

# Evaluating the interconnection latency costs on the performance of a CMP with multisliced L2

Mario Donato Marino - *e-mail: mario@regulus.pcs.usp.br*  
Computing Engineering Department - Polytechnic School - USP

## Abstract

*Nowadays market is moving to have multiple cores on the same chip (Chip Multiprocessors - CMP). Each core has its own processor, its private L1 and a L2, which can be private or shared with other processors. Several solutions with 2 and also with 8 cores are already available in the market. The tendency is to have L2 sliced and its L2 slices connected through a crossbar. What is the impact of the crossbar over the performance of these CMPs? We have constructed a model to answer this question. This model is evaluated with a full-system simulation, using 32 processors, under SPLASH-2 benchmarks. Previous results show that the execution time is increased of about 15% when the interconnection latency is increased from the ideal case (0 cycle) to 14 cycles.*

## 1 Introduction

The processor market is moving to have multiple cores on the same chip, also named chip multiprocessors - CMP [20]. For example, AMD [9], IBM [10] and Intel [11] have dual-core solutions that are already on the market, not only for commercial and industrial ones, but also for domestic and gaming [12]. A eight-core [17] solution is already available. Soon we will also have for the server market CMP based machines with 4 and also with 16 [21] or more cores. The market is moving towards CMPs because [1, 2, 3, 4, 5, 16, 19] they are very attractive: better dissipation, scalability, performance [13] and have less energy [3, 18] consumption when compared to a wider one-

processor.

The presence of multi-cores increases the pressure for more data, and also the latencies due to wire delays [3, 19]. The way to reduce both aspects is by focusing on sharing some parts of their caches and maintaining others private.

In terms of latency, we should consider either the latency of each cache L1 or the latency of each L2. In order to have good performance, designs and implementations of L2 through multisliced banks have been preferred [1, 2, 3, 6, 7, 13] due to their low latency. The main goal of the paper is to evaluate the impact of the latency over the speedup of a parallel program over a multisliced L2 organization of a CMP. Through a full-system simulation and running SPLASH-2 [15] benchmarks, we have modelled a CMP with a multisliced L2 varying the latency of the interconnection.

The rest of the paper is organized as follows. Section 2 describes possible multi-core organizations. Section 3 discusses the methodology and Section 4 presents results. Section 5 concludes our work.

## 2 Interconnection of CMPs - Latency of L1pL2s

In this section we describe some configurations considered for tiled CMPs. They are based on a tile replicated in a 2-D mesh configuration, where each tile contains:

- (i) its processor, with a L1 instruction and L1 data caches;
- (ii) one L2 cache;

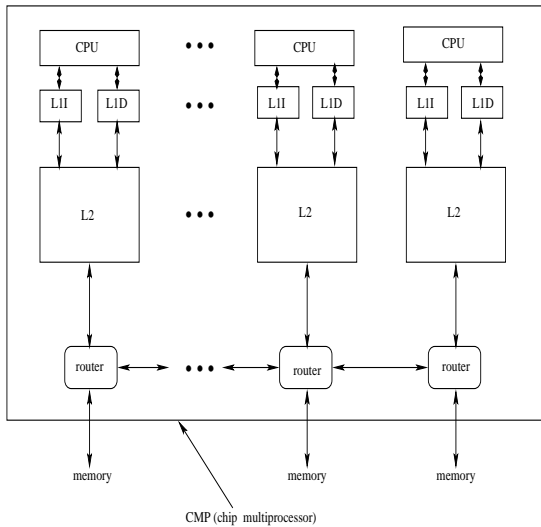


Figure 1: private L1 and L2

(iii) a connection to the network. Figures 1 and 2 illustrate that.

They differ on the L2 implementation which can be private or shared, i.e., in the latter case slices of L2 cacheclusters. Then, let's name them according to their configuration: (i) private L1 data cache and private L2: *L1pL2p*; (ii) private L1 data cache and shared L2 slices: *L1pL2s*. Due to fabrication aspects and due to better speedups a big one L2 cache idea has been abandoned[1, 2, 13].

The configuration (i) is illustrated in figure 1, with its private L1 and L2 caches.

