CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

Response time analysis of memory-bandwidth- regulated multiframe mixed-criticality systems

Ishfaq Hussain
Muhammad Ali Awan
Pedro Souto
Konstantinos Bletsas
Eduardo Tovar

CISTER-TR-211006

2021/12/14

Response time analysis of memory-bandwidth- regulated multiframe mixed-criticality systems

Ishfaq Hussain, Muhammad Ali Awan, Pedro Souto, Konstantinos Bletsas, Eduardo Tovar

CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: hussa@isep.ipp.pt, awa@isep.ipp.pt, pfs@fe.up.pt, ksbs@isep.ipp.pt, emt@isep.ipp.pt

https://www.cister-labs.pt

Abstract

The multiframe mixed-criticality task model eliminates the pessimism in manysystems where the worst-case execution times (WCETs) of successive jobs varygreatly by design, in a known pattern. Existing feasibility analysis techniquesfor multiframe mixed-criticality tasks are shared-resource-oblivious, hence un-safe for commercial-o_-the-shelf (COTS) multicore platforms with a memorycontroller shared among all cores. Conversely, the feasibility analyses thataccount for the interference on shared resource(s) in COTS platforms do notleverage theWCET variation in multiframe tasks. This paper extends the state-of-the-art by presenting analysis that incorporates the memory access stall inmemory-bandwidth-regulated multiframe mixed-criticality multicore systems.An exhaustive enumeration approach is proposed for this analysis to furtherenhance the schedulability success ratio. The running time of the exhaustiveanalysis is improved by proposing a pruning mechanism that eliminates thecombinations of interfering job sequences that subsume others. Experimentalevaluation, using synthetic task sets, demonstrates up to 72% improvement interms of schedulability success ratio, compared to frame-agnostic analysis.

Response time analysis of memory-bandwidth-regulated multiframe mixed-criticality systems

Ishfaq Hussain[a,*], Muhammad Ali Awan[a], Pedro F. Souto[a,b], Konstantinos Bletsas[a], Eduardo Tovar[a]

[a] *CISTER Research Centre and ISEP/IPP, Porto, Portugal*
[b] *University of Porto, FEUP-Faculty of Engineering, Porto, Portugal*

Abstract

The multiframe mixed-criticality task model eliminates the pessimism in many systems where the worst-case execution times (WCETs) of successive jobs vary greatly by design, in a known pattern. Existing feasibility analysis techniques for multiframe mixed-criticality tasks are shared-resource-oblivious, hence unsafe for commercial-off-the-shelf (COTS) multicore platforms with a memory controller shared among all cores. Conversely, the feasibility analyses that account for the interference on shared resource(s) in COTS platforms do not leverage the WCET variation in multiframe tasks. This paper extends the state-of-the-art by presenting analysis that incorporates the memory access stall in memory-bandwidth-regulated multiframe mixed-criticality multicore systems. An exhaustive enumeration approach is proposed for this analysis to further enhance the schedulability success ratio. The running time of the exhaustive analysis is improved by proposing a pruning mechanism that eliminates the combinations of interfering job sequences that subsume others. Experimental evaluation, using synthetic task sets, demonstrates up to 72% improvement in terms of schedulability success ratio, compared to frame-agnostic analysis.

Keywords: Mixed criticality system, Multiframe task systems, Multiprocessor system on chip(MPSoC), Memory bandwidth regulation

1. INTRODUCTION

The trends in real-time embedded domains (e.g., automotive and avionics) favor *mixed-criticality* systems, where computing tasks of different criticalities co-exist and share resources. A task's criticality denotes the severity of that task failing, in conjunction with the probability of such failure. Higher-criticality tasks hence require stricter, costlier methodologies and worst-case execution

*Ishfaq Hussain
Email address: hussa@isep.ipp.pt (Ishfaq Hussain)

time (WCET) estimation techniques. Scheduling arrangements must also ensure that misbehaving lower-criticality applications cannot affect the timing behavior of critical ones. Vestal's mixed-criticality model does so (and promotes resource efficiency) by using multiple WCET estimates per task, with different confidence level. Its static variant (SMC) [1], picks the appropriate estimate for each task, as input to worst-case response time (WCRT) analysis. The adaptive (AMC) variant (also [1]), has modes and mode-specific WCETs. At mode switch (triggered by exceedance of a WCET estimate), lower-criticality tasks are dropped and more pessimistic WCETs are assumed for remaining tasks.

