Proceedings of the 12th IEEE Real Time Congress on Nuclear And Plasma Sciencies. pp. 157-159. Valencia June 2001

Using locking caches in preemptive real-time systems

A. Marti Campoy, A. Perles Ivars, J. V. Busquets Mataix

Abstract-- **In multitask, preemptive real-time systems, use of cache memories make difficult the estimation of the response time of tasks, due to the dynamic, adaptive and non-predictable behavior of cache memories. But embedded and critical applications need the increase of performance provided by cache memories.**

This work presents a comprehensive method to use locking caches is these scenarios. Locking cache mechanisms allows to load and lock the content of the cache to ensure it will remain unchanged during execution. By ensuring this, the cache is totally predictable, thus conventional algorithms can be used to accomplish the schedulability analysis. In addition, a predictable cache allows a more accurate estimation of response time of tasks.

Nowadays, locking cache scheme is present is several commercial processors, and only minor hardware modifications would be necessary in order to obtain the best performance. To select the contents to be preloaded in cache, a genetic algorithm is developed. This algorithm selects the set of instructions to be locked in cache that give the better performance, and simultaneously estimate a tight upperbound of the response time of tasks, making simultaneously the schedulability analysis. Experimental results show that in a variety of scenarios, this method obtains better performance than non-locking caches in addition to simplify the analysis.

I. Introduction

CACHE memories help to increase system performance, but its unpredictable behavior raises two problems in real-time systems:

- It is difficult to obtain the Worst Case Execution Time (WCET) of a task. A task may remove its own instructions from cache (intrinsic interference). The execution time of one instruction depends on the cache contents.

- It is hard to obtain the response time of a task. In a preemptive multitasking environment, when a task resumes execution, the cache contents may be different, varying its execution time (extrinsic interference). The schedulability analysis must consider this effect when calculating the response time of any task in a system.

This work presents a technique based on locking cache to reduce the unpredictability of the cache. This enables the use of simple and well-known algorithms to make the schedulability analysis, in contrast to the complex analysis techniques presented in [1-5]. Compared to other works to improve cache determinism [6,7], the use of locking cache deals together with both problems: intrinsic and extrinsic interference, and not only one of them.

Even though our technique is focused on determinism and worst case scenarios, we have also obtained good average case performance trough carefully selection of the cache contents.

Authors belongs to the Department of Computer Engineerig, Technical University of Valencia, Spain. (e-mail: {amarti, aperles, vbusque}@ disca.upv.es).

II. Cache Model

Contemporary processors usually implement a locking cache scheme. In particular, our technique requires the following characteristics:

Cache can be totally locked or unlocked. When cache is locked, there are no new tag allocations.

If the processor addresses an instruction that is in the locked cache, this instruction is served from cache.

The processor may use an instruction buffer to take advantage of spatial locality even if the block is not in cache.

Cache can be loaded using a cache-fill instruction, selecting the memory block to be loaded.

Cache must be fully associative, at least in locking mode operation.

During system start-up, a small routine (basically a loop executing cache fill instruction) is executed to preload the cache. Once the cache is full, it is locked. Preloaded instructions can belong to any task of the system, and may be either large consecutive instruction sequences or small, individual separate blocks. When the system begins its full-operational execution, the instruction cache is loaded with a well-know set of instructions, and its contents will never change, eliminating both intrinsic and extrinsic interferences.

III. System Analysis

A. Schedulability analysis

We propose the Cached Response Time Analysis (CRTA) presented in [8]. CRTA uses recursively the following formula to obtain the response time of any task:

$$w_i^{n+1} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil x(C_j + T_{miss}) \qquad (1)$$

where W_i is the response time of task τ_i, C_i is the WCET of τ_i considering the effect of cache memory, B_i represents the time task τ_i is blocked, T_j is the period of task τ_j, hp(i) is the set of tasks with higher priority than task τ_i. Finally, the cache-related preemption delay (T_{miss}) is easily calculated since the cache contents are unchanged during execution.

