CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

Mixed-Criticality Systems with Partial Lockdown and Cache Reclamation Upon Mode Change

Konstantinos Bletsas
Muhammad Ali Awan
Pedro Souto
Benny Åkesson
Eduardo Tovar

CISTER-TR-170507

2017/06/27

Mixed-Criticality Systems with Partial Lockdown and Cache Reclamation Upon Mode Change

Konstantinos Bletsas, Muhammad Ali Awan, Pedro Souto, Benny Åkesson, Eduardo Tovar

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: ksbs@isep.ipp.pt, muaan@isep.ipp.pt, pfs@fe.up.pt, kbake@isep.ipp.pt, emt@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

In mixed-criticality multicore systems, the appropriate degree of isolation between applications of different criticalities is a primary objective. However, efficient utilization of the platform 19s processing capacity and other resources is still desirable and important. In recent work, we, therefore, proposed an approach that reclaims cache resources assigned to low-criticality tasks when these are dispensed with, in the event of a system mode change. The reclaimed cache resources are reassigned from the lower-criticality tasks to the remaining higher-criticality tasks to improve performance. The per-task cache partitions can either be configured to hold frequently accessed (1chot 1d) pages, locked in place, or they can be used dynamically, with cache lines moved in and out. The first option simplifies WCET analysis while the second option simplifies the act of cache reconfiguration at runtime.Meanwhile, the performance implications of the two options are not immediately obvious. Therefore, in this work-in-progress, we explore an arrangement that combines both approaches, in order to achieve the best tradeoff between efficient analysis, low reconfiguration overheads and good schedulability Simple per task cache partitions (without page locking) are to be used for the portion of the cache that is subject to reclamation. At mode switch, the high-criticality tasks keep the pages they had locked in the cache and get additional partitions, out of reclaimed cache, to bring other pages in and out as needed.

© CISTER Research Center
www.cister.isep.ipp.pt

Mixed-Criticality Systems with Partial Lockdown and Cache Reclamation Upon Mode Change

Konstantinos Bletsas*†, Muhammad Ali Awan*†, Pedro Souto‡*, Benny Akesson*†, Eduardo Tovar*†
*CISTER Research Centre, Porto, Portugal
†ISEP, Polytechnic Institute of Porto, Portugal
{muaan, ksbs, kbake, emt}@isep.ipp.pt
‡ University of Porto, Faculty of Engineering, Portugal
pfs@fe.up.pt

***Abstract*—In mixed-criticality multicore systems, the appropriate degree of isolation between applications of different criticalities is a primary objective. However, efficient utilisation of the platform's processing capacity and other resources is still desirable and important. In recent work, we therefore proposed an approach that reclaims cache resources assigned to low-criticality tasks when these are dispensed with, in the event of a system mode change. The reclaimed cache resources are reassigned from the lower-criticality tasks to the remaining higher-criticality tasks to improve performance. The per-task cache partitions can either be configured to hold frequently accessed ("hot") pages, locked in place, or they can be used dynamically, with cache lines moved in and out. The first option simplifies WCET analysis while the second option simplifies the act of cache reconfiguration at run-time. Meanwhile the performance implications of the two options are not immediately obvious. Therefore, in this work-in-progress, we explore an arrangement that combines both approaches, in order to achieve the best tradeoff between efficient analysis, low reconfiguration overheads and good schedulability Simple per-task cache partitions (without page locking) are to be used for the portion of the cache that is subject to reclamation. At mode switch, the high-criticality tasks keep the pages they had locked in the cache and get additional partitions, out of reclaimed cache, to bring other pages in and out as needed.**

I. Application domain and challenge

Real-time embedded systems in different domains (automotive, avionics, aerospace) host computational tasks of different criticalities. Lower-criticality tasks must not interfere unpredictably with higher-criticality tasks at run-time, because a deadline miss by a high-criticality task can be disastrous. Conversely, rigid prioritisation by criticality and/or using the same development processes and worst-case execution time (WCET) estimation techniques for lower-criticality software as for high-criticality software is inefficient in terms of platform utilisation and engineering cost. Such issues are exacerbated with the move to multicores. Therefore, the research community has focused on assembling a toolset of scheduling models and techniques for (i) efficient use of processing capacity and (ii) schedulability guarantees for all tasks under typical conditions subject to (iii) ensured schedulability of high-criticality tasks in all cases. Most works [1] build on Vestal's model [2], as refined by Baruah and Burns [3]. This versatile model views the system operation as different modes, whereby only tasks of a certain criticality or higher execute; additionally, different WCETs estimates are assumed for the same task in each mode that it can be a part of, with corresponding degrees of confidence. In the simpler case of just two criticality levels (and two modes, L and H), whenever a task overruns its WCET estimate for the L-mode, the system switches to the H-mode, and lower-criticality tasks are dispensed with.