AMC/SMC was combined with the *multiframe task model* [2] in [3], to combat one common source of pessimism and inefficiency. In many systems, the WCETs of successive jobs by the same task vary greatly by design, according to a known pattern. Ignoring such variation, by using the maximum WCET for all jobs, inflates processor requirements. For example, MPEG codecs [4] feature different kinds of frames (P, I or B) in a repeating pattern, with very different worst-case processing requirements. Moreover, Avionics Digital Video Bus (ADVB) [5] frames are transmitted uncompressed, to minimise encoding/decoding delays, but the types of ADVB frames (e.g., header, audio or video) differ in end-node processing requirements. Other industrial applications [6, 7] collect small amounts of data periodically and store them in batch after N periods, in one costly operation. The multiframe model efficiently covers such scenarios, but until [3] did not apply to mixed-criticality applications.

This work makes [3] applicable to multicores, by incorporating *memory access regulation* and *stall analysis*. When multiple cores access memory via a shared memory controller, as in many commercial-of-the-shelf (COTS) chips used in embedded systems, regulation makes the system more timing-predictable. We assume a mechanism like *MemGuard* [8] from the SCE framework [9] that can be easily implemented in modern COTS platforms. Each core has a memory access budget, to consume within a given regulation period. Any core exceeding its budget is stalled until its replenishment, at the start of the next regulation period. In [10], such regulation was combined with SMC/AMC, but not for multiframe tasks. Conversely, in [11], it was applied to multiframe tasks, but not mixed-criticality. This work extends the state-of-the-art with:

1) New schedulability analysis for multiframe mixed criticality multicore systems, that incorporates the stall due to memory regulation.

2) An accuracy improvement for this analysis that provides tighter schedulability results, by considering all possible releases of higher-priority tasks.

3) A running-time optimisation for the analysis in 2), that prunes combinations of higher-priority task releases whose interference is dominated by others.

Experimental evaluations, using synthetic tasks, show up to 72% higher unweighted schedulability success ratio vs frame-agnostic analysis.

The paper is organized as follows: Section 2 discusses the state-of-the art. The system model is presented in Section 3. Section 4 discusses existing results on multiframe mixed-criticality systems and stall analysis. It closes with an example demonstrating that a task set deemed schedulable by a stall-oblivious analysis may be unschedulable when running in a system with memory regula-

tion. Section 5 presents a stall-aware WCRT analysis for memory-bandwidth-regulated multiframe mixed-criticality systems. Sections 6 and 7 offer optimisations to accuracy and analysis running time, respectively. Section 8 evaluates the new analyses, in terms of weighted schedulability and running time, compared to frame-agnostic analyses. Section 9 concludes.

2. Related Work

Multiframe Task Model. This task model by Mok and Chen [2] allows the WCETs of the successive jobs of the same task to vary with a repetitive pattern. The schedulability analysis in [2] vastly improves on conservative analyses that use a task's highest WCET estimate as the WCET of each job. Baruah et al. [12] improved on that by considering the actual frame pattern instead of an accumulatively monotonic reordered pattern [2]. The multiframe task model was further generalized [13] by allowing additional task attributes to differ among frames. Zuhily and Burns [14] devised exact analysis for multiframe tasks.

Mixed-Criticality Scheduling. The two main variants (for comprehensive review see [15]), static (SMC) [1] and adaptive mode-based (AMC) [16], of Vestal's mixed-criticality model [17] characterize each task with multiple WCET estimates, with a corresponding degree of confidence associated with a different criticality level (up to the task's own). In SMC [1], any job by a lower-criticality task is terminated, if it overruns its most conservative WCET estimate. Baruah, Burns and Davis [16] ported standard WCRT analysis to SMC. They also proposed an adaptive mixed-criticality (AMC) scheduling technique that adds modes of operation. Two schedulability tests for AMC were devised: AMC-rtb and the tighter, but more complex, AMC-max. This work was extended to arbitrary deadlines [18] and an arbitrary number of criticality levels [19].

