Simulation of Asymmetric Access Time Cache in the Multi-Core Architecture

Julio C. S. Anjos, Felipe Severino, Cláudio R. Geyer, Philippe O. A. Navaux
UFRGS - Universidade Federal do Rio Grande do Sul
Instituto de Informática
Rio Grande do Sul, RS, Brasil
{jcsanjos,flseverino,geyer,navaux}@inf.ufrgs.br

Abstract

The creation of multi-core structures establishes a significant improvement in the performance of processors. However, this improvement is still insufficient for Tera-scales processing of applications. In this context, to achieve a high degree of parallelism and control of multiple Threads in the execution of programs, it is necessary to implement a greater number of cores within a single chip. The aim is to compare the asymmetric access time to the L2 cache with the symmetric access time and to confront some of the obtained results with the experiment obtained by Jie Tao et al. *[12]. The results suggest the use of asymmetric access time to the cache memory might allow the use of a lesser amount of the cache memory, maintaining the same effort than in another architecture using cache with symmetric access times.*

1. Introduction

The demand for greater processing capacity propitiated the use of two or more CPUs simultaneously on a single chip, which previously were used separately to process tasks. The new models of hardware increase the performance of processors by replicating the number of processor cores on a chip, using two or more units of processing within a single hardware to obtain a better performance.

The development of a multi-core technology creates new questions about other devices, such as memory, which has a significant impact in the systems performance. Taking the use of one or more intermediate levels of memory, known as cache memory to minimize the length of time to access the main memory. Sharing resources with more than one core help in sharing cost, boosting resource utilization and reducing the losses for traffic coherence [11].

When more than one cache is used, there should be a mechanism to deal with different values for the same data in the whole cache, a problem known as cache coherence. A reading inconsistency is called a read miss and a writing inconsistency is called write miss. Some protocols were developed to deal with the cache coherence, such as directory and snooping [9], which recognize the inconsistency in runtime. The problem is reduced, but it is not completely eliminated.

1.1. Interconnection between processors

The connection of processors and memories about interconnection network has a direct impact in the performance [9]. Usually the interconnection models use bus or crossbar switches, which have low cost and good performance for few cores, but these solutions have problems when the number of cores is increased. Some commons problems are [6]:

1. Parallel with high need of bandwidth to access share memories;
2. Latencies caused by the distance among the internal components in the chip;
3. Lack of routing efficiency;

A single chip, made with many cores, transmits their use by applications that use massively parallel processing and independent processes, often with real-time requirements. Our goal is to make the simulation of the use of caches with asymmetric latencies with the Simics simulator, using the NAS Benchmark to check the data performance in shared and private caches, with architecture of 8 cores.

Briefly, in section 2 there are related works and techniques to increase performance of studies done by other researchers. In section 3 it is presented the proposal of this work, while section 4 has detailed the experiment with simulations, used with modeling and methodology, the results are presented in section 5 and conclusions and statements of future work are in section 6.

2. Related Works

Several studies have been made by manufacturers [3], [7] and by researchers [1], [2], [5], [12] trying to identify an ideal model that enables the use of various cores with a meaningful speed-up increase, lower consumption of energy and low heat generation by watts.

According to Patterson and Hennessy [8], the cache optimization to increase performance, impacts in the techniques complexity that will be used according to the summary in Table 1.

	Factors			
Technique	Hit Time	Miss Penalty	Miss Rate	Hardware Complexity
Large block size	NI	-	+	0
Large cache size	-	NI	+	1
Higher associativity	-	NI	+	1
Multilevel caches	NI	+	NI	2
Read priority over writes	NI	+	NI	1
Avoiding address translation during cache indexing	+	NI	NI	1

NI – doesn't impact, - means that the technique decreases the factor and + means that the technique increases the factor, for the hardware complexity, 0 represents the lower value and 2 represents the upper value.

Table 1: Technical to Apply versus Cache Performance

The creation of multi-core architectures, significantly incremented the performance of the processors yet insufficient to process information related to petabytes, as applications of weather simulation, earthquakes forecasting, cataclysm , hurricanes, or as physical simulations such as the LHC - (Large Hadron Collider) among others. This motivates us to develop new researches.