II. Motivation

This line of work is motivated by our view that when designing scheduling arrangements for mixed-criticality systems, (i) the consideration of more detailed architectural models and (ii) the leveraging of their properties can provide both greater confidence in the analysis and improved schedulability. We also noticed, that just as the classic Vestal model dynamically adjusts the allocation of one resource (the processor) whenever a mode change occurs, the same principle could be put to use for other resource types. So, in our latest work [4], we proposed an arrangement whereby the shared last-level cache on a multicore is partitioned among the tasks and, at mode change, the cache resources assigned to the lower-criticality tasks are reallocated to the remaining higher-criticality tasks, when the former ones are discarded. The principle leveraged is that a task's WCET time (and estimates of it) is a function of the cache resources available to the task. However, this principle can be exploited in different ways.

In particular, we assumed that each task will *lock* its most-frequently accessed ("hottest") pages into its partition, using Coloured Lockdown [5]. At mode change, the portion of the cache reclaimed from the lower-criticality tasks would be reused for bringing in and locking in additional hot pages of the high-criticality tasks. This was the "default" arrangement but we also acknowledged the possibility of instead entirely foregoing any page locking in the cache and just using per-task partitions, with their contents (cache lines) dynamically updated at run-time, as lines are brought in and evicted. One way or the other, at mode change, the high-criticality tasks receive more cache.

The main reason for considering the alternative arrangement (without locking) is that of potentially substantial latencies associated with unlocking pages of L-tasks at mode change, bringing in pages by H-tasks in their place and locking them in place.

Refraining from the use of in-cache page locking would largely solve the problem of high cache reconfiguration latencies at mode change, in practice. However there are other analysis and performance implications from choosing this approach.

Cache-aware static WCET analysis of a task using a dedicated cache partition without any locking is computationally more complex and necessitates keeping track of a larger state, compared to the case of when the cache partition holds locked memory pages. To the extent that this necessitates simplifying (pessimistic) assumptions for the sake of tractability, it may add pessimism to the WCET estimation. We expect that, in some cases, such pessimism may result in higher WCET estimates for a task whose cache partition of a given size is used dynamically than if page locking were used – whatever the actual (but, ultimately, unknowable) exact WCETs are. Static WCET analysis techniques are to be used for WCET estimates used in the high-criticality mode.

Conversely, for the low-criticality mode WCET estimates, a measurement-based approach could be used, perhaps in conjunction with statistical analysis and Extreme Value Theory, to estimate excedance probabilities for a given execution time. We conjecture that measurement-based estimates of WCET could be lower, more often than not, when a task's cache partition is used dynamically than in the case of holding locked pages.

These conjectures would need to be experimentally tested, with benchmarks on real hardware but in any case, the mode of use of the cache partition has implications on the WCET estimates derivable for the task. Therefore, we seek to identify how to most appropriately combine the two approaches, for the best performance in terms of mixed-criticality schedulability.

III. Problem Statement

A. The proposed "Partial-Lockdown" arrangement

Consider a mixed-criticality multicore system where the shared last-level cache is partitioned among the tasks. At mode change, the cache portions originally assigned to the L-tasks are redistributed to the remaining H-tasks, as described earlier. In order to keep the latency of this cache reconfiguration low, we therefore mandate, in this work, that

- The cache partitions of the L-tasks be used dynamically in L-mode, to bring lines in an out, as needed.
- Upon being reassigned to the H-tasks, these same cache lines are to be again used dynamically, by the H-tasks.

This avoids the high latencies from unlocking/re-populating/locking anew every line of such a reclaimed partition. However, for the H-tasks, it would still be practical to arrange for some hot pages to be locked in place in their cache partitions at startup, throughout the L-mode, and even after the switch to H-mode.

Figure 1 illustrates the Partial-Lockdown arrangement we propose. In the L-mode, the H-task cache partitions (τ_1 to τ_4), shown in dark blue) may use some (or all) of the available lines to hold some of their hot pages (locked in place); the rest of an H-task's partition may be employed dynamically. As for the L-tasks (τ_5 to τ_9), their entire partitions (shown in light green) are used dynamically. In the figure, the bigger rectangles with the pin symbol in the corner stand for locked pages; the smaller narrower rectangles stand for parts of the cache whose lines are populated and replaced dynamically.