Memory Bandwidth Regulation. Timing predictability in safety-critical multicore systems with shared resources is challenging. Many works offer analysis for computing the interference on shared resources and integrate their effect into the schedulability analysis [20, 9, 21, 22, 23]. Mancuso et al. [9] developed a worst-case stall analysis with a software-based memory regulation mechanism MemGuard [8] for fixed-priority-scheduled partitioned multicores and integrated it into schedulability analysis. Yao et al. [21] and Pellizzoni and Yun [22] generalized Mancuso's analysis [9] by allowing uneven memory budget assignment to cores, for greater efficiency when their bandwidth requirements are too diverse. Mancuso et al. [24] explicitly consider the known memory access budgets of other cores to further improve the stall analysis. Agrawal et al. [25] also proposed a dynamic memory bandwidth assignment mechanism that varies the budgets over time, based on the application requirements and formulated schedulability analysis for this arrangement. Awan et al. [26] extended the SCE model [27] with periodic EDF servers and per-server memory budgets and explored stall analysis for multiple memory controllers [28].

Some works integrate the stall analysis into the response time analysis of mixed-criticality systems [29, 10, 21]. Yun et al. in [29] regulate the memory access rates of cores running low-criticality tasks. Yao et al. [21] later generalized this by regulating the accesses from all cores. To achieve isolation from each other, low- and high-criticality tasks are scheduled on separate cores, which may inefficiently utilize COTS platforms. In response, Awan et al. [10] let tasks of different criticalities share cores and presented a low-complexity schedulability test based on AMC-rtb. That was improved [28] by dynamically adjusting the memory bandwidth upon mode change and an AMC-max-based test. Kritikakou et al. [30, 31] deal with interference on a high-criticality task from low criticality tasks running on a different core, similarly to a mode change [16]. High-criticality tasks are continuously monitored and any low-criticality tasks exceeding a pre-specified execution time limit are stopped from further executing. A multicore implementation of that work [31] demonstrated its effectiveness and a dynamic [32] variant reduces the time spent in the controller.

Stalls from memory regulation were incorporated to SMC, AMC-rtb/-max in [10, 28]; and to the WCRT analysis of multiframe tasks in [11]. Multiframe tasks and mixed criticalities were also combined [33, 3], but for single-core systems (by nature, requiring no memory regulation) – not multicore systems with memory-bandwidth regulation, as in this work.

3. System Model

3.1. Hardware Platform

As our work relies on [21] and extends its results, it is bound by the assumptions therein. Specifically, K identical cores $\{P_1, P_2, ...P_K\}$ share a memory controller through which main memory is accessed. Any speculative units and prefetchers are disabled. The scheduling policy for both memory controller and interconnect is round robin. For the regulation of memory accesses, a mechanism like Memguard[1] [8], that requires no information about tasks running on other cores, is used. This allows to compute the response time of a task independently of the workload on other cores. This relaxes the certification effort. The outer cache is partitioned or private to each core or can be even partitioned among tasks to avoid cache-related preemption delay issues. Performance measuring counters count the memory accesses. As in [8, 21], the memory access time is considered constant. Although not a requirement, the main memory can be partitioned (e.g. via PALLOC [34]) for improved predictability.

The memory access budget Q_k of a core k is the maximum number of accesses it can make in a regulation period P. If a core depletes its budget, it is stalled for the rest of the current regulation period. At the start of each regulation period, a core's budget is replenished. We assume that the regulation periods

[1]Specifically, all that is needed is (i) hardware timers (ii) performance monitoring counters to track last-level-cache misses and (iii) the generation of an exception upon a counter overflowing. All these features are commonly found in COTS platforms [8].

are equal and synchronised for all cores. The memory bandwidth share of a core is $b_k = \frac{Q_k}{P}$, and $\sum_{k=0}^{K} b_k \leq 1$, i.e., the total allocated bandwidth does not exceed the total available bandwidth. As in [21], the length P of the regulation period is expressed in multiples of the constant memory access latency.

3.2. Task model

As in [33], each task τ_i has a criticality level κ_i. The set of criticality levels is totally ordered, and each job of τ_i has multiple WCET estimates, one for each criticality level lower than or equal to κ_i, with corresponding degree of confidence. For simplicity, we assume just two criticality levels, high (H) and low (L). Unlike [10, 28], the jobs of τ_i do not necessarily have the same WCET estimates, but their WCET estimates are periodic. I.e., for some integer F_i, jobs n and $n+F_i$ of task τ_i have the same WCET estimates. We call a sequence of F_i consecutive jobs a *superframe*, and each of its jobs a *frame*.