We are interested in configuration (ii), i.e. in *L1pL2s*, because some previous papers [1, 9, 13] have shown the superiority of the scheme *L1pL2s* over the *L1pL2p*. Figure 2 illustrates a CMP with some shared clusters L2, each of these shared by 2 processors (CPUs). The tendency of the market confirms dual-core Power5[10] and Opteron[9], and eight-core Niagara[17] are examples of that. *L1pL2s* cacheclusters communicate with the others by the network, with the distributed directory held as a replicated set of L2 tags. Each CPU has its private L1 data and L1 instruction caches, which are connected to a shared cluster L2: each cluster of L2 is shared among 2, 4 or more processors, i.e., a set of processors shar-

ing L2. The main advantage of having a set of processors sharing L2 is that one of them can reuse a line brought by the other processor from memory. So, if there is a miss in L1, the line is searched in a L2 cachecluster through the network, which it's a way to move lines from remote L2s. If the line is found in the L2 cachecluster, we avoid the congestion of the network which connects them. If the line is not found in the cluster, it can be found in one of the other shared L2 cacheclusters, minimizing the off-chip accesses to the main memory. The latency of a L2 miss depends not only on the number of hops needed, but also on the network congestion from the other L2 cacheclusters and also on the coherency messages (to maintain the clusters coherent).

Resuming, according to memory hierarchy, we can model a path going from L1 to memory:

1. if data isn't in L1, there's a miss; thus, there's a penalty for that;
2. if data isn't in one L2 cachecluster, it should be tried from other one. So, at least it's necessary one hop to cross the cross-bar plus the penalty of L2 cachecluster. If data is not found in the next cachecluster, it must be searched on the others; as a result, other hops will be necessary to find it, increasing the latency of all this operation. Concluding, if there are n caches, n-1 hops plus the latency of L2 cachecluster are needed to find it;
3. the time to perform each invalidation (MESI) should also be considered.

The number of processors sharing a clustered L2 can be increased for 4, 8 and even 16 processors, obviously depending on the implementation technology available. When it comes to locality, considering 2 processors sharing L2, as shown in figure 2, the fact of sharing L2 favors the probability of finding an useful shared line used by both processors, reducing the number of L2 misses for the specific line. On the worst case and according to the pattern of the addresses accessed (which is directly dependent of the program) we can have more capacity

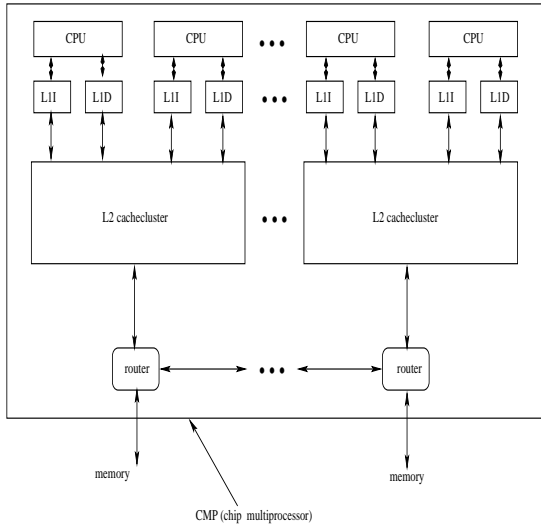


Figure 2: CMP organization: private L1 and 2 processors sharing a clustered L2

and conflict misses, clearly reducing possible speedups.

### 3 Methodology

Since the main goal of the paper is to evaluate the impact of the interconnection, we have changed the the latency of the crossbar and evaluate its impact on the execution time.

#### 3.1 Simulation environment and parameters

We use Simics 2.0.16 [14] to perform a full system simulation. We simulate the L1pL2s for 32 processors arranged in 8 L2 cache clusters of 4 cores, varying the latency of the crossbar. Each CPU is a UltraSparc III, in-order issue, with an ideal pipelining and  $IPC = 1$ .

Concerning L1, we assume 64 kB cache size (32 kB of data + 32 kByte of instructions), 64B-line write-back, 4-way associative, with latency of 3 cycles. LRU is adopted as the replacement mechanism of L1.

