Measuring Cache Related Pre-emption Delay on a Multiprocessor Real-Time System

Filip Sebek {fsk@mdh.se}
Dept. of Computer Engineering, Mälardalen University, Västerås, Sweden

Abstract

Cache memories in real-time systems can increase performance, but to the cost of unpredictable behavior which gives loose bounds on the worst case execution time analysis. Task pre-emption cause a swap of cache contents with an initial performance dip that is considered as a delay. This delay is necessary in execution time analysis and must be added to each task-switch to determine if the task sets are schedulable.

Cache performance and costs have traditionally been estimated through trace-driven simulations, but since representative traces and a true simulation models are hard to accomplish, a "physical" measurement at the system might be the only way to get the status of the system.

This paper suggests two methods to measure the cache related pre-emption delay on a Power PC750 multiprocessor system by using the processors' built-in performance monitor. One method is completely hardware-based and the other has a minimal software support. Both methods pass information to an external monitor system that stores data with timestamps in a database for further analysis.

1 Introduction and motivation

Cache memories are today common in computer systems to bridge the response time from primary memory to boost up performance. Since the small cache is too small to hold all data and instructions, blocks are swapped in and out depending of what section of code in the program that is handled at the moment. The swapping results to a variable memory access time, which make execution time analysis in real-time systems very tricky to analyze.

There are two categories of cache block swap-outs [1];

- Intrinsic (inter-task) behavior depends on the internal design and execution path of the task. Two functions or data areas in the task may compete for the same cache space with cache misses and a performance loss as a result. Increasing the cache memory size or associativity can reduce these effects.

- Extrinsic (intra-task) behavior depends of the environment and the others tasks' inter-task behavior. At context-switch the cache contents will be more or less displaced by the new running task. This performance loss is also called *cache related pre-emption delay* (CRPD). To eliminate the CRPD and extrinsic cache effects, Kirk suggests to partition the cache into segments and assign task their own segment of the cache [2]. This can also be accomplished in software by locating code and data so they won't map and compete for the same areas in the cache [3].

Both these problems must be solved or correctly analyzed to be able to give an accurate and tight *Worst Case Execution Time* (WCET). The WCET is the base to all scheduling of a task set — without it one cannot determine if the task set is schedulable or not in a real-time system.

Basumallick and Nilsen identifies the CRPD in the Real-Time environment in [4] with the formula $C' = C + 2\delta + \gamma$, where C' is the new WCET, C stands for the unmodified WCET, δ is the execution time for the operating system to make a context-switch (two are needed for a pre-emption) and γ symbolize the maximum cache related cost by a pre-emption.

Even though many different analysis methods and recent research has been able to bound WCET tighter with many different solutions, they are still not applicable in industrial systems due to their limitations. See for instance [5, 6, 7, 8]. A safe approach is to assume that the complete cache must be reloaded on a context-switch, but since this shouldn't be true in for instance a system with small but many tasks, the schedule would lead to an underutilization of CPU resources. Industrial developers are today in a great need of real values to implement new software based products on high-performance processors.

To the best of our knowledge the only work that has been presented to measure the CRPD is Mogul and Borg's trace driven simulation of a UNIX-system [9]. Mogul and Borg measured the delay (γ) to $200 - 400\mu s$ of a task. The traces were however not taken from a real-time system and all the time-slices were of equal size. The cache memories are today larger and more complex than those the simulations were performed at.

Performance estimation on cache memories has traditionally been made with trace driven simulation. The simulations are mainly made at single programs or tasks and with absence of operating system, pipelining, complex cache structures, prefetching features etc. This paper presents methods how to measure the CRPD at a real running multi-processor system.

2 The multi-processor system

SARA — Scaleable Architecture for Real-Time Applications — is a research project with a Motorola Compact PCI backplane bus with Power PC750-processor boards [10]. See figure 1.

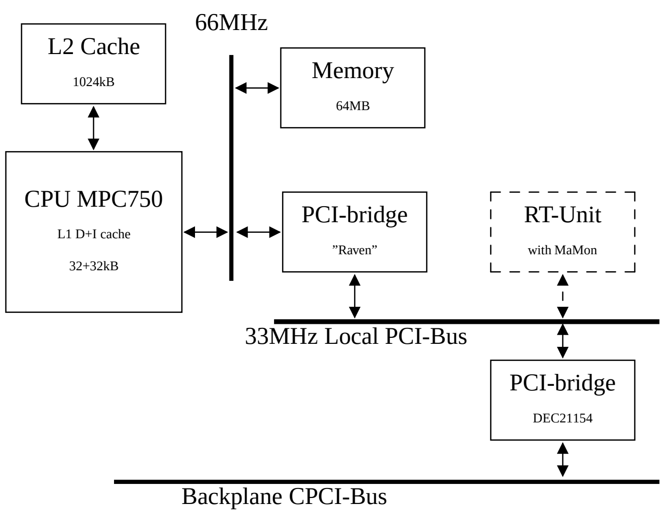

Figure 1: CPU-card. The RT-Unit is only on master cards.

