Static Probabilistic Timing Analysis for Multicore Processors with Shared Cache

Robert I. Davis, Jack Whitham,
Computer Science Department, University of York, UK
rob.davis@york.ac.uk, jack.whitham@york.ac.uk

Dorin Maxim
INRIA Nancy-Grand Est, France
dorin.maxim@inria.fr

I. Introduction

As a result of stringent requirements on size, weight, and power consumption (SWaP), as well as the need to provide advanced new functionality through software, critical real-time embedded systems in the aerospace, space, and automotive markets are beginning to make use of multicore processors for hard real-time applications. The deployment of hard real-time applications on advanced multicore processors requires a number of challenges to be overcome. A wealth of research has been done on scheduling and schedulability analysis for multicore platforms (see [7] for a survey); however, this work needs to be underpinned by safe and tight estimates of the Worst-Case Execution Time (WCET) of component tasks. Considerable work has been done on WCET analysis assuming non-pre-emptive execution on single processors (see [13] for a survey), and the impact of Cache Related Pre-emption Delays (CRPD) has been taken into account for multitasking systems [1], [2], [11]. However, research into adapting WCET techniques to address the challenges of multicore processors, and specifically the use of caches that are shared between two or more processors, is in its infancy.

Many multicore processors make use of a cache hierarchy with private L1 caches per core and an L2 cache shared between cores. Sharing the L2 cache provides a significant performance boost over private L2 caches of the same total size, improving average-case performance and energy efficiency; however, the inter-core cache interference makes the WCET analysis problem extremely challenging. The problem is exacerbated by timing anomalies: The worst-case path for code executed in isolation may no longer be the worst-case path when inter-core cache contention is accounted for – see Figure 2 in [14].

The simplest and, as far as we are aware, the only compositional solution available so far is to assume that all L2 accesses are cache misses; however, as L2 latencies are typically high this is extremely pessimistic. Initial papers in this area [10], [14], [15], and [16] provide analysis which has complex dependencies on the detailed execution behaviour of contending tasks on other cores. However, details of the contending tasks may not necessarily be available for analysis, and even if they are, then such cross dependencies go against requirements for composability, which are necessary for the efficient development and integration of complex systems.

II. Probabilistic Approach

One possible solution is to make use of randomisation techniques, in particular random cache replacement policies, which make the probability of pathological cases vanishingly small. Static Probabilistic Timing Analysis (SPTA) has been developed for single processor systems assuming both evict-on-access [4] and evict-on-miss policies [8], with analysis of probabilistic CRPDs also given in [8]. Such approaches have the potential to provide an increase in the level of performance of hard real-time systems that can be guaranteed with respect to an acceptable threshold for the timing failure rate [3], particularly as the bounds provided degrade far less rapidly, with respect to incomplete information about the execution history, than the WCET computed assuming deterministic cache replacement policies [12].

It is our conjecture that in the multicore case, with shared caches, random cache replacement policies have the potential to break dependencies on execution history and specific inter-core cache contention, and so permit effective probabilistic WCET analysis that supports timing composition. This is particularly important in the case of mixed-criticality systems where the tasks running on other processors may be of lower criticality. Separation via a simple interface, independent of task behaviour or misbehaviour, ensures that lower criticality tasks have a strictly bounded effect on higher criticality tasks of interest through interactions via the shared cache. A similar argument applies to open systems, where non-real-time tasks may be downloaded and run on a specific core. Here the complete set of tasks is not available at design time for analysis.

A. Single processor SPTA

We now recap on SPTA for single processor systems with an evict-on-miss random cache replacement policy for the instruction cache and no data cache. With the evict-on-miss policy, whenever an instruction is requested and is not found in the cache, then a randomly chosen cache line is evicted and the memory block containing the instruction is loaded into the evicted location. We assume an N-way associative cache, and hence the probability of any cache line being evicted on a miss is $1/N$.

For simplicity, we assume a single path program (extensions to multipath programs are given in [8]) comprising a fixed sequence of instructions. We represent these instructions via the memory blocks they access, with a superscript indicating the *re-use distance k.* (The re-use distance is the maximum number of evictions since the last access to the memory block containing that instruction). For example, $a, b, a^1, c, d, b^3, c^2, d^2, a^5, e, b^4, f, e^2, g, a^5, b^4, h$. For each instruction, the probability of a cache hit is lower bounded by:

$$P_{hit}(k) = \left(\frac{N-1}{N}\right)^k \tag{1}$$

Provided that $k < N$, otherwise $P^{hit}(k) = 0$, (details of this latter restriction are given in [8]). Given fixed costs for the cache-hit latency (e.g. $H = 1$) and the cache miss latency (e.g. $M = 10$), then an upper bound pWCET distribution of a program can be computed as the convolution ($\otimes$) of the probability mass functions (PMFs) of each instruction. For example, given two instructions with PMFs with cache hit probabilities of 0.8 and 0.7 respectively, we get a pWCET distribution for the 'program' that has a probability of the execution time being 2 on any given run of 0.56, a probability that it will be 11 of 0.38, and a probability that there will be two cache misses and hence an execution time of 20 of 0.06.