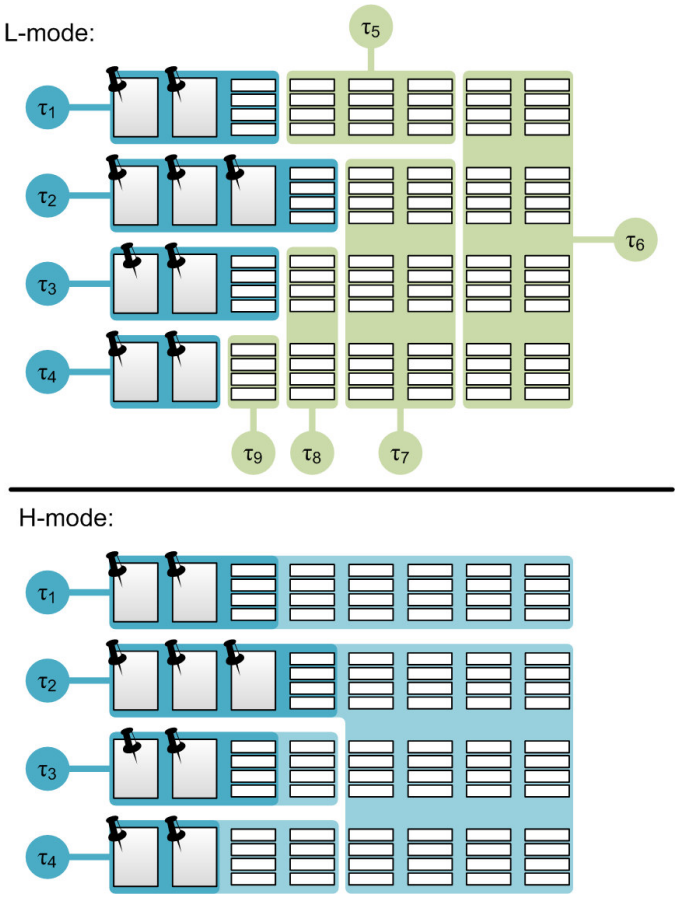

Fig. 1: Illustration of the Partial-Lockdown hybrid cache partitioning arrangement in the two modes.

In the H-mode, the H-task partitions are enlarged with the reclaimed portions (shaded in lighter blue) of the cache that were previously part of the L-tasks' partitions. Those parts of the cache will also be used dynamically by the H-tasks.

It follows that a task's WCET (estimate) is now a function of two variables:

- The number σ of the task's pages (selected in order of hotness, from a predetermined ranking) that are locked in its partition.
- The size π of the part of the task's partition[1] that is available for dynamic use (i.e., with lines populated and evicted at run-time).

This is a generalisation of what held, e.g., in [6], where the "progressive lockdown curve" indicated the WCET estimate as

[1]For symmetry and apples-to-apples comparison, instead of bytes or lines, we also expect the size π to be specified in terms of how many memory pages would fit in there.

a non-increasing function of just one parameter (the number of locked pages). [2]

B. The objective

Assuming partitioned EDF scheduling, with per-task L-mode deadline-scaling, as conceived by Ekberg and Yi [8], how to identify what the appropriate values for σ and π are, for each task, in each of the two modes, in order to have a schedulable system (if at all possible)?

IV. Proposed approach and preliminary results

To solve the problem, we need (i) WCET estimation techniques for creating the parametric 3-dimensional WCET curves, as a function of σ and π, for each task; (ii) appropriate schedulability analysis, to which these WCET estimates will be fed; and (iii) good heuristics for selecting σ and π, for each task and mode.

For parametric H-WCET estimation, we will be considering cache-aware static WCET analysis approaches. Accesses to locked pages will be always-hit upon reuse. Some potentially useful principles for parametric WCET characterisation have been identified in [9]. For the parametric L-WCETs, we could use measurements in conjunction with the application of statistical analysis and Extreme Value Theory, as in our work on WCET estimation for GPU kernels [10].

