CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Conference Paper

Mixed-criticality scheduling with memory regulation

Muhammed Ali Awan
Konstantinos Bletsas
Pedro Souto
Benny Akesson
Eduardo Tovar
Jibran Ali

CISTER-TR-160604

Mixed-criticality scheduling with memory regulation

Muhammed Ali Awan, Konstantinos Bletsas, Pedro Souto, Benny Akesson, Eduardo Tovar, Jibran Ali

CISTER Research Center

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:

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

Abstract

The state-of-the-art models and schedulability analysis for mixed-criticality multicore systems overlook low-levelaspects of the system. To improve their credibility, we therefore incorprate, in this work, the effects of delays from memory contention on a shared bus. Specifically, to that end, we adopt the predictable memory reservation mechanism proposed by the Single Core Equivalence framework. Additionally, we explore how the reclamation, for higher-criticality tasks, of cache resources allocated to lower-criticality tasks, whenever there is a criticality (mode) change in the system, can improve schedulability.

Mixed-criticality scheduling with memory regulation

Muhammed Ali Awan*, Konstantinos Bletsas*, Pedro Souto†*, Benny Akesson*, Eduardo Tovar*, Jibran Ali*
*CISTER/INESC-TEC, ISEP/IPP, Portugal †Faculty of Engineering, University of Porto, Portugal

***Abstract*—The state-of-the-art models and schedulability analysis for mixed-criticality multicore systems overlook low-level aspects of the system. To improve their credibility, we therefore incorprate, in this work, the effects of delays from memory contention on a shared bus. Specifically, to that end, we adopt the predictable memory reservation mechanism proposed by the Single Core Equivalence framework. Additionally, we explore how the reclamation, for higher-criticality tasks, of cache resources allocated to lower-criticality tasks, whenever there is a criticality (mode) change in the system, can improve schedulability.**

I. Introduction

On mixed-criticality scheduling: The integration of functionalities of different criticalities on a multicore necessitates some scheduling isolation between criticality levels. The most notable analytical model for those problems, by Vestal [1], in its basic form, assumes two criticality levels, high and low. High-criticality tasks (H-tasks) have two different estimates of their worst-case execution time (WCET): The L-WCET which is *de facto* deemed safe, and the H-WCET, which is *provably* safe and possibly much greater. For low-criticality tasks, only the L-WCET is defined. There are two modes of operation. The system boots and remains in low-criticality mode (L-mode) as long as no job (instance of a task) executes for longer than its L-WCET. However, if any job exceeds its L-WCET then the system immediately switches into high-criticality mode (H-mode) and drops all L-tasks. It is pessimistically assumed that in H-mode all jobs by H-tasks (including any existing jobs at the time of the mode switch) may execute for up to their H-WCET. Under these assumptions, it must be provable offline that (i) no task misses a deadline in L-mode and (ii) no H-task misses a deadline in H-mode.

Various scheduling approaches for this model exist. Below, we only discuss works employing *deadline-scaling*. This technique originated with the EDF-VD (Earliest Deadline First - with Virtual Deadlines) scheduling algorithm [2] for implicit-deadline mixed-criticality sporadic task sets. EDF-VD uses standard EDF scheduling rules but instead of reporting the real task deadlines to the scheduler, for the purpose of scheduling decisions, it reports shorter deadlines (if needed) for H-tasks during L-mode operation. In doing so, it prioritises H-tasks more than conventional EDF would, over parts of the schedule. This allows H-tasks to be sufficiently ahead of schedule such that they can catch up with their true deadlines if any task overruns its L-WCET. After the switch to H-mode, the true H-task deadlines are used for scheduling and L-tasks are dropped.

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).

EDF-VD proportionately scales all H-task deadlines by a common factor. Ekberg and Yi improve on this by using distinct scaling factors for different H-tasks and a more precise demand bound function (dbf) based schedulability test [3].

Access isolation for caches, memory buses and DRAMs: A prominent approach for making multicores more predictable regarding sources of contention is Single-Core Equivalence (SCE) [4]. Under SCE, fixed-priority scheduling is used and MemGuard [5], a periodic software mechanism, regulates memory accesses from different cores. Over a fixed interval called the *regulation period*, all cores get an equal "slice" of the overall memory bandwidth. This assumes that all memory accesses go through the same memory controller. MemGuard stalls any core that exceeds its share, until the start of the next regulation period. The analysis must consider such *regulation stalls* in addition to conventional *contention stalls*, caused by contention between different cores at the DRAM controller.