$$\begin{pmatrix} 1 & 10 \\ 0.8 & 0.2 \end{pmatrix} \otimes \begin{pmatrix} 1 & 10 \\ 0.7 & 0.3 \end{pmatrix} = \begin{pmatrix} 2 & 11 & 20 \\ 0.56 & 0.38 & 0.06 \end{pmatrix} \quad (2)$$

We note that for larger numbers of instructions, the probability of a large number of cache misses quickly becomes vanishingly small, as illustrated by the graphs of 1-CDFs or exceedance functions shown in Figure 1 and Figure 2 below.

B. Open problem: Multicore SPTA with shared cache

In the multiprocessor case, we would like to develop SPTA that derives the probabilities of cache hits / misses and hence the pWCET distribution for a sequence of instructions (i.e. a task) running on one core, given an instruction cache with an evict-on-miss random replacement policy that is shared with one or more other cores which also execute tasks that can cause evictions. (For clarity and simplicity in our initial investigation we consider only instruction cache and do not consider a cache hierarchy). Clearly there is a mutual dependency between tasks; however, we aim to use simple interface models to restrict the information required to analyse each task and hence support a compositional approach to system development.

For example:

(i) A simple interface might state that the tasks running on each contending core can cause at most $\lceil t/M \rceil$ evictions in a time interval of length t, where M is the cache miss latency. This interface description holds irrespective of the details of the tasks and hence supports composition.

(ii) An upper bound may also be derived on the number of evictions caused in a time interval by all cores, reflecting the limited bandwidth of the memory bus. Due to limited bandwidth, there can be at most $\lceil t/b \rceil$ evictions in a time interval of length t, where b is the time required to transfer one cache block.

(iii) More complex analysis of the number of bus requests may be possible. Thus more complex interfaces might provide sub-additive functions giving the maximum number of evictions that can possibly occur due to all cores as a function of the length of the time interval. We note; however, that modelling the maximum number of bus requests from tasks on other cores as done in [5], [6], while providing more precise analysis, goes against the requirements for composition. Related work could look at how hardware might potentially police a maximum rate of evictions due to tasks on a particular core.

To fully develop the theory in this area, a detailed model of the behaviour of the hardware is needed to correctly and accurately model the worst-case number of evictions that can occur due to contention from other cores in a time interval of length t, as well as to determine the maximum amount of time required for a sequence of instructions to execute on one core given some number of cache misses.

A first attempt at solving our open problem integrates simple interface models of additional evictions into existing SPTA.

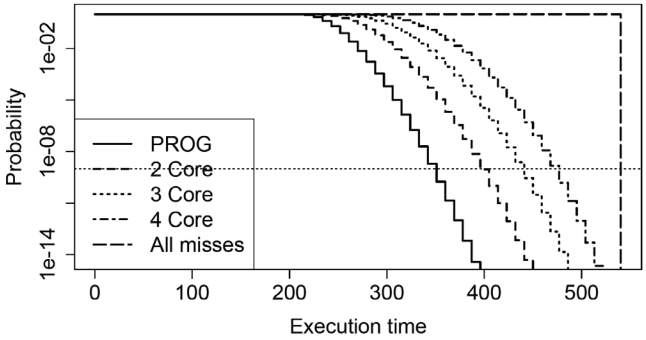

Figure 1: pWCET distributions (1-CDF) for FAC

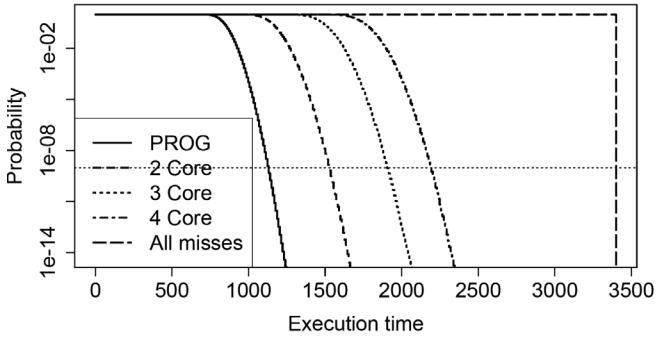

Figure 2: pWCET distributions (1-CDF) for FIBCALL

As an example, we model the contention from other cores as increasing the maximum number of evictions between instructions and hence their re-use distances. With m cores, the re-use distance k of each instruction (assuming execution in isolation) could potentially be increased by contention to $(m-1)+mk$. Figure 1 and Figure 2 show the effect that this increase in re-use distances has on the pWCET distributions for the FAC and FIBCALL programs from the Malardalen

benchmark suite [9]. PROG represents the program run in isolation, while the lines on the graphs for 2 Core, 3 Core, and 4 Core show the effect on the pWCET distribution due to contention from other cores. We observe that the increase in execution time at a given probability (shown for 10^{-9}) though significant, is much less than would have to be assumed for a deterministic system. In the deterministic case, with no knowledge of which cache blocks programs on other cores are using, the only safe assumption is the 'all misses' scenario shown as the final line in the figures. (Note, the results are for an evict-on-miss random cache replacement policy, a cache block size of 1 instruction, and a cache size of N=128 blocks).