A special *master card* is equipped with a Real Time Unit (RTU)[11] that controls the execution of the tasks on all processor cards. The RTU is a high performance and performance predictable hardware implementation of an operating system that handles scheduling and other real-time operating system services. No other software is needed. The other processor cards are used as *slaves* to increase application performance. All communication between tasks (inter and intra-processor) is performed through a *virtual bus* which simplifies application development[12]. A special device called *Multipurpose Application Monitor (MAMon)* [13] is connected to the RTU.

Today MAMon and the RTU co-exists in the same FPGA, and besides increased performance it is a very practical and cost effective way to eliminate problems with PCB-layout and other hardware manufacturing issues.

3 The measurement

The Motorola PowerPC 750-processor (MPC750) is equipped with a performance monitor [14, 15] with four dedicated registers which count predefined events such as cache misses, miss predicted branches, number of fetched instructions, and other occurrences. The monitor function is meant to be used to tune performance on software applications and to help system developers to debug their systems.

To measure the CRPD time directly is not possible since the MPC750 performance monitor is not that advanced. The performance monitor at the processor is set to count cache misses and instruction fetches, which is the information needed to calculate instruction miss ratio. To calculate the data miss ratio, the data misses, and the number of loads and stores must be counted. By continuously measuring the miss ratio, the CRPD can be calculated and presented.

One problem is to distinguish the extrinsic misses from the intrinsic, which is by the suggested measure model impossible to do exactly since the performance monitor is unable to recognize types of the misses. Our approach is to determine the *average miss-ratio-level* of the task and subtract it from the miss-ratio after the context-switch. Figure 2 illustrates a scenario with a context-switch from a task with 22 percent average miss-ratio to another task with 12 percent.

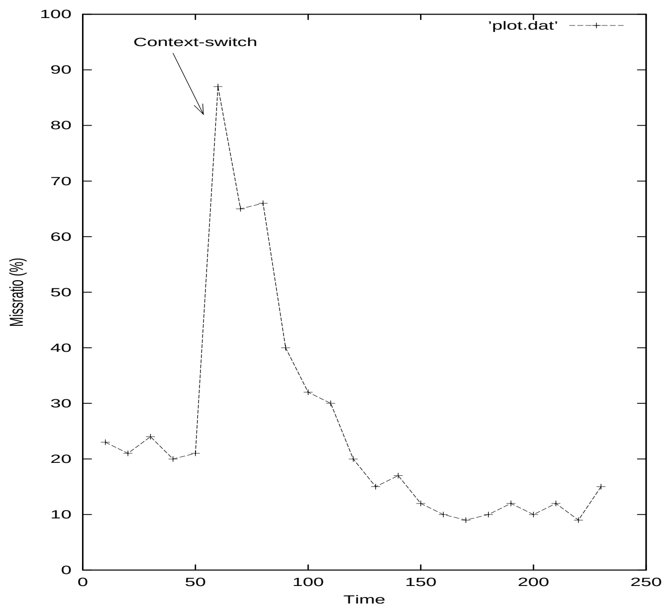

Figure 2: Miss-ratio during a context-switch.

3.1 With software support

A small, simple, cyclic task — "MonPoll" — polls the MPC750's performance monitor registers and pass them to MAMon where they get time stamped. An alternative solution is to store the data in the task's local memory, but the writing to memory would compete with the application's need for cache memory with a serious swap of cache data as a result. A third possibility is to stream the data through an Ethernet card as UDP-messages to a remote host on the network, but these actions leads to substantially longer execution and a swap of contents in the instruction cache.

MAMon is non-intrusive and provide the service needed with a minimum of software code. Only one simple C-code assignment is all that is needed to store a value in a MAMon register. MAMon sends (through a parallel port) timestamps, register values and other predefined task information to a database on an external host where it is stored for further analysis and graphical presentation. See figure 3.

Even if it is smallest possible, the MonPoll-task will however interfere and pollute the cache result by its own presence in the executing environment. It will also use system resources and decrease performance by increasing the execution load. The cache content could be unaffected by turning off the cache memories during the polling, but that would on the other hand have an influence on the execution time.