SCE's stall-aware schedulability analysis [6], characterises each task by its *WCET in isolation* and its worst-case number of *residual memory accesses*. These correspond to the WCET when no other task is present in the system and an upper bound on the number of memory accesses by the task that go all the way to DRAM. It is also assumed that each task has its most frequently accessed pages locked in place in the shared last-level cache by the Colored Lockdown [7] mechanism. This arrangement promotes determinism by eliminating inter-task interference in the cache and Mancuso et al. use this information to more tightly characterise the WCET in isolation and the number of residual memory accesses. Both quantities decrease as the number of locked pages increases [6]. Using these derived attributes for each task, Mancuso et al. then calculate contention stall and regulation stall terms for the tasks that add to their response time, assuming round-robin memory arbitration. Their analysis also assumes *DRAM Bank Partitioning* via the OS-level memory allocator PALLOC [8].

Contribution #1: We combine Ekberg and Yi's work and SCE. The rationale is that the former overlooks the effects of contention for platform resources. Conversely, SCE provides a form of scheduling isolation over those resources that is handy for safety-critical applications, but is criticality-oblivious.

Contribution #2: We identify and leverage an opportunity for improved schedulability and resource efficiency, afforded by such a unified approach. Consider a system conforming to the SCE principles (including the cache partitioning) and scheduled by a variant of Ekberg and Yi's algorithm. When a mode change occurs, L-tasks are dropped, which means that their cache partitions are no longer of any use. We therefore assign them to the remaining H-tasks, after the mode change,

to accommodate additional pages by the latter. This re-uses resources that would otherwise be idle and lowers the H-WCETs of jobs released after the mode change, leading to better scheduling performance. The challenge then is to develop schedulability analysis that safely quantifies the improvement.

II. System model and assumptions

We assume m identical physical cores (π_1 .. π_m) accessing DRAM via a shared memory controller. A core can have many outstanding memory requests. Prefetchers/speculative units are disabled. Our assumptions are inline with SCE [4]:

- The last-level cache is shared among the cores. The Colored Lockdown mechanism [7] is used to mitigate the intra-/inter-core interference. It allows a task to lock its most frequently used pages in the last-level cache, providing a deterministic approach to compute the residual number of memory accesses and WCET as a function of the number of locked pages.
- DRAM-bank partitioning (e.g., using PALLOC [8]) is not strictly required. However, lower and upper bounds on the memory access time of a single transaction (L_{min} and L_{max}, respectively) are needed as inputs. These can be computed as in [6]. The DRAM controller is round-robin.
- The memory bandwidth is managed via the MemGuard [5] OS-level regulation mechanism. It uses the the worst-case memory access time L_{max} to determine the maximum number of memory accesses, K, during a regulation period P, as $K \stackrel{\text{def}}{=} \frac{P}{L_{max}}$. The memory bandwidth of K memory accesses is distributed equally among different cores, i.e. $K_i \stackrel{\text{def}}{=} \frac{K}{m}$ requests per regulation period for core π_i. This budget is allocated at the start of the regulation period. The execution of a core is stalled until the end of current regulation period as soon as it exhausts its per-regulation-period budget of K_i memory requests.

Each task τ_i has a minimum inter-arrival time T_i, a relative deadline D_i (with $D_i \leq T_i$) and a criticality level $\kappa_i \in \{L, H\}$ (low or high, respectively). The subsets of low-criticality and high-criticality tasks are defined as $\tau(L) \stackrel{\text{def}}{=} \{\tau_i \in \tau | \kappa_i \in L\}$ and $\tau(H) \stackrel{\text{def}}{=} \{\tau_i \in \tau | \kappa_i \in H\}$. However, since (i) the (actual) WCET of a task depends on its number of pages (selected in order of access frequency) locked in place in the last-level cache and (ii) different *estimates* of that WCET (derived via different techniques), are to be used for the L-mode and H-mode, we extend Vestal's model by assuming that for each task, a different L-WCET/H-WCET pair can be available as input, for each possible value of the number of locked "hot" pages. For example $C_i^L(6)$ denotes the L-WCET of τ_i when this task is configured with its 6 "hottest" pages locked in the cache. In detail:

The Colored Lockdown approach determines the most frequently accessed (hot) pages for each task through the profiling framework in [7]. The WCET of a task in isolation is computed as a function of the number of hot pages locked in the last-level cache and represented as a *progressive lockdown curve* (WCET vs locked pages in last-level cache). The increase in number of locked pages in the last-level cache decreases the last-level cache misses and, consequently, also the WCET of the task. The approach proposed by Mancuso et al. [7] is a measurement-based technique, so its outputs are not provably safe, but they can serve as L-WCETs. However, some static analysis tools comprehensively cover all possible control flows (or even some infeasible paths) in a task, and these can be used for estimating the H-WCETs.