3. Goals

The use of caches with symmetrical architecture is well known [11] our experiment simulates asymmetrical cache memory architecture with the following characteristics:

1. The contection will be simulated through the use of latencies among the L2 memories;

2. Introducing a configuration with asymmetric latencies as shown in Figure 1 to L2 private cache's cores, and compare with a symmetric cache memory configuration model shown in Figure 2 with L2 cache shared at each 2 cores;

3. It will be considered the use of 8 cores to access the bus simulated with the Simics Simulator as they were on a single chip;

4. To evaluate the simulations it will be applied NAS *Parallel Benchmark*, to be able to have the same simulation environment of Jie Tao *et al.* [12].

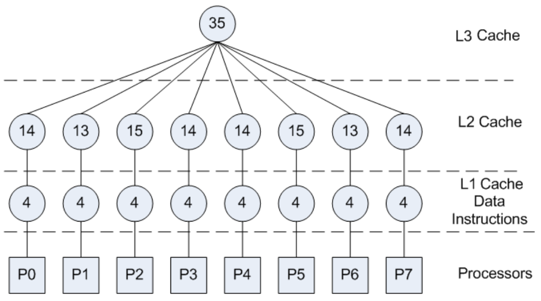

Figure 1: Latency asymmetric cache memory model

Figure 2 displays the latency model with the access times of memory hierarchy used for each level with the L2 cache shared; in this case the time (on cycles) is the same for all memories.

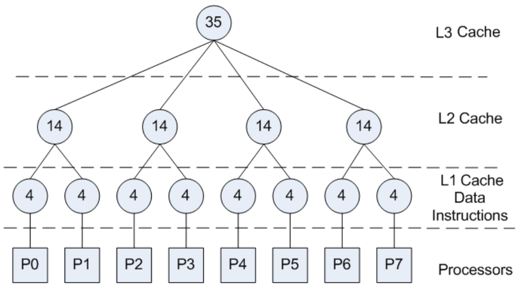

Figure 2: Latency cache memory symmetric model

On the other hand in Figure 1 the latency model for the memory hierarchy has different time for each L2 cache, therefore, it is simulated an access contention to the bus, however without cache level routing, then data is shared in the L3 cache level. The aim is to compare the asymmetric access time to the L2 cache with the symmetric access time and to confront some of the obtained results with the experiment obtained by Jie Tao *et al.* [12]. In the end, it is concluded that the obtained results suggest that the use of asymmetric access times for a cache memory might allow a lesser cache hardware without significantly compromising the amount of the miss cache.

4. Methodology

The simulations were made with the Simics version 4.0, using the Software NAS Benchmark 3.3, it was created by NASA Ames Laboratory, it is built with eight algorithms, where five are of the kernel type (IS, EP, MG, CG, FT) and three are compact applications (LU, SP, BT) [4].

For this experiment only algorithms BT and CG are used, compiled with OpenMP , and also using Magic-Instruction, which is a Simics call to obtain run time calls to make measurements.

In the simulation it is utilized the SunFireWalnut which represents the Sun Enterprise 6500 server with UltraSPARC II processor, the Ubuntu Server - 2.6 kernel operating system, using 8 cores with the latencies shown in Table 2

Cache	Size	Associative	Cycles	Update
L1	32 KB	4-way	4	write-through
L2	256 KB	8-way	10	write-back
L3	8MB	16-way	35	write-back

Table 2: Caches Configurations

The latencies on cycles are originated from the hardware configuration. All memories have 64 lines. The cache replacement policy uses LRU (Least Recent Used), which eliminates from the cache the less used blocks recently used [9].

4.1. Modeling

It is built two different L2 configurations to model this experiment: private and sharing. Figure 3 shows our test environment, using up 8 cores. Each core has one L1 cache for data and instructions and one L2 private cache, the latency of each core is given by the total distance from this to the L3 cache memory.

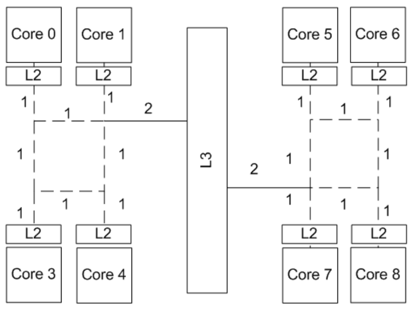

Figure 3: 8 Cores - L2 private cache

Type	Cache latency	Add latency	Total
Private L2	10 cycles	Variable	Variable
Sharing L2	10 cycles	4 cycles	14 cycles

Table 3: Cache latency configurations

As shown in Table 3 and Figure 3, to calculate the latency times of L2 private cache memory:

As an example in the Core0, the latency times are calculated summing the cache latency from Table 3 plus the access time on bus in each circuit in Figure 3 (1 + 1 + 2 = 4) equal to 14 cycles.