B. *Worse Case Execution Time (WCET)*

If order to obtain the WCET we use an extension of the path analysis using the cache-control flow graph [9]. It is practical since there are no changes in cache contents during task execution. The execution time of an instruction is:

-For a vertex V_i belonging a block loaded and locked in cache, its execution time E_i is: $E_i = T_{hit} * I_i$

-For a vertex V_i belonging a block not loaded nor locked in cache, its execution time E_i is: $E_i=T_{miss}+(T_{hit}*I_i)$

where T_{miss} is the time of execution from main memory, T_{hit} is the time of execution from cache memory and I_i is the number of instructions of vertex V_i. Fig. 1 shows an example of building the c-cfg and the expression to calculate the WCET.

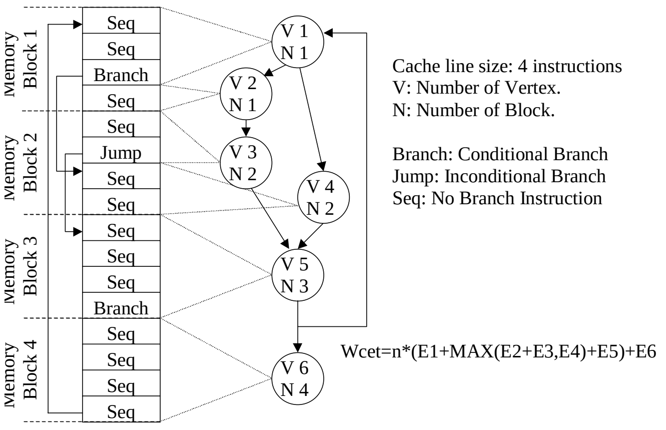

Fig. 1. Exemple of cache-control flow graph and WCET caluclation.

IV. Cache contents selection

Randomly locking instruction offers predictability but not guarantee good response time of tasks. Instructions to be loaded must be carefully selected, looking for the best scenario. This scenario is a set of memory blocks that provides the shortest execution times, thus providing the best response time of tasks to obtain schedulability.

Direct selection of instruction is not possible because several factors are involved in the execution time in multitask, preemptive system, including interactions between tasks.

Exhaustive search, including branch and bound, presents an intractable computational cost, since the number of possible solutions is very large.

Genetic algorithms [10], performing a randomly-directed search, are appropriate to solve the problem. The developed algorithm returns the list of memory blocks to be loaded and locked in cache, providing a sub-optimal solution. The algorithm models a possible solution (a set of main-memory blocks locked in cache) called individual. A set of individuals is evaluated, calculating the system average response time using the CRTA previously presented. The best individuals are selected, combining them in new individuals. The process is repeated a given number of times.

The algorithm gives both the list of blocks to load and lock, and the schedulability analysis of the system.

V. Experimental results

Experiments are carried out using the SPIM simulator to simulate the execution of several task sets using seven cache sizes (from 1KB to 64KB) and five map-function (direct, 2-way, 4 way, full associative and locking cache). Moreover, the genetic algorithm estimates the response time of tasks and produces the list of blocks (which is used for simulation). For the experiments, T_{miss} is 10 units and T_{hit} is 1 unit.

Fig. 2 shows the variation of the response time either estimated (by analysis) or simulated using a locking cache. Each bar means the number of experiments that exhibits the variation printed in the x-axis.

Fig. 3 shows the variation of the mean response time either using a standard cache or a locking one. The figure shows the distribution of the variation depending on the cache size. The lines represent the mean and median.

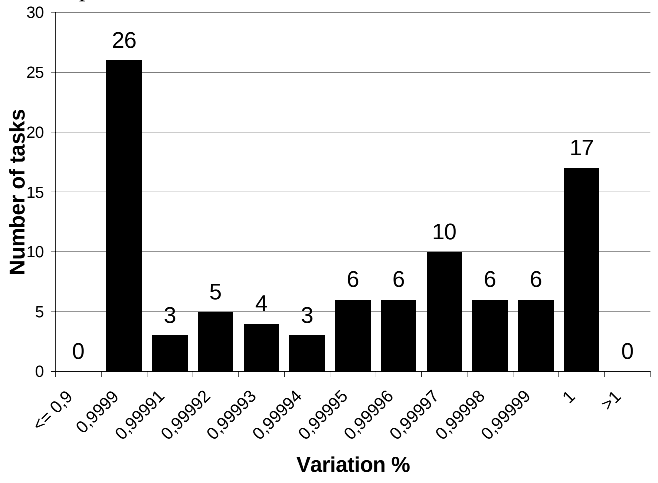

Fig. 2. Number of experiments grouped by the variation between estimated versus simulated response time.

