Compositional Memory Systems for Data Intensive Applications

A.M. Molnos[(*)(**)], M.J.M. Heijligers[(**)], S.D. Cotofana[(*)], J.T.J. van Eijndhoven[(**)]
[(*)] Delft University of Technology, The Netherlands
email:{ancutza, sorin}@dutepp0.et.tudelft.nl
[(**)] Philips Research Laboratories, The Netherlands
email:{marc.heijligers, jos.van.eijndhoven}@philips.com

Abstract

To alleviate the system performance unpredictability of multitasking applications running on multiprocessor platforms with shared memory hierarchies we propose a task level set based cache partitioning. We evaluate our approach on a CAKE platform with three Trimedias, one MIPS and a shared level 2 cache using a picture in picture benchmark. We compare the performance implications of two types of cache partitioning namely set based. Our experiments indicates that associativity based cache partitioning induces at least 30% performance degradation, whereas set-based partitioning provide 27% performance improvement when compared to non-partitioned cache scenario.

1. Introduction

The increase in size and complexity of state-of-the-art multimedia applications requires high performance hardware platforms with large memory bandwidth. To fulfil the bandwidth requirements usually memory hierarchies (caches) are used [3]. Besides the default probabilistic behaviour for sequential code, caches induce unpredictability because parallel tasks can influence each others performance by flushing each others data out of the cache.

Several alternatives of cache partitioning techniques are reported in literature. They require the processors' instruction set architecture or the compiler to be modified [4] which increase time-to-market when using standard processor cores or apply only to fully associative caches [7] [5]. To our knowledge, none of them compare the performance impact of different partitioning techniques.

In this paper we propose a cache partitioning mechanism to alleviate this problem. More in particular, the problem addressed in this article can be formulated as follows: given a multitasking application, concurrently running on a multiprocessor architecture with shared memory hierarchy, find a strategy that preserves the individual tasks' performance predictability. To address the problem we propose two ways to partition the cache and we evaluate our scheme using a picture in picture application running on a CAKE multiprocessor platform with shared level 2 cache and 3 Trimedia cores and 1 MIPS core.

2. Cache partitioning

The underlying idea of the proposed solution is to give exclusive cache parts to the applications' tasks that run on the multiprocessor platform. When knowing the cache behaviour of individual tasks, based on the compositionality property induced by cache partitioning, one can predict the performance for the overall system. To determine the number of cache misses for individual tasks, one can analyse the program code [2], or use simulation.

With respect to conventional cache organisation [3] we identify two main possible types of partitioning: based on associativity (called column caching in [1]) - every task gets a number of ways from every set of the cache (Figure 1a) and based on sets - every task gets a number of sets from the cache (Figure 1b).

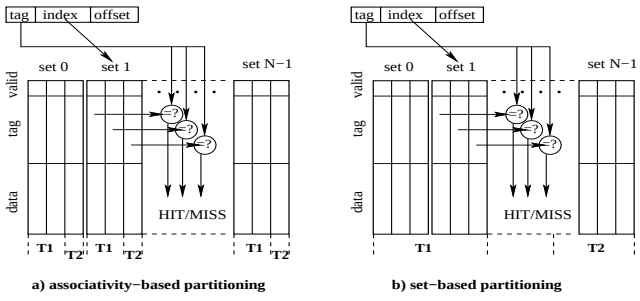

Figure 1. Types of cache partitioning

Both types of cache partitioning are suitable for addressing the considered predictability problem. With respect to

1530-1591/04 $20.00 (c) 2004 IEEE

implementation implications, associativity based partitioning requires a replacement policy that uses a task identifier, but the granularity of the partitioning is limited by the number of ways in a set. Set based partitioning implies a translation of the addresses that can be done using partition informations provided by the operating system but it has no granularity limitation. In the next section we compare the performance (measured in number of misses) of the two types of partitioning.

