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, F. Larsson, A. Moestedt, B. Werner. “Simics: A Full System Simulator Platform”. Computer, vol. 35, No. 2, pp. 50-58, Feb. 2002.
[9] M. Martin, D. Sorin, B. Beckman, et. al. “Multifacet’s General Execution-driven Multiprocessor Simulator (GEMS) Toolset”, Compt. Archit. News, vol. 33, No. 4, pp. 92-99, November 2005.
[10] N. Muralimanohar, R. Balasubramonian, N. P. Jouppi, ”Architecting Efficient Interconnects for Large Caches with CACTI 6.0. IEEE Micro vol. 28 No, pp. 69-79, 2008.
[11] S. Przybyski, M. Horowitz and J. Hennessy. “Performance Tradeoffs in Cache Design”, Proceedings of the 15th Annual International Symposium on Computer Architecture, pp. 290-298, 1988.
[12] S. Przybyski, M. Horowitz and J. Hennessy. “Characteristics of Performance-Optimal Multi-Level Cache Hierarchies”, Proceedings of the 16th Annual International Symposium on Computer Architecture, pp. 114-121, 1989.
[13] J.H. Saltzer. “A Simple Linear Model of Demand Paging Performance”, CACM, vol. 17, pp. 181-186, 1974.