Leaving the polling task as is in the running system can "solve" the problem. To make a system safe and act as it did under the test phase during its development, probes must

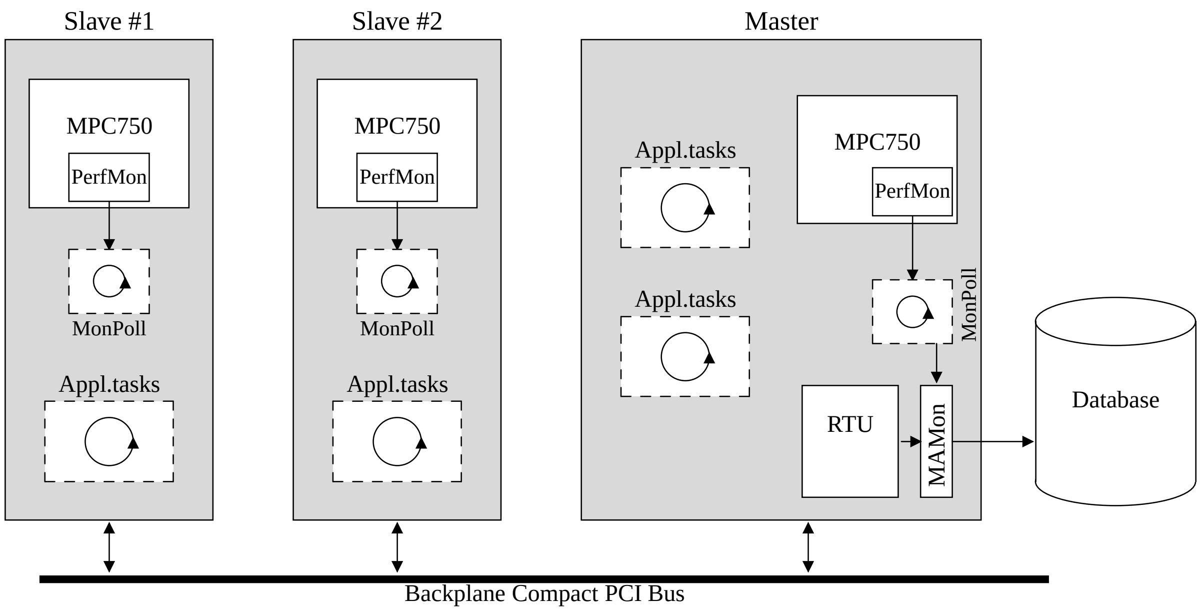

Figure 3: SARA system. Dashed boxes are software implementations.

never be removed [16]. The cost of an observable system is to dedicate some percentage of the capacity to software probes. The less frequent probing is performed, the less interference, but with a loss of valuable data. If the performance loss is unacceptable in a running system, the ”MonPoll”-task must be removed and to get a more correct performance value, measurements at different sample frequency must be performed. By extrapolating these values more correct performance estimations will be at hand. This method is applicable on soft real-time systems due to its more or less polluting nature.

3.2 Completely in hardware

The register values from the performance monitor can be read through special pins on the processor called JTAG-pins (defined by IEEE 1149). To interpret those signals on an oscilloscope by ”hand” is a non-trivial task due to the large amount of information presented in just a wave-diagram. To cope with this translation and interpretation problem a special hardware device communicates with the performance monitor through JTAG and ”download” the requested data. This device could then pass the information further to MAMon through either dedicated peer-to-peer links or a special bus with MAMon and those devices attached to it. The device acts exactly as the ”MonPoll” task described in the previous section but in this case it is implemented in hardware. This method might be suitable for hard real-time applications since it doesn’t interfere the caches or the execution time.

Many hardware constructions can however not run at full speed when JTAG is used which is the fact for some CPUs in the Motorola PPC-family. The positive part of this approach is its non-polluting and non-interfering measuring, but the drawbacks are, as mentioned, an underutilization of CPU performance and hardware costs.

4 A synthetic workbench

No good standard benchmark suits are available today to measure cache memory effects in real-time systems. Non-real-time benchmarks such as SPEC or Dhrystone are just single programs without (interfering) tasks. Rhealstone[17] on the other hand just tests real-time operating system issues such as task-switches, deadlock handling and task communication. The test applications are to small to test cache memory issues and were not meant to do so either.

The test will therefore be performed on synthetically generated task sets where the amount of tasks and the data and instruction size of all the tasks will be generated by a special program. Task interaction, priorities, and cycle time can also be set. By altering the mentioned parameters and measure the hit ratio over time, a pretty good view of how the CRPD will be available. The method has successfully been used in for instance Busquets-Mataix *et al*’s work[18].

5 Conclusions and Future work

Performance estimation on cache memories has traditionally been made with trace driven simulation with only pieces of a complete trace and simplified simulators. To be able to make correct and tight worst-case execution time analysis all types of delays and interference in the execution must be identified. One such property is the Cache Related Pre-emption Delay (CRPD), which is a product of extrinsic cache behavior.