3. Experimental framework

The impact on performance of the cache partitioning was studied for the non-communicating tasks case. For our experiments we use a practical instance of the CAKE multiprocessor platform [6] having three Trimedias cores (for computation) and one MIPS core (for control) (with their level 1 cache) and a shared level 2 cache (1MB, 8 ways, 512B block size). We assume a picture in picture (PiP) application with three independent mpeg2 decoders running on different video streams as benchmark.

Corresponding with each of the five presented cases, the individual tasks' number of misses were measured for the cache configurations described in Table 1. For the shared case the applications are concurrently executed on the multiprocessor with no L2 cache partitioning. For the rest of the cases each task receive a certain amount of L2 cache. Using compositionality the individual misses can be added to obtain the total systems' performance. The simulation results are presented in Table 1 and Figure 2.

	all three decoders	decoder 1 128x128 201 frames	decoder 2 720x608 8 frames	decoder 3 352x240 24 frames
case1 (shared)	[1M 8ways] 1234639	- -	- -	-
case2 (set)	- -	[256k 8ways] 423941	[256k 8ways] 447481	[512k 8ways] 55712
case3 (assoc)	- -	[256k 2ways] 1213233	[256k 2ways] 1806211	[512k 4ways] 96005
case4 (assoc)	- -	[256k 2ways] 1213233	[384k 3ways] 475677	[384k 3ways] 362182
case5 (assoc)	- -	[384k 3ways] 392689	[384k 3ways] 475677	[256k 2ways] 1729862

Table 1. Number of level 2 misses

In terms of cache misses, Figure 2 indicates that partitioning at associativity level (case 3, 4, 5) gives a performance degradation of at least 30% when compared with the shared cache performance (case 1). This degradation is present because by dividing the cache every task uses a smaller part of it than in the shared cache case. However in set based partitioning case we experience a performance increase by 27% due to the disparition of inter-task conflicts.

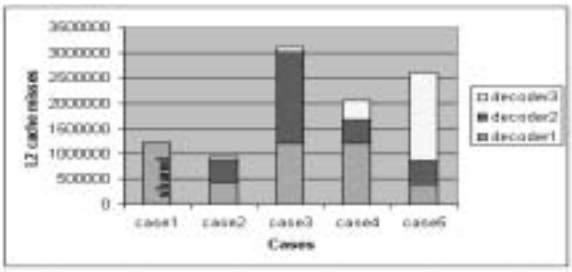

Figure 2. Cache misses for different partitioning types

4. Conclusions

We proposed a strategy, based on cache partitioning, that guarantees performance predictability of the overall multiprogrammed system when knowing the individual task's behaviour.

Experimental results suggest that set-based cache partitioning can bring to the system not only predictability but also a gain in performance. A case study for a picture in picture application shows up to 27% improvement in number of cache misses when compared to the shared cache case. We also found that associativity based partitioning always degrades memory hierarchy performance (at least 30%).

Finding the partitioning ratio for the best overall system performance gain will be addressed in future research.

References

[1] D. L. Chiou. *Extending the Reach of Microprocessors: Column and Curious Caching*. PhD thesis, Massachusetts Insitute of Technology, 1999.

[2] S. Ghosh, M. Martonosi, and S. Malik. Cache miss equations: An analytical representation of cache misses. In *International Conference on Supercomputing*, pages 317–324, 1997.

[3] J. L. Hennesy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, San Fransisco, CA, third edition, 2003.

[4] H. Muller, D. Page, J. Irwin, and D. May. Caches with compositional performance. *Proc. Embedded Processor Design Challenges*, pages 242–259, 2002.

[5] H. S. Stone, J. Truek, and L. Wolf, Joel. Optimal partitioning of cache memory. *IEEE Transactions on computers*, 41(9):1054–1068, 1992.

[6] P. Stravers and J. Hoogerbrugge. Homogeneous multiprocessing and the future of silicon design paradigms. In *In International Symposium on VLSI Technology, Systems, and Applications (VLSI-TSA), Proceedings*, april 2001.

[7] G. E. Suh, S. Devadas, and L. Rudolph. Analytical cache models with applications to cache partitioning. *Proc. Thirteenth IASTED International Conference on Parallel and Distributed Computing Systems*, 2001.