In the Core1, the latency times are calculated summing the cache latency from Table 3 3 plus the access time on bus in each circuit in Figure 3 $(1 + 2 = 3)$ equal to 13 cycles.

In the L2 shared cache memory, the latencies are shown directly in Table 3, equal to 14 cycles.

5. Results

To represent the architecture on the graphics it was used the following BT/CG legend which represents the kind of test, shared cache memory equaling to a "par" and private cache memory equaling to "ind".

As shown in Figure 4, when increasing the numbers of cores, fewer cycles are used to the execution of the application. A greater asymmetry of latencies simulates a better access controlling on the bus resulting in a greater contention of L2 private cache memory compared to shared cache memory.

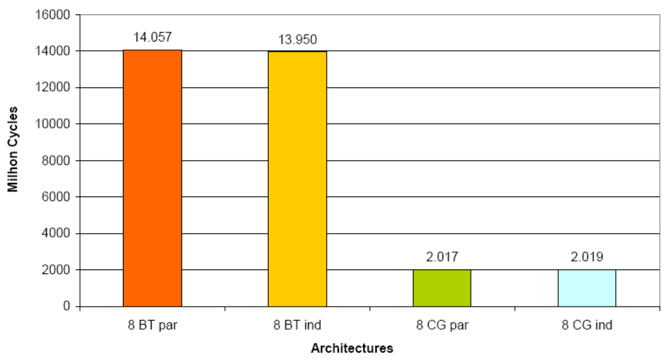

Figure 4: Execution cycles

The values are given on millions of cycles. The standard deviation$\sigma < 0.00001\%$ qualifies the measure.

The cycles in Figure 4 and execution steps in Figure 5 show that there is not difference on architecture with shared cache or private cache memory. It is because they have small distances between the processors within the chip then the asymmetry introduced in the cache memory is not noticeable to the point of influencing the applications executions behaviors.

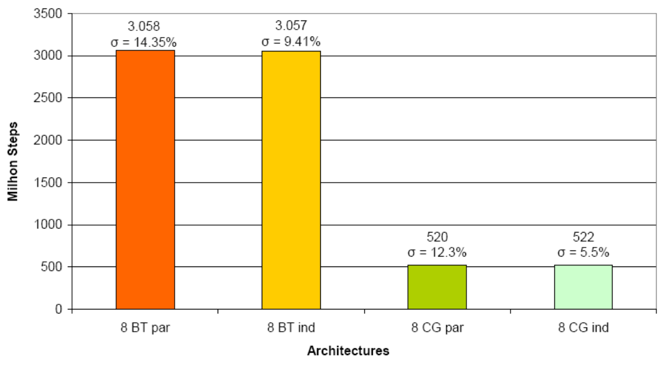

Figure 5: Execution step (instructions) numbers

In Figure 6, the expectative was to have a greater number of miss, due to the reduction of the cache size but this did not happen. The results show that using architecture with asymmetric access time, the miss number was not impaired, being equivalent to the access time with share symmetric cache.

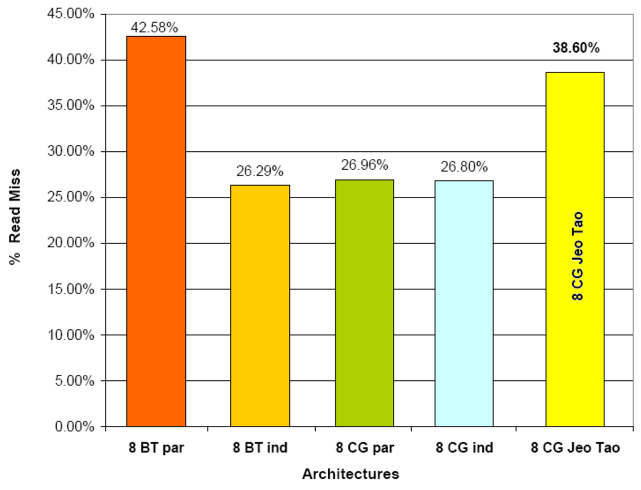

Figure 6: Read miss on L2 cache memory

The BT standard deviation was $\sigma < 2.74\%$ and CG was $\sigma < 0.25\%$. The BT simulation implements more cache writing and it is justified in Figure 6 with a higher number of miss read for the architectures.