When it comes to each slice of L2, we assume 1 MB cache size for each cluster of L2, 128B-line write-back, 4-way associative, with a 10-cycle

Table 1: summary of L1pL2s simulation parameters

parameter	considered values
Processor	UltraSparc III, in order, IPC=1
# of processors	32 processors
L1 cache line size	64 B
L1 cache associativity	4-way
L1 cache size	64 kB , i.e.,32 kB d + 32 kB i
L1 latency	3 cycles
L1 replacement policy	LRU
L2 cache line size	128 B
L2 cache associativity	4-way
L2 latency	10 cycles
L2 replacement policy	LRU
MESI transaction	10 cycles
1-hop latency	0 to 14 cycles, step=2
# of processors sharing L2	4 processors/cluster
cluster of shared L2	8
cache size:L2 cache cluster	1 MB/cluster
memory latency	200 cycles

latency in case of a hit of L2, and in order processor. The number of processors sharing L2 is 4. We have changed the latency of the interconnection of the multisliced L2 from 0 cycle to 14 cycles, with step equals to 2, to model the intra-chip network among the L2 cacheclusters.

It's important to notice that the MESI transaction latency is fixed to 10 cycles and that we assume infinite memory with a 200-cycle latency. Table 1 just summarize all parameters we used in our experiments with L1pL2s.

#### 3.2 Benchmarks evaluated

In this subsection we comment the workloads used to evaluate and compare the schemes proposed.

We have chosen all the applications from SPLASH-2 [15] benchmarks, which have been used by the research community for years for evaluation of shared memory architectures. All benchmarks were compiled with gcc 3.3.0 and run for class-A input sizes. The procedure used to evaluate our benchmarks was ever to run them immediately after booting the operating

Table 2: benchmarks evaluated

benchmark	input parameters	description
Barnes	16384 part., k=123	Barnes-Hut
FMM	2 clusters, 16384 particles	Adaptive Fast Multipole
Ocean	n=258, e=1e-07 r=20000, t=28800	Ocean Simulation
Radiosity	batch, room	Hierarchical Radiosity
Raytrace	m=64, car.env	Ray Tracer
Volrend	input=head	Volume Renderer
Water	Nsquared, 512 particles	Water Simulation without Spatial Data Structure

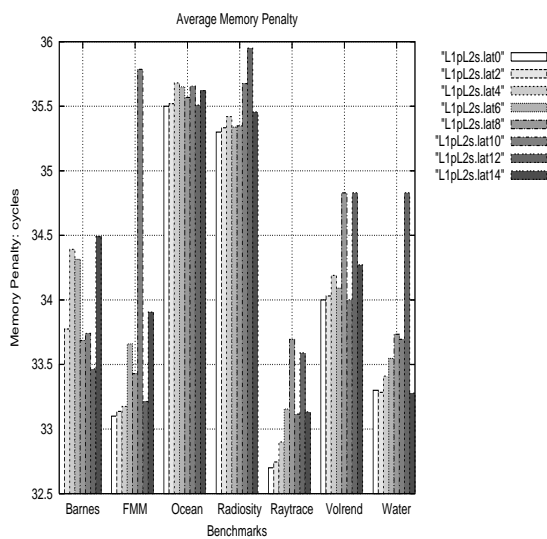


Figure 3: average memory penalty - per program

system (Solaris 9). Time is measured in number of cycles. In order to reduce and minimize errors of measurement, we have run each simulation 10 times, although all the results haven't show significant variability[12] - 4.5 %.

Table 2 summarizes the parameters used by the SPLASH-2 benchmarks that we have run.

## 4 Simulation Results

We evaluate the results of the simulations considering the following parameters: average memory penalty, execution time (speedup) and

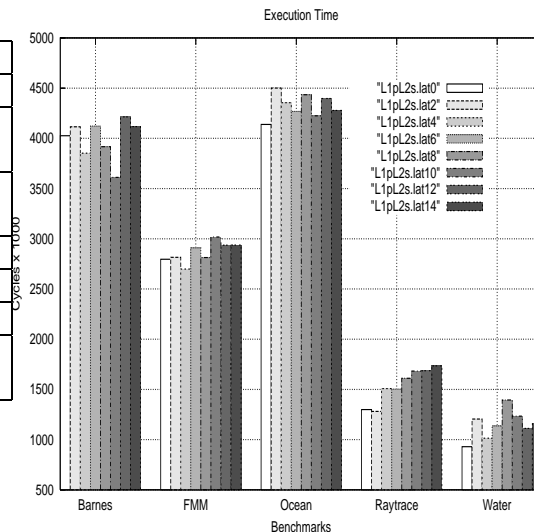


Figure 4: execution time - SPLASH2

number of coherency messages .

The first observation is that due to differences of magnitudes of the parameters evaluated among the benchmarks, just for scaling reasons, the execution time of some were separated from the others and analyzed separated.