In terms of schedulability analysis, we can largely piggy-back on the existing analysis from our recent work [4]. It requires (at most) three WCET estimates per task τ_i:

- $C_i^L(\sigma_i, \pi_i^L)$, the WCET estimate of τ_i for the L-mode, assuming σ_i locked pages and size π_i^L for its dynamic cache.
- $C_i^H(\sigma_i, \pi_i^L)$, the WCET estimate of τ_i for the H-mode, assuming σ_i locked pages and size π_i^L for its dynamic cache. This is only defined for H-tasks and pertains to a carry-over job (i.e., released before the mode change but completed after it).
- $C_i^H(\sigma_i, \pi_i^H)$, the WCET estimate of τ_i for the H-mode, assuming σ_i locked pages and size $\pi_i^H \geq \pi_i^L$ for its dynamic cache. This is, again, only defined for H-tasks and characterises jobs released after the mode change.

However, to make it more realistic, we intend to extend that analysis by incorporating the cache reconfiguration latencies, occuring at the mode change. Those latencies would also depend on the variables σ_i, π_i^L and π_i^H that specify the cache partitioning arrangement in the two modes.

As for suitable heuristics for setting σ_i, π_i^L and (where applicable) π_i^H for each task, we do not yet have any strong intuition, but we have some ideas about how to get there.

V. Envisioned solution

We intend to try various approaches and see what works best or learn from them to design an even better heuristic.

[2]The WCET is not always non-increasing with the size of task's dynamically used partition but a non-increasing over-approximation typically entails little loss of precision [7].

Mirroring our approach in [4] we could identify a design metric that correlates with schedulability (such as, for example, the system utilisation) and then optimise for that. For example, we could encode the parametric WCET curves as variables and use an ILP solver to come up with an assignment that minimises the utilisation in the L-mode. Subsequently, we could do the same for the H-mode, subject to the outputs for the L-mode. Given though that the parametric characterisation of the WCET uses two parameters, σ and π, care should be taken to keep this tractable. We expect that this may imply a more coarse-grained partitioning of the cache than in [4].

Alternatively, we could use a meta-heuristic, like simulated annealing or genetic algorithms, to explore the design space.

To the extent that we can identify patterns of good solutions from the output of the ILP or the metaheuristic, we could also come up with our own rules-of-thumb and express them as heuristics. Another thing to consider is to over-approximate the (probably, very irregular) three-dimensional parametric WCET curves with more regular ones, that can be described by relatively simple (e.g., low-order polynomial) equations, in order to leverage their mathematical properties and formulate conceptually clear heuristics.

This work fits into our vision of extending the Vestal model with reclamation of multiple resources types, at mode change.

Acknowledgments

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234); also by by FCT/MEC and the EU ARTEMIS JU within project ARTEMIS/0001/2013 – JU grant nr. 621429 (EMC2).

We thank the reviewers of our ECRTS 2017 paper [4], for their insightful comments.

References

[1] A. Burns and R. Davis, "Mixed criticality systems – a review (9th ed.)," Dept. of Computer Science, University of York, Tech. Rep., Jan. 2017.

[2] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in Proc. RTSS, 2007, pp. 239–243.

[3] S. Baruah and A. Burns, "Implementing mixed criticality systems in Ada," in Proc. 16th Ada-Europe Conf., 2011, pp. 174–188.

[4] M. A. Awan, K. Bletsas, P. F. Souto, B. Åkesson, and E. Tovar, "Mixed-criticality scheduling with dynamic redistribution of shared cache," in Proc. 29th ECRTS, 2017.

[5] R. Mancuso, R. Dudko, E. Betti, M. Cesati, M. Caccamo, and R. Pellizzoni, "Real-time cache management framework for multi-core architectures," in Proc. 19th RTAS, 2013, pp. 45–54.

[6] R. Mancuso, R. Pellizzoni, M. Caccamo, L. Sha, and H. Yun, "WCET(m) estimation in multi-core systems using single core equivalence," in Proc. 27th ECRTS, 2015, pp. 174–183.

[7] S. Altmeyer, R. Douma, W. Lunniss, and R. I. Davis, "On the effectiveness of cache partitioning in hard real-time systems," Real-Time Systems, vol. 52, no. 5, p. 598643, 2016.

[8] P. Ekberg and W. Yi, "Bounding and shaping the demand of mixed-criticality sporadic tasks," in Proc. 24th ECRTS, 2012, pp. 135–144.

[9] J. Reineke and J. Doerfert, "Architecture-parametric timing analysis," in 20th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2014, pp. 189–200.

[10] K. Berezovskyi, F. Guet, L. Santinelli, K. Bletsas, and E. Tovar, "Measurement-based probabilistic timing analysis for graphics processor units," in Proceedings of the 29th International Conference on Architecture of Computing Systems (ARCS), 2016.