Investigation Through Set Associativity on Shared L2 Cache Memory in a Multi-Core Chip Architecture

Marco A. Z. Alves, Philippe O. A. Navaux
Informatics Institute - Universidade Federal do Rio Grande do Sul - UFRGS
{marco.zanata, navaux}@inf.ufrgs.br

Abstract

Memory latency has been one of the most performance bottleneck in single-core processors and nowadays on multi-core processors. As future one can spot that many cores shall be available with great performance, but still needing more cache performance to obtain good memory throughput for all processing cores. Considering the present context, this paper examined the effect of isolated and shared L2 cache by two cores, in a 32 cores chip multiprocessor, where the results shows that even with reduction of cache miss, the final performance is dominated by the cache latency. Thus, some evaluations on set associativity were made in order to obtain more cache hits and thus, try to lead a final performance gain. The results consider increase on set associativity on both L1 and L2 cache, where some cache miss reductions where observed but considering the increase of cache latency it was not enough to lead a performance gain.

1. Introduction

Nowadays, computer systems looks for high thread level parallelism, this can be observed mainly on the increase of multi-core processors projects, which even having more than one core, are designed in the same physical area of an single-core processor [10], what is just possible with the usage of more simple processing units for each core. Besides more parallelism on its same size, multi-core processors presents other attractive properties as simpler control unit and less power consumption in many cases. The future high-performance processors shall have tens or even hundreds of cores [6]. In this context, increases on the number of cores should consider that progress must be made in all areas to support this type of technology as the memory hierarchies, the coherence control, the interconnection networks and others.

Nowadays some commercial multi-core processors are using shared L2 cache [3], but at same time, others multi-core architectures are adopting isolated L2 cache [11] for each core. Both architectures, sharing or isolating L2 cache have some great and bad points, but it is not clear how should be the best cache memory organization for multi-core or even many-core processors. The work presented on [2] shows that for some applications the amount of cache miss can be reduced using shared L2 cache, but on the other hand, if the application is not architecture aware, this sharing can lead to a decrease on the final performance.

The set associativity is a well know cache memory mapping strategy where each block can be placed in *n* different ways, where *n* is the set associativity used, its adopted normally to increase the cache hit rate [5], but this strategy has the drawback of the hit time increase. The results presented on [4] aim to study the performance of shared cache on multi-threaded architectures using trace driven simulators, the study shows that both cache size and set associativity needs to increase as the number of threads increase in order to maintain comparable performance.

The present paper aims to establish the performance impact of some cache memory parameters on processors with multiple processing cores considering real memory latencies, this way studying how the memory cache shared between multiple cores and the increase of set associativity affects the performance, considering the improvement on the cache hit rate and estimates on cache latency. In this paper the instruction set architecture simulation was applied to study the system performance of a 32 cores chip multi-core, with shared and isolated L2 cache organizations on the first experiment and with changes in the set associativity on the second experiment, executing NAS parallel benchmark to represent a typical workload.

This paper is organized as follows, the section Methodology brings some methodological considerations, the model and latency estimates, simulation, and the workload information, the Results and Analysis section presents the experimentation results and data analysis, the Conclusion and Future Work section presents the main conclusions and some interesting future work topics, then finally the Acknowledgement is presented.

L2 Slices	Cores per Slice	Size per Slice	L1 Cache Set Assoc.	L2 Cache Set Assoc.	L1 R/W Latency (ns)	L2 R/W Latency (ns)	L1 R/W Penalty Cycles	L2 R/W Penalty Cycles
32	1	1 MB	2 Ways	8 Ways	0.74	1.77	2	4
16	2	2 MB	2 Ways	8 Ways	0.74	2.16	2	5
32	1	1 MB	4 Ways	8 Ways	0.78	1.77	2	4
16	2	2 MB	4 Ways	8 Ways	0.78	2.16	2	5
32	1	1 MB	2 Ways	16 Ways	0.74	2.62	2	6
16	2	2 MB	2 Ways	16 Ways	0.74	2.83	2	6

Table 2. Modeled Cache Systems and Cache Organizations.