As shown in Patterson and Hennessy [8], increasing the size of cache implies in a lesser number of errors.

The proportionality of the cache size per core is not the same, as shown by the hardware configuration in the experiment. In the L2 private cache memory we have 256 KB per core while in the L2 shared cache memory, this represents 50% less memory hardware necessary and the errors caused for miss in the memory cache for conflict and capacity are bigger. Observe that using the asymmetric latency on L2 private cache, decreases the effect of the conflict miss.

For the CG application the miss behavior is similar. Compared to the results obtained by Jie Tao *et al.* [12] with miss of 38.6% in the CG (in the same environment), we see that the data obtained is relevant.

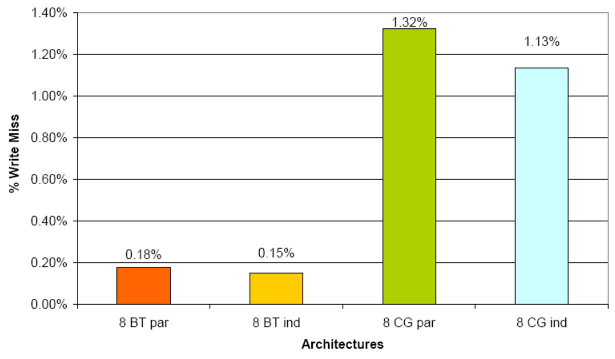

Figure 7: Write miss on L2 cache memory

In Figure 7 we verified that the writing miss rate is within acceptable values, maintaining the proportionality. The percentage of correct answers in writing of 98,68% to 99,68% is a good result. The range of the standard deviation of $0.01\% < \sigma < 0.02\%$, shows a good accuracy.

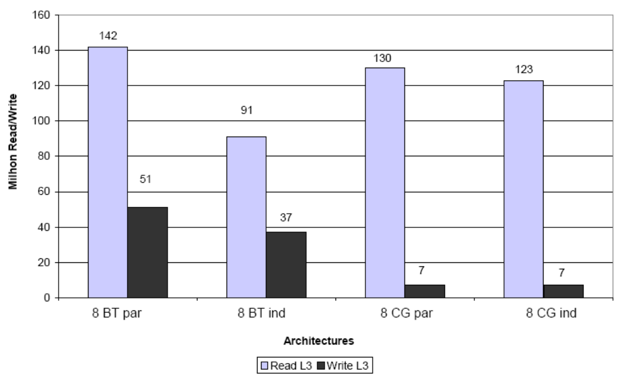

Figure 8: L3 Cache read/write for BT and CG benckmarks

The use of L3 cache in Figure 8 shows that a great part of share data in reading happened on this cache, compared with the L2 cache memory behavior, indicates that data on L3 cache memory were also shared through this level.

Figure 9 the result of 98.79% and 96.78% on L3 cache memory read hits, in CG Benchmark, is a result of conflict factors , such as miss collisions associated with cache update policy and the use of write-back, as shown on Patterson and Hennessy [8] and Jie Tao *et al.* [12].

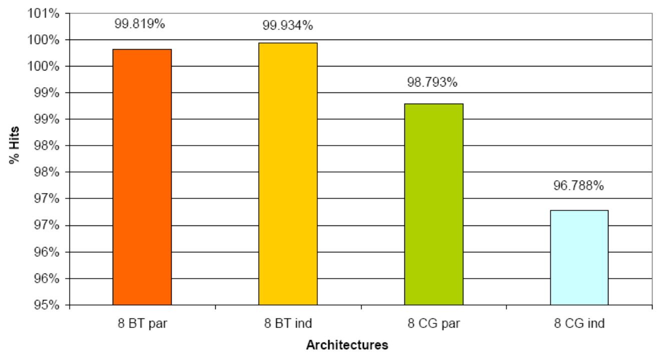

Figure 9: L3 Cache read hits

Figure 10 shows a level of acceptable hits, if we compare with equivalent experiments again [12] (that have values of 97,83% hits for the worst case) then 99,89% in this experiment is very acceptable, for the CG behavior on 8 cores where data meet shared in L3 cache memory exclusively.