Thus, each task is characterized by the 5-tuple $\tau_i = (\vec{C}_i^L, \vec{C}_i^H, D_i, T_i, \kappa_i)$, where κ_i is the criticality level, T_i is the minimum interarrival time, $D_i \leq T_i$ is the relative deadline, and $\vec{C}_i^L$ and $\vec{C}_i^H$ are the WCET F_i-tuples for the L and H criticality levels, respectively. For example, for a task with $\kappa_i = H$ (an "H-task"), $\vec{C}_i^L = \left(C_{i,0}^L, C_{i,1}^L, C_{i,2}^L, ..., C_{i,(F_i-1)}^L\right)$ consists of the L-level WCET estimates ("L-WCETs") of F_i consecutive jobs and, accordingly, $\vec{C}_i^H = \left(C_{i,0}^H, C_{i,1}^H, C_{i,2}^H, ..., C_{i,(F_i-1)}^H\right)$ consists of their H-level WCET estimates ("H-WCETs"). For a task with $\kappa_i = L$ (an "L-task") only $\vec{C}_i^L$ is defined.

Memory accesses and CPU computation do not overlap, to comply with [21]. Thus, each frame's κ-WCET, $C_{i,j}^k$, is given by $C_{i,j}^\kappa = C_{i,j}^{e|\kappa} + C_{i,j}^{m|\kappa}$, where $C_{i,j}^{m|\kappa}$ is the worst-case memory access time estimate and $C_{i,j}^{e|\kappa}$ is the worst-case CPU computation time estimate. Accordingly, the κ-WCET of job f is denoted by the pair $\left(C_{i,(f \bmod F_i)}^{e|\kappa}, C_{i,(f \bmod F_i)}^{m|\kappa}\right)$. Tasks are partitioned to the cores and scheduled preemptively with fixed priorities, assigned offline by any algorithm.

3.3. Mixed-Criticality Models

A critical issue in mixed-criticality models is to ensure that a job that overruns its WCET does not cause jobs of higher-criticality tasks to miss their deadline. This paper, builds on the adaptive mixed criticality (AMC) approach.

AMC assumes as many system modes as criticality levels. The system executes only tasks of criticality not smaller than the level corresponding to the current operating mode. If a job of any task exceeds its WCET estimate for the criticality level corresponding to the current mode, the system drops all tasks whose criticality level corresponds to the current mode and switches to the next highest mode. The initial mode is the lowest one, which includes all tasks.

A key assumption is that a WCET estimate for a given criticality level is never less conservative than the WCET for a lower criticality level, thus providing a higher level of confidence that it will not be overrun. For example, in a system with only two criticality levels, L and H, the L-WCET is smaller

than or equal to the H-WCET. Generalizing to our model, for every frame k of every H-task i, $C_{i,k}^{e|L} \leq C_{i,k}^{e|H}$ and $C_{i,k}^{m|L} \leq C_{i,k}^{m|H}$.

4. Background

This work builds on multiframe mixed-criticality analyses for single-core systems [3, 33] and Yao et al. memory stall and response time analyses [21].

4.1. Multiframe response time analysis

Baruah et al. [12] derived a sufficient response time analysis for multiframe task sets, based on the one for fixed priority preemptive scheduling, i.e. :

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{1}$$

The above recurrence upper-bounds the interference by higher priority task τ_j during an interval of duration R_i by multiplying τ_j's worst-case execution time, C_j, by the number of job releases of τ_j in that interval. In a multiframe task model however, because different jobs may have different WCETs, counting the number of job releases is not enough. Thus Baruah et al. defined function:

$$g(\tau_i, k) = \begin{cases} 0 & \text{if } k = 0 \\ \max\limits_{0 \leq j < F_i} \sum\limits_{\ell=j}^{j+k-1} C_{i,(\ell \bmod F_i)} & \text{if } 1 \leq k \leq F_i \\ q \cdot g(\tau_i, F_i) + g(\tau_i, r), \text{where } q = k \operatorname{div} F_i,\ r = k \bmod F_i & \text{otherwise} \end{cases} \tag{2}$$