Component	Parameter	Value
Chip Multiprocessor	Cores	32
	Execution	In-Order
	Frequency	2 GHz
	CPI	1.0
L1 Cache	Size	32 KB + 32 KB
	Line Size	32 B
	Set Associative	Variable (Table 2)
	Feature Size	45 nm
	R/W Latency	Variable (Table 2)
	Replacement	LRU
	Write Policy	Write-Through
L2 Cache	Size	1 MB per Core
	Line Size	64 B
	Set Associative	Variable (Table 2)
	Feature Size	45 nm
	R/W Latency	Variable (Table 2)
	Replacement	LRU
	Write Policy	Write-Through
Main Memory	Size	1 GB
	Feature Size	65 nm
	Latency	100 Cycles

Table 1. Base Components Parameters.

Name	Description	Operations per Thread
BT.W	Block Tridiagonal Solver	1.14 Mop/s
CG.W	Conjugate Gradient	0.42 Mop/s
MG.W	Multigrid	1.14 Mop/s
EP.W	Embarrassing Parallel	0.08 Mop/s
SP.W	Solve Scalar Pentadiagonal System	0.74 Mop/s
LU.W	Decomposition LU	1.20 Mop/s
IS.W	Integer Sort	0.09 Mop/s
FT.W	3D FFT PDE	0.58 Mop/s
UA.W	Unstructured Adaptative	0.01 Mop/s

Table 3. Workload Description.

2. Methodology

Performance evaluation is required at every stage in the life cycle of a computer system, including its design and manufacturing [7]. On design of an processor architecture some different evaluation techniques can be used, but the appropriate selection of techniques, performance metrics and workloads for a system could be used to compare architectural and organizational modifications. In terms of evaluation techniques three mainly methodologies may be considered: analytical modeling; simulation and; measurement.

For a general purpose computer architecture evaluation with complex memory organization, simulation offer good features once it does not need prototyping nor hard and imprecise analytical formulations. Thus, for our study, all the evaluations were made using simulation techniques.

2.1. Modeling and Simulation

The simulation environment used was Simics (ver. 3.31), from Virtutech [9], which was chosen because it is a full system simulator platform at the instruction set level. Thus, the results of execution time are measured in executed instructions and cycles. The number of cycles is given by the number of instructions executed added to the stall cycles generated by the latency of each architecture component.

On the first proposed experiment consider a base CMP (Chip Multiprocessor) with 32 cores, each core with its own L1 cache simulated with isolated and shared L2 Cache. Table 1 show the parameters used on the base system, the modeled parameters where based on a chip multiprocessor featured in 45 *nm* integration technology and with the main memory build on 65 *nm* technology.

The simulated latencies for all memory hierarchy models considers estimates based on the parameters used on each simulation, where the latencies shall have differences as the modeled cache change. The simulation latencies modeled were obtained based on CACTI memory modeling tool version 5.3. Table 2 brings the specific values for each cache parameters evaluated.

2.2. Workloads

Application benchmarks are formed with representative subset of application functions of well know algorithms. Thus, such benchmarks are generally described in terms of functions to be performed. In this paper, the workload NPB (NAS Parallel Benchmark) [8], was used for all evaluations, this benchmark is formed with a set of applications where each application represents one different kernel of numerical methods described in details on Table 3. The NPB workload used was version 3.3 compiled with SunStudio 11 with -*fast* performance parameter.

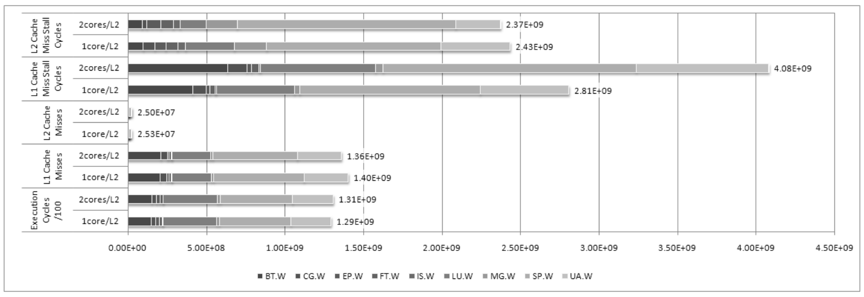

Figure 1. Execution cycles, caches misses and cache miss stall cycles for NPB benchmark execution on systems with isolated and shared L2 cache.