This paper suggests two methods to measure hit-ratio by using the processors built-in performance monitor. Either a software task or a hardware device can poll the data from the performance monitor. The software solution is performed with a minimum of code to reduce a probe effect. The polled information is passed further to a system monitor that also collects and stores context-switch information and other predefined events in a database. Since all events are time stamped miss ratio can be continuously monitored and the CRPD can be calculated. The methods will give applicable values to execution time analysis since they measure real time on a real running system. A complete hardware solution might seem to be a better solution since it could be intrusive free, but the data is only available when the processor is running at half speed if JTAG is to be used.

Standard benchmarks don't cause pre-emption and that is why CRPD cannot be measured at those. Generated task sets with synthetic tasks of different numbers, sizes, relations, cycle time etc. will generate pre-emptions to be measured and will also give a figure how exact the suggested measurement method is.

Due to heavy performance loss and costly redesign of PCB-layout, the hardware device solution is excluded and only the software solution is motivated to implement in the future work.

References

[1] Anant Agarwal, Mark Horowitz, and John Hennessy. An analytical cache model. *ACM Theory of Computing Systems*, 7(2):184–215, May 1989.

[2] David B. Kirk. SMART (strategic memory allocation for real-time) cache design. In IEEE Computer Society Press, editor, *Proceedings of the Real-Time Systems Symposium - 1989*, pages 229–239, Santa Monica, California, USA, December 1989. IEEE Computer Society Press.

[3] Andrew Wolfe. Software-based cache partitioning for real time applications. In *Proceedings of the Third International workshop on Responsive Computer Systems*, September 1993.

[4] Swagato Basumalik and Kelvin D. Nilsen. Cache issues in real-time systems. In *Proceedings of the ACM SIGPLAN Workshop on Language, Compiler, and Tool Support for Real-Time Systems*, June 1994.

[5] Hiroyuki Tomiyama and Nikil Dutt. Program path analysis to bound cache-related preemption delay in preemptive real-time systems. In *Proceedings of 8th International Workshop on Hardware/Software Codesign (CODES2000)*, pages pp. 67–71, May 2000.

[6] Yau-Tsun Steven Li, Sharad Malik, and Andrew Wolfe. Performance estimation of embedded software with instruction cache modeling. *ACM Transactions on Design Automation of Electronic Systems*, 4(3):257–279, July 1999.

[7] C. Healy, R. Arnold, F. Mueller, D. Whalley, and M. Harmon. Bounding pipeline and instruction cache performance. *IEEE Transactions on Computers*, 48(1):53–70, January 1999.

[8] Henrik Theiling, Christian Ferdinand, and Reinhard Wilhelm. Fast and Precise WCET Prediction by Seperate Cache and Path Analyses. *Real-Time Systems*, 18(2/3), May 2000.

[9] Jeffrey C. Mogul and Anita Borg. The effect of context switches on cache performance. In *Proceedings of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 75–84, Santa Clara, CA, USA, April 1991.

[10] Lennart Lindh, Tommy Klevin, and Johan Furunäs. Scaleable architecture for real-time applications – SARA. In *Proceedings of SNART 1999*, Linköping, Sweden, 1999.

[11] Johan Furunäs, Johan Stärner, Lennart Lindh, and Joakim Adomat. RTU94 – real time unit 1994 – reference manual. Technical report, Dept. of computer engineering, Mälardalen University, Västerås, Sweden, January 1995.

[12] Peter Nygren and Lennart Lindh. Virtual communication bus with hardware and software tasks in real-time system. In *Proceedings for the work in progress and industrial experience sessions at 12th Euromicro conferance on Real-time systems*, June 2000.

[13] Mohammed El Shobaki. Non-intrusive hardware/software monitoring for single- and multiprocessor real-time systems. Technical report, Mälardalen Real-Time Research Centre, Västerås, Sweden, April 2001.

[14] Motorola Corp. *MPC750 RISC Microprocessor Users Manual*, August 1997.

[15] Motorola Corp. *Errata to MPC750 RISC Microprocessor Users Manual*, July 1999.

[16] Henrik Thane. *Monitoring, Testing and Debugging of Distributed Real-Time Systems*. Doctorial thesis, Royal Institute of Technology, KTH and Mälardalen University, Stockholm and Västerås, Sweden, May 2000.

[17] Rabindra P. Kar. Implementing the rhealstone real-time benchmark. *Dr. Dobb's Journal*, pages 46–55 and 100–104, April 1990.

[18] J. V. Busquets-Mataix, A. Wellings, J. J. Serrano, R. Ors, and P. Gil. Adding instruction cache effect to schedulability analysis of preemptive real-time systems. In *IEEE Real-Time Technology and Applications Symposium (RTAS '96)*, pages 204–213, Washington - Brussels - Tokyo, June 1996. IEEE Computer Society Press.