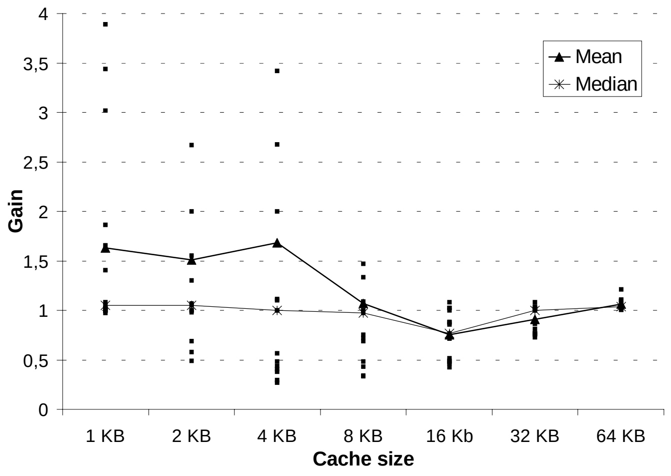

Fig. 3. Variation between the best mean response time using standard cache versus the estimated response time using locking. It is shown the median and mean.

VI. Conclusions

This work presents a novel technique to exploit the use of locking caches in preemptive real-time systems. The hardware required is nowadays present in some off-the-shelf processors, and only minor modifications to the tools are needed to obtain good results.

The main advantage of the technique is that the cache gets predictability, allowing an easy and fast schedulability analysis by well-known algorithms.

Experimental results show that the estimated response time is a very tight upperbound of actual system response time. Also, in some cases, determinism is reached without loss of performance.

REFERENCES

[1] F. Mueller and J. Wegener. "A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints." Proceedings of Fourth IEEE Real-Time Technology and Applications Symposium, Denver, Colorado, USA June 3-5, 1998.

[2] Y. S. Li, S. Malik, and A. Wolfe. "Cache Modeling for Real-Time Software: Beyond Direct Mapped Instruction Caches." Proceedings of the Seventeenth IEEE Real-Time Systems Symposium, December 1996.

[3] S. S. Lim, Y. H. Bae, G. T. Jang, B. D. Rhee, S. L. Min, C. Y. Park, H. Shin, K. Park, and C. S. Kim. "An Accurate Worst Case Timing Analysis Technique for RISC Processors." Proceedings of the Fifteenth IEEE Real-Time Systems Symposium,pp.97-108, December 1994.

[4] S. Basumallick and K. D. Nilsen. "Cache Issues in Real-Time Systems." ACM SIGPLAN Workshop on Language, Compiler, and Tool Support for Real-Time Systems, June 1994.

[5] C. Lee, J. Hahn, Y. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, C. S. Kim. "Enhaced Analysis of Cache-related Preemption Delay in Fixed-Priority Preemptive Scheduling." Proceedings of the 18th IEEE Real-Time Systems Symposium, December 1997.

[6] D. B. Kirk. "SMART (Strategic Memory Allocation for Real-Time) Cache Design." Proceedings of the 10th IEEE Real-Time Systems Symposium, pages 229-237, December 1989.

[7] J.V. Busquets-Mataix, J.J. Serrano, A.J. Wellings. "Hybrid Instruction Cache Partitioning for Preemptive Real-Time Systems." 9th Euromicro Workshop on Real-Time Systems, 271-276, Toledo, Spain, June 1997.

[8] J. V. Busquets-Mataix, A. J. Wellings, J.J. Serrano, R. Ors and P. Gil. "Adding Instruction Cache Effect to an Exact Schedulability Analysis of Preemptive Real-Time Systems." 8th Euromicro Workshop on Real-Time Systems, pages 8-15, L'Aquila, Italy, June 1996.

[9] A. Martí, X. Molero, A. Perles, F. Rodriguéz, J.V. Busquets. "Combined Intrinsic-Extrinsic Cache Analysis for Preemptive Real-Time Systems." Real-Time Programming 2000. 25th IFAC Workshop on Real Time Programming. Ed. Pergamon, 2000

[10] A. Martí, A. Pérez, A. Perles, J.V. Busquets. "Using Genetics Algorithms in Content Selection for Locking- Caches." IAESTED International Conference on Applied Informatics. Acta Press. Innsbruck, Austria, 2001