3. Results and Analysis

This section presents the results obtained running the NPB workload applications on the modeled experiments, obtained based on one execution of each experiment, where each experiment was executed once previously in order to reduce transient effects as cold start effects [1].

The first evaluation was studying the L2 sharing impact on multi-core processor, considering one 32 cores CMP, with two organizations, the first considering one L1 cache and one L2 cache for each core (1core/L2). On the second using one core with its own L1 cache and each L2 cache shared for two cores (2cores/L2). Both organizations modeled with the same total amount of L2 cache memory, using the total of 32 MB, with 1 MB for each core.

Figure 1 show information about the first two simulations considering an isolated and a shared L2 cache. On this experiment, one can observe that execution cycles increases 1.03% as the L2 cache becomes shared for two cores and it happens even with the reduction of 1.39% on the number of cache misses on L2 cache. However, this behavior is very easily explained when the cache memory access time latency is considered, where the latency from the first for second simulation had 22.03% of increase.

The second experiment evaluates the impact of the increase of set associativity on the two previous analysed cache organizations. Thus, based on the first experiment parameters, the set associativity parameter was doubled for both L1 and L2 cache, once a time, in order to increase the cache hit rate and thus, increase the execution performance. Figure 2 show the results of cache miss and lost stall cycles generated by cache misses for L1 change on set associativity, where the experiment with L1 cache using 4 ways set associativity had impacted on L1 cache miss decrease of 3.74% on 1core/L2 and reduction of 1.54% on 2cores/L2, and on the L2 cache miss increase of 29.75% and reduction of 0.60% for 1core/L2 and 2cores/L2 respectively. In terms of wasted stall cycles by cache latency, the total of cycles wasted was 11.81% greater for 1core/L2 and 1.19% lower for 2cores/L2 comparing the modified L1 set associativity with the base configuration used on the first experiment. Figure 3 show the results of cache miss and stall latency generated by cache misses for L2 change on set associativity, where the L2 cache using 16 ways set associativity which achieved L2 cache miss increase of 9.06% on 1core/L2 and reduction of 25.41% on 2cores/L2. In terms of wasted stall cycles by cache latency, the total of cycles

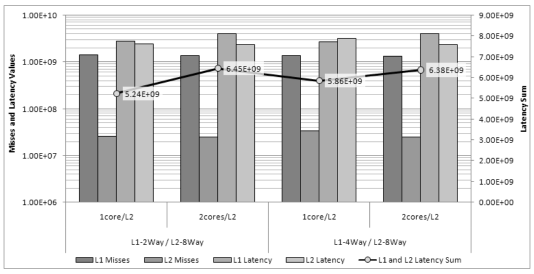

Figure 2. Cache Misses and Stall Latency by increase on set associativity of L1 cache.

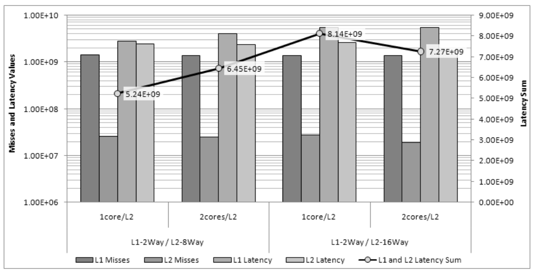

Figure 3. Cache Misses and Stall Latency by increase on set associativity of L2 cache.