By observing figure 4, for Barnes, we can notice an increase of up to 5% in the execution times. Since this benchmark has a small amount of communication among processors, i.e., most data fit in L1 [1, 2] (high hit rate), the increase of the interconnection latency doesn't affect the memory penalty, as can be noticed in figure 3. The number of invalidates is low, as observed in figure 7, resulting in a low influence on the execution time.

In the same way, an increase of up to 3% is noticed for FMM (figure 4). FMM has a similar behavior to Barnes related to L1 hit rate (high - [1, 2]), so minimal effects on the memory penalty are noticed. The number of invalidates is low (figure 7) and, as a result, a low influence on the execution time is presented.

We can notice an increase of up to 8% (figure 4) in the execution time for Ocean. Its L1 hit ratio is low [1, 2] then L2 is more effectively used; as a consequence L2 interconnection latency effects appear (figure 3) and can

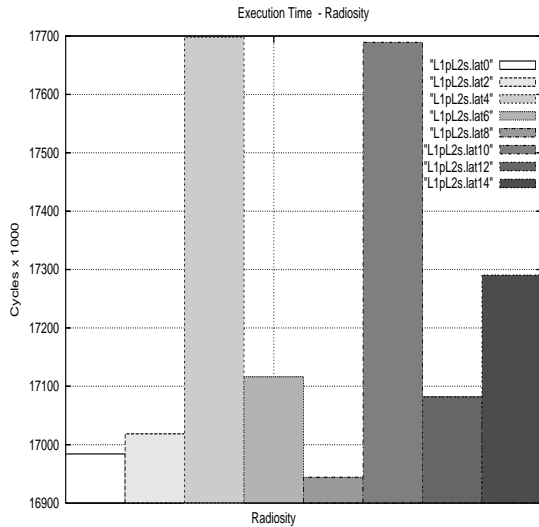


Figure 5: execution time - Radiosity

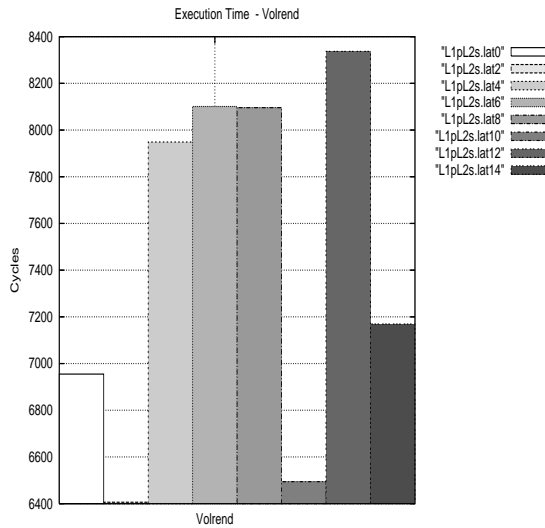


Figure 6: execution time - Volrend - SPLASH2

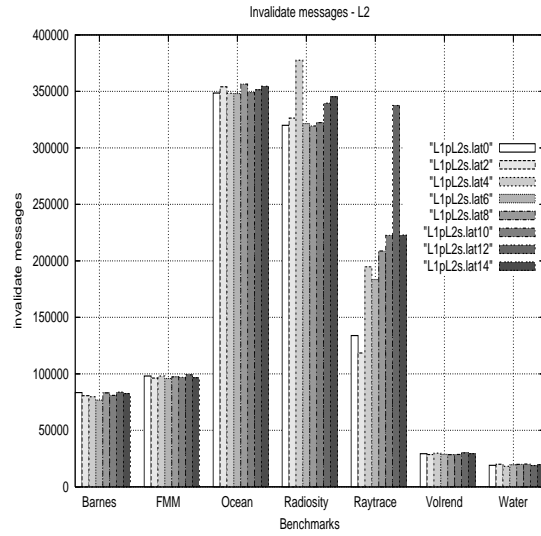


Figure 7: number of invalidates - L2

also be confirmed by observing the behavior of figure 7, where it's possible to notice a great number of L2 invalidates.

Radiosity presents a high hit rate (about 99.2%), which means that most data completely fit on L1, which can either be confirmed by observing the execution time in figure 5 or by observing the increment of the number of invalidates (figure 7) when the latency of the crossbar is increased from 0 to 14 cycles.