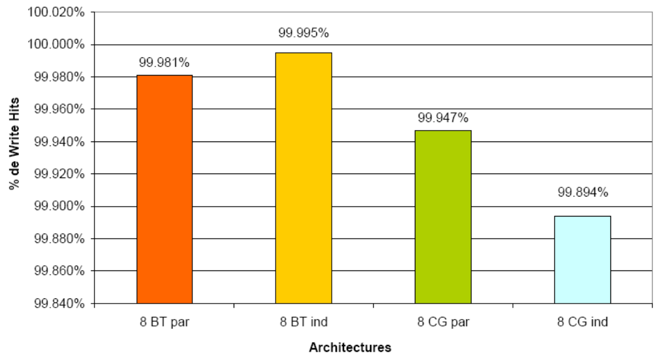

Figure 10: L3 Cache write hits

6. Conclusions

The use of asymmetric latency on L2 private cache memory with few cores, we did not verify a significant reduction on the conflict effect in the cache, but it seems that the access control to the bus with this technique tend to be more efficient for the BT and CG tests applied in experiments. On Figure 6 of the previous section, it is suggested that the asymmetry time use, might allow a more efficient use of resources of hardware than the symmetric approach, however it will still need more tests with other types of applications.

When you do not keep the proportionality size cache memory per core it might lead to a higher miss number caused by conflict and by capacity. The effect of the use of asymmetric latency is the reduction of these conflicts without significant increase of errors, compared to the symmetric cache latency model. The fact of obtaining a inferior number of read miss to the one obtained by Jie Tao *et al.* [12] experiment of 38,6% for the CG application, compared with 26,96% for symmetric cache and 26,80% for shared cache with a lesser hardware, indicates that there are relevances in the results of performed experiment.

For future works, a simulation with16 and 32 cores is still necessary for RubyGems simulator model [10] within Simics to be able to have the routing and also to execute other simulations models of the NAS, to observe if the behavior of the architecture will be the same with other applications formats.

References

[1] H. Aboudja and J. Simonson. Real-time systems performance improvement with multi-level cache memory. *CCECE '06. Canadian Conference on Electrical and Computer Engineering*, pages 78–81, May 2006.

[2] M. A. Z. Alves, H. C. Freitas, and P. O. A. Navaux. Investigation of shared l2 cache on many-core processors. In *ARCS '09 - 22th International Conference on Architecture of Computing Systems 2009*, volume 1, pages 21–30. VDE Verlang, 2009.

[3] AMD. Product brief: Second-generation amd opteron, January 2009.

[4] D. Bailey, E. Barszcz, et al. The nas parallel benchmarks. Technical Report RNR-94-007, Nasa, March 1994.

[5] P. Chaparro, J. González, G. Magklis, Q. Cai, and A. González. Understanding the thermal implications of multicore architectures. *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, 18(08):1055–1065, August 2007.

[6] H. C. Freitas, P. O. A. Navaux, and T. G. S. Santos. Noc architecture design for multi-cluster chips. *IEEE International Conference on Field Programmable Logic and Applications*, pages 53–58, September 2008.

[7] J. Held, J. Bautista, and S. Koe. Intel corporationfrom a few cores to many: A tera-scale computing research overview. *Intel Technology Journal*, 2006.

[8] J. L. Henessy and D. A. Patterson. *Computer Architecture - A quantitative approach*, volume 1. Morgan Kaufmann, 4 edition, 2007.

[9] C. A. F. D. Rose and P. O. A. Navaux. *Arquiteturas Paralelas*, volume 1 of *Livros Didáticos 15*. Sagra Luzzatto, 1 edition, 2003.

[10] RubyForge. Rubygems manuals. http://docs.rubygems.org/.

[11] F. N. Sibai. On the performance benefits of sharing and privatizing second and third-level cache memories in homogeneous multi-core architectures. *Microprocess. Microsyst.*, 32(7):405–412, 2008.

[12] J. Tao, M. Kunze, and W. Karl. Evaluating the cache architecture of multicore processors. In *PDP '08: Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, pages 12–19, Washington, DC, USA, 2008. IEEE Computer Society.