By safely modelling accesses to the hot pages locked-in by Colored Lockdown as "always hit", the static analysis tool can derive tighter WCET estimates than it would without this knowledge – and the improvement will be greater the more pages are locked in the cache. Hence, a progressive lockdown curve similarly exists for the H-WCET $C_i^H(\cdot)$.

Similarly to the WCETs, we extend the use of techniques of different conservativeness for estimating the residual number of memory accesses (last-level cache misses) of each task in L-mode and H-mode. Similar to WCET estimates, these values are functions of locked pages in the last-level cache ($\mu_i^L(\cdot)$, $\mu_i^H(\cdot)$) that decrease with the number of locked pages in the last-level cache. The utilisation of a task in L-mode (H-mode) of operation is defined as $U_i^L(\sigma) \stackrel{\text{def}}{=} \frac{C_i^L(\sigma)}{T_i}$ (resp., $U_i^H(\sigma) \stackrel{\text{def}}{=} \frac{C_i^H(\sigma)}{T_i}$), for σ locked pages in the last-level cache. We assume fully partitioned scheduling. i.e, no task ever migrates.

III. Cache scaling for mixed criticality system

A. Memory regulation stall

The MemGuard regulation mechanism bounds the number of DRAM accesses by each core. A core can be stalled for two main reasons. MemGuard allocates a memory access budget to each core per regulation period P. A core exceeding its budget is stalled until the start of the next regulation period; this is a *regulation stall*. As all cores share the DRAM controller, any delay in serving the memory accesses of a core due to accesses by other cores is a *contention stall*.

Mancuso et al [6] assume an even allocation of memory access budgets among cores and round-robin memory bus sharing. With these assumptions, the worst-case regulation-induced stall is computed for a given task τ_i performing $\mu_i(\cdot)$ residual memory accesses. It is shown that the regulation stall always dominates the contention stall in a given regulation period P. The duration of such a stall is $(P - K_i L_{min})$. The contention stall for any number of memory accesses $K_q \leq K_i$ within a given regulation period P is equal to $(m-1)K_q L_{max}$ under round robin arbitration. It has been shown [6, Th. 1]) that the stall due to $\mu_i(\cdot)$ memory requests issued by a task τ_i can be computed as:

$$\begin{aligned} stall(\tau_i, c, \sigma) = &\left\lceil \frac{\mu_i^c(\sigma)}{K_i} \right\rceil (P - K_i L_{min}) \\ &+ (m-1)(\mu_i^c(\sigma)) - (\left\lceil \frac{\mu_i^c(\sigma)}{K_i} \right\rceil - 1) K_i) L_{max} \end{aligned} \quad (1)$$

where, $c \in \{L, H\}$ and σ is the number of locked pages.

The L-WCET and H-WCET of a task τ_i with interference from other cores on the shared bus with σ locked pages in the last-level cache are defined as follows.

$$\bar{C}_i^L(\sigma) \stackrel{\text{def}}{=} C_i^L(\sigma) + stall(\tau_i, L, \sigma) \quad (2)$$

$$\bar{C}_i^H(\sigma) \stackrel{\text{def}}{=} C_i^H(\sigma) + stall(\tau_i, H, \sigma) \quad (3)$$

These values are computed using Equations (1)–(3), $\forall \sigma \in \{0, \ldots, \sigma_T\}$ on the progressive lockdown curves derived for the L-mode and H-mode of operation.