Raytrace has a low L1 hit ratio [1, 2], so the side effects of the L2 interconnection latency appear: an increment of up to 15% (figure 4) on the execution time. According to figure 7, we notice a great number of L2 invalidates which also confirms that. The average memory penalty (figure 4) is more affected than the previous benchmarks by the increment of the crossbar latency, because the invalidates are more time relevant when compared to the previous ones.

Volrend has a similar behavior of Raytrace, but has a higher L1 hit rate [1, 2]. We can notice an increment of the execution time when the latency is increased as shown in figure 6. Disconsidering the measurement variations, it's possible to observe an increment of about 10% when it's increased from 0 to 14 cycles, not so far from the measurement varia-

tion. The number of invalidates is maintained according to figure 7.

Water has a similar behavior to Volrend when it comes to the L1 hit rate [1, 2]. The execution times increase of up to 10% (figure 4). The number of invalidates is low, as shown in figure 7, so the effects on the execution time are low as well.

## 5 Conclusions

The goal of this study is to evaluate how the L2 interconnection latency affects the performance of the parallel applications of SPLASH-2, considering the most adopted implementation of L2 (multisliced). The main conclusion is that the interconnection latency affects the benchmarks with a low hit rate of L1, i.e., the benchmarks which use more effectively the L2 cache. Results show that the interconnection latency can decrease the performance in up to 15%.

We are likely to consider other configurations with different implementations of L2 and evaluate the interconnection effects.

We intend to make other tests also with other scientific benchmarks, such as SPEC, and other commercial benchmarks. Eventually, we intend to apply the victim cache proposed by Zhang and Asanovic [1] in our experiments.

## References

- [1] Zhang M., Asanovic K., Victim Replication: *Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors*, ISCA 2005, USA.
- [2] Chisti Z., Powell M.D., and Vijaykumar T.N., *Optimizing Replication, Communication, and Capacity Allocation in CMPs*, ISCA 2005, USA.
- [3] Kumar R., Zyuban V., Tullsen D.M., *Interconnections in Multi-core Architectures: Understanding Mechanisms, Overheads and Scaling*, ISCA 2005, USA.
- [4] Waingold E. et al, *Baring it all to Software: Raw Machines*, Computer 1997.
- [5] Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams, Taylor M.B et al, Proceedings of ISCA 2004.
- [6] Nagarajan R.N., Sankaralingam K., Burger D., Leckler S.W., *A Design Space Evaluation of Grid Processor Architectures*, ISCA 2001.
- [7] Sankaralingam K., Nagarajan R.N., Liu H., Kim C., *Exploiting ILP, TLP, and DLP with the Polymorphic TRIPS Architecture*, IEEE, 2003.
- [8] Cascaval C. et al, *Evaluation of a Multithreaded Architecture for Cellular Computing*, 2002.
- [9] <http://www.amd.com>
- [10] <http://www.ibm.com>
- [11] <http://www.intel.com>
- [12] <http://www.research.scea.com>
- [13] Barroso L et al., *Piranha: a scalable architecture based on single-chip multiprocessing*, ISCA, 2002.
- [14] <http://www.simics.net>
- [15] Woo S. Ohara M., Torrie E., Singh J.P., Gupta A.; The SPLASH-2 programs: Characterization and Methodological Considerations. In Proceedings of the 22nd. Annual Symposium on Computer Architecture, p. 24-36, 1995.
- [16] Rakesh Kumar, Dean M. Tullsen, Norman P. Jouppi, Parthasarathy Ranganathan. *Heterogeneous Chip Multiprocessors*, Computer, vol. 38, no. 11, pp. 32-38, November, 2005.
- [17] <http://www.sun.com>
- [18] Chun Liu, Anand Sivasubramaniam, Mahmut Kandemir. *Optimizing Bus Energy Consumption of On-Chip Multiprocessors Using Frequent Values*, pdp, p. 340, 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'04), 2004.
- [19] K. Olukotun et al., *The Case for a Single-Chip Multiprocessor*, in Intl. Conf. on Architectural Support for Programming.
- [20] Huh J. Burger D., *Keckler S. Exploring the design space of future CMPs*, PACT, 1997.
- [21] Villa F., Acacio M., Garcia J., Memory Subsystem Characterization in a 16-Core Snoop-Based Chip-Multiprocessor Architecture, technical report: <http://ditec.um.es/~jmgarcia/papers/CMP-final.pdf>