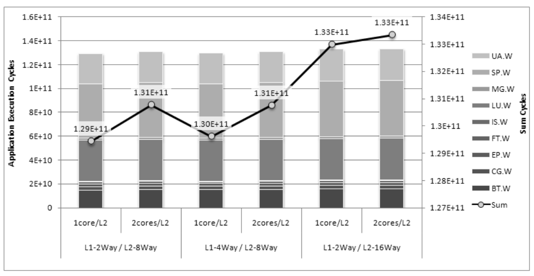

Figure 4. Execution cycles for NPB benchmark execution on different set associativity.

wasted was 6.79% greater for 1core/L2 and 26.20% lower for 2cores/L2 comparing the modified L2 set associativity with the base configuration used on the first experiment. Figure 4 show the plots about the execution cycles for each execution and each application. Its clear that even changing the set associative, the execution cycles for the NPB benchmark execution increased comparing the configuration of isolated L2 cache with shared L2 cache, where the execution cycles suffered increases on both L1 and L2 changes of set associativity, where the Figure 2 and Figure 3 associated with the cache parameters (Table 2) explains the behavior of the increase on execution cycles.

The performance was not improved by neither decreasing the cache miss by sharing the L2 cache for more cores, nor increasing the set associativity, what is explained by the increase of latency that cache suffers when its size or the associativity increases.

Considering the sharing of L2 cache, it lead the system to a decrease on cache misses, but on the other hand, as the L2 cache size increases its latency also increases, thus, the execution time of all experiments increased. The set associativity also had great impact on decrease of cache miss when the cache was shared for two cores, where the change on L2 cache associativity showed good results in terms of cache hit, but all modifications on set associativity where not able to reduce the execution time, once the cache hit does not pay the increase on latency.

4. Conclusion and Future Work

On the context of chip multi-core processors, this paper evaluated some cache memory organizations in order to investigate the L2 cache sharing impact on a CMP, and also try to examine the impact of set associativity on isolated and shared L2 cache memory.

On the first experiment of sharing one L2 cache for each 2 cores, it lead the system to a decrease on cache misses, the same occurs when the set associativity where doubled, but as the L2 cache size increases and as the set associativity increases too, the cache memory latency also increases, thus, the execution time of all experiments increased, showing that execution on both cases where degraded by cache latency.

For future work, we consider to expand the experimentation for more cores sharing the same L2 cache slice, and combine the sharing with the increase of set associativity and block size, in order to study the impact of all factors trying to reduce the cache miss rate and thus execution time.

5. Acknowledgment

This work was supported in part by Projeto Universal and CNPq (Brazilian Government).

References

[1] A. R. Alameldeen, C. J. Mauer, M. Xu, et al. Evaluating non-deterministic multi-threaded commercial workloads. *Computer Architecture Evaluation using Comercial Workloads*, 2002.

[2] M. A. Z. Alves, H. C. Freitas, F. R. Wagner, and P. O. A. Navaux. Influência do compartilhamento de cache l2 em um chip multiprocessado sob cargas de trabalho com conjuntos de dados contíguos e não contíguos. In *Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD)*, 2007.

[3] J. Chang, S. Member, and M. Huang. The 65-nm 16-mb shared on-die l3 cache for the dual-core intel xeon processor 7100 series. *Journal of Solid-State Circuits*, 42(4), 2007.

[4] Y.-Y. Chen, J.-K. Peir, and C.-T. King. Performance of shared cache on multithreaded architectures. *IEEE Euromicro International Conference on Parallel, Distributed, and Network-Based Processing - PDP*, pages 541–548, 1996.

[5] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Elsevier, Inc., United States of America, fourth edition, 2007.

[6] Intel. Advancing multi-core technology into the tera-scale era. *Intel Teraflops Research Chip*, 2007.

[7] R. Jain. *The art of computer systems performance analysis*. J. Wiley, New York, 1991.

[8] H. Jin, M. Frumkin, and J. Yan. The openmp implementation of nas parallel benchmarks and its performance. In *NAS Technical Report NAS-99-011*. NASA Ames Research Center, 1999.

[9] P. S. Magnusson, M. Christensson, J. Eskilson, et al. Simics: A full system simulation platform. *IEEE Computer Micro*, 2002.

[10] K. Olukotun et al. The case for a single-chip multiprocessor. *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2006.

[11] L. Peng, J. Peir, T. K. Prakash, Y. Chen, and D. Koppelman. Memory performance and scalability of intel's and amd's dual-core processors: A case study. In *International Performance, Computing, and Communications Conference (IPCCC)*, 2007.