B. Last-level cache allocation

Let σ_i^L and σ_i^H be the number of pages by τ_i in the last-level cache in the two modes and σ_T be the total number of pages that fit in that cache. A good heuristic could be to set the σ_i^L values such that the total task set utilisation in L-mode is minimised. Intuitively, lower utilisation correlates with better schedulability. An optimal way to solve this heuristic would be the following Integer Linear Programming model (ILP):

$$\text{Mimimise} \sum_{\forall \tau_i \in \tau} \frac{\bar{C}_i^L(\sigma_i^L)}{T_i}, \text{ subject to} \sum_{\forall \tau_i \in \tau} \sigma_i^L \leq \sigma_T \quad (4)$$

Next, without backtracking, we determine the σ_i^H values.

$$\text{Mimimise} \sum_{\forall \tau_i \in \tau(H)} \frac{\bar{C}_i^H(\sigma_i^L)}{T_i} \quad (5)$$

$$\text{s.t. } \sigma_i^H \geq \sigma_i^L, \forall \tau_i \in \tau(H) \text{ and } \sum_{\forall \tau_i \in \tau(H)} \sigma_i^H \leq \sigma_T \quad (6)$$

The constraint in Equation (4) ensures that the total number of allocated pages does not exceed the available capacity of the cache. Pages abandoned by the L-tasks in the H-mode are "recycled" for H-tasks by the constraints in Equation (6). One counter-intuitive property of our proposed heuristics is that there may be a scenario in which the $\bar{C}_i^H(\sigma_i^H) \leq \bar{C}_i^L(\sigma_i^L)$. This could happen if the effects of the reduction of residual memory accesses for H-tasks in H-mode, from the additional pages, offset the pessimism from using a more conservative estimation technique for H-WCETs than for L-WCETs. Therefore, unlike the "classic" Vestal's model, we may have $\bar{C}_i^H(\sigma_i^H) \leq \bar{C}_i^L(\sigma_i^L)$ and better schedulability in the H-mode when compared to Ekberg and Yi's analysis.

C. Schedulability analysis

In the last step, we analyse the intra-core interference that a task may have due to memory accesses by other tasks running on the same core and integrate it to the schedualability analysis. In this paper, we have adapted Ekberg and Yi's schedulability analysis [3] to our system model. Ekberg and Yi proposed a demand-bound-function (dbf) based analysis that assumes tasks are scheduled with Earliest Deadline First (EDF). The deadlines of H-tasks are shortened in L-mode, so that they can stay "ahead of schedule" and perform a smooth transition from L- to H-mode of operation. In our approach, we currently perform no deadline-scaling but get a similar effect by allocating extra pages to the H-tasks in the last-level cache in H-mode[1]. Since Ekberg and Yi's technique is still usable as a schedulability test when the scaling factors are given as input, we use a scaling factor of 1 (no scaling) for all tasks. As in that work, we check for schedulability in the L-mode with a dbf-based [9] test. These derivations do not consider the fact that a task may be preempted by other tasks allocated to the same core. With the memory regulation employed, each preempting job in the worst-case may exhaust the allocated memory access budget within a given period P and hence, may be causing the core to stall upon the resumption of the preempted job. A job cannot preempt another job more than once. Therefore, one approach is to include the preemption overheads into the budget of preempting task.

Similar considerations apply to the adaptation of the test for cheching the schedulability in H-mode. Note though that any H-jobs caught up in the mode change may execute for up to $\bar{C}_i^H(\sigma_i^L)$ time units, since only subsequent H-jobs will execute with more hot pages (σ_i^H) in the cache.

IV. Conclusions

Using low-level information about the hardware platform and access regulation mechanisms, has potential to improve schedulability and confidence in the analysis of mixed-criticality systems. This work-in-progress paper is one step in that direction. Currently, we are developing a mixed criticality schedulability test that incorporates intra-core interference while ensuring the schedulability in L-mode, H-mode and the transition phase. Next, we will develop a testbed that incorporates these details and provides a platform to compare its performance with state-of-the-art mixed criticality scheduling algorithms. We hope this practical approach will outperform the existing state-of-the-art solutions.

References

[1] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in 28th RTSS, 2007.

[2] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in 24th ECRTS, July 2012, pp. 145–154.

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

[4] L. Sha, M. Caccamo, R. Mancuso, J.-E. Kim, M.-K. Yoon, R. Pellizzoni, H. Yun, R. Kegley, D. Perlman, G. Arundale et al., "Single core equivalent virtual machines for hard realtime computing on multicore processors," Univ. of Illinois at Urbana Champaign, Tech. Rep., 2014.

[5] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memory bandwidth management for efficient performance isolation in multi-core platforms," Trans. Computers, vol. 65, no. 2, pp. 562–576, Feb 2016.

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

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

[8] H. Yun, R. Mancuso, Z.-P. Wu, and R. Pellizzoni, "PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms," in 20th RTAS, April 2014, pp. 155–166.

[9] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," J. Real–Time Syst., 1990.

[1] In time, we intend to also incorporate (and speed up) the deadline-scaling.