The provision of WCETs and pWCETs form the basis for schedulability analysis. The context for the problem presented is one of higher level task allocation and scheduling. Assuming, for example, partitioned scheduling, pWCETs are needed as part of the schedulability analysis for each processor, and hence as part of the task allocation algorithm. These pWCETs could potentially vary depending on the detailed task allocation, indicating the need for an integrated approach. However, with simpler models fully supporting composition, then it would be sufficient to obtain pWCETs that account for the maximum impact of contention. Such pWCETs distributions would be independent of the task allocation and could be used directly in a separate task allocation and schedulability analysis algorithm, i.e. without integration. Further, with the development of a SPTA for shared caches, a comparison can be made to see if better pWCET estimates are obtained with a shared cache or with partitioning of the cache to individual cores.

ACKNOWLEDGEMENTS

The authors would like to thank Luca Santinelli from ONERA for his help with the example figures.

REFERENCES

[1] S. Altmeyer, R.I. Davis, C. Maiza "Cache related Pre-emption Delay aware response time analysis for fixed priority pre-emptive systems". In proceedings Real-Time Systems Symposium, pp. 261-271, 2011.

[2] S. Altmeyer, R.I. Davis, C. Maiza "Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems" . Real-Time Systems, Volume 48, Issue 5, Pages 499-526, Sept 2012.

[3] F. Cazorla, E. Quinones, T. Vardanega, L. Cucu, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim. PROARTIS: Probabilistically analysable real time systems. *ACM Transactions on Embedded Computing Systems (to appear)*, 2013.

[4] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quinones, and F. J. Cazorla. Measurement-Based Probabilistic Timing Analysis for Multi-path Programs. In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), 2012.

[5] D. Dasari, B. Andersson, V. Nelis, S.M. Petters, A. Easwaran, L. Jinkyu, "Response Time Analysis of COTS-Based Multicores Considering the Contention on the Shared Memory Bus," In Proceedings 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp.1068,1075, 16-18 Nov. 2011.

[6] D. Dasari, V. Nelis, "An Analysis of the Impact of Bus Contention on the WCET in Multicores," In Proceedings High Performance Computing and Communication & International Conference on Embedded Software and Systems (HPCC-ICESS), pp.1450,1457, 25-27 June 2012.

[7] R.I. Davis, A. Burns, "A Survey of Hard Real-Time Scheduling for Multiprocessor Systems", ACM Computing Surveys, 43, 4, Article 35 (October 2011), 44 pages. DOI 10.1145/1978802.1978814.

[8] R. I. Davis, L. Santinelli, S. Altmeyer, C. Maiza, and L. Cucu-Grosjean. "Analysis of probabilistic cache related pre-emption delays". In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS) 2013. (Techical report available from http://www-users.cs.york.ac.uk/ robdavis/).

[9] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper. The Malardalen WCET benchmarks – past, present and future. In B. Lisper, editor, *WCET 2010*, pages 137–147, Brussels, Belgium, July 2010. OCG.

[10] Y. Li, V. Suhendra, Y. Liang, T. Mitra, and A. Roychoudhury,"Timing analysis of concurrent programs running on shared cache multi-cores," Proceedings of the Real-Time Systems Symposium (RTSS), 2009.

[11] W. Lunniss, S. Altmeyer, C. Maiza and R.I. Davis, "Integrating Cache Related Pre-emption Delay Analysis into EDF Scheduling" In proceedings Real-Time and Embedded Technology and Applications Symposium 2013.

[12] E. Quinones, E. Berger, G. Bernat, and F. Cazorla. "Using Randomized Caches in Probabilistic Real-Time Systems". In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), pages 129–138, 2009.

[13] R.Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckman, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, P. Stenstrom. TheWorst case execution time problem - overview of methods and survey of tools. In ACM Transactions on Embedded Computing Systems, January 2007.

[14] J. Yan and W. Zhang, "WCET analysis for multi-core processors with shared L2 instruction caches," Proceedings of the Real-Time and Embedded Technology and Applications Symposium (RTAS), 2008.

[15] W. Zhang, J. Yan "Static Timing Analysis of Shared Caches for Multicore Processors", Journal of Computing Science and Engineering, Vol. 6, No. 4, December 2012, pp. 267-278

[16] W. Zhang and J. Yan, "Accurately estimating worst-case execution time for multi-core processors with shared directmapped instruction caches," Proceedings of the 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, (RTCSA) 2009.