which upper bounds the maximum cumulative execution time of k consecutive jobs of task τ_i. They also defined function:

$$G(\tau_i, t) = g\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil\right) \tag{3}$$

that upper bounds the maximum cumulative execution time of jobs of τ_i over an interval of length t. Baruah et al. used $G(\tau_j, R_i)$ to upper-bound the interference from higher-priority task τ_j in the WCRT recurrence for multiframe tasks:

$$R_i = g(\tau_i, 1) + \sum_{j \in hp(i)} G(\tau_j, R_i) \tag{4}$$

4.2. AMC-max response-time analysis

The AMC schedulability analysis considers the execution in L-mode, in H-mode and upon mode switch. The analyses for L-mode and H-mode use the standard WCRT recurrence of fixed-priority scheduling (1) with the corresponding WCET estimates for all jobs. For the response time upon mode switch, [16] proposes two analyses: AMC-rtb and AMC-max. Here, we focus on the latter,

which provides tighter bounds by taking into account the time instant s when the mode switch occurs (relative to the release of the job under analysis).

In [16], AMC-max is also based on the standard WCRT recurrence of fixed-priority preemptive scheduling as shown in (5).

$$R_i^*(s) = C_i^H + \sum_{j \in hpL(i)} \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1 \right) C_j^L + \sum_{k \in hpH(i)} \left(C_k^L \left(\left\lceil \frac{R_i^*(s)}{T_k} \right\rceil - M(k, s, R_i^*(s)) \right) + C_k^H M(k, s, R_i^*(s)) \right) \quad (5)$$

$$M(k, s, t) = min \left\{ \left\lceil \frac{t - s - (T_k - D_k)}{T_k} \right\rceil + 1, \left\lceil \frac{t}{T_k} \right\rceil \right\} \quad (6)$$

In the RHS of (5), the first summation upper bounds the interference by higher priority L-tasks, whereas the second summation upper bounds the interference by higher priority H-tasks. Critical for the safety of this analysis is the parameter M, (6), which upper bounds the number of jobs of higher priority task τ_k that complete after the mode switch; those execute, in the worst-case, for C_k^H each.

Note that (5) depends on s, so we need to compute the value of s that leads to the maximum value of the response time, i.e.

$$R_i^* = max R_i^*(s), \forall s \quad (7)$$

By analysis of the RHS of (5), Baruah et al. argue that it is enough to compute $R_i^*(s)$ only for values of s that correspond to the release times of higher-priority L-tasks, when the first job of each of these tasks is released at the same time as the job under analysis and the other jobs arrive as soon as possible. Note that one needs to check only values of s smaller than the response time in L-mode, R_i^L, otherwise the task under analysis would complete before the mode switch.

4.3. AMC-max for multiframe mixed-criticality

Because in the mixed-criticality model, each job in a multiframe of a H-task has two WCET-estimates, one for L-mode and another for H-mode, [3], defines the cumulative L-WCET of any sequence of k jobs of task τ_i, $g^L(\tau_i, k)$, an extension of the $g(\tau_i, k)$ function, (2), from [12]:

$$g^L(\tau_i, k) = \begin{cases} 0 & \text{if } k = 0 \\ \max\limits_{0 \le j < F_i} \sum\limits_{\ell=j}^{j+k-1} C^L_{i,(\ell \bmod F_i)} & \text{if } 1 \le k \le F_i \\ q \cdot g^L(\tau_i, F_i) + g^L(\tau_i, r), \text{where } q = k \operatorname{div} F_i,\ r = k \bmod F_i & \text{otherwise} \end{cases} \quad (8)$$

$g^H(\tau_i, k)$ for cumulative H-WCET is defined analogously. The cumulative-over-time WCET function $G(\tau_i, t)$, given by (3), is likewise extended in [3] to:

$$G^L(\tau_i, t) = g^L\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil\right) \quad \text{and} \quad G^H(\tau_i, t) = g^H\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil\right) \quad (9)$$

Furthermore, to analyze the response time upon mode switch, [3] defines $g^*(\tau_i, \ell^L, \ell^H)$ to compute the cumulative WCET of a sequence of $\ell^L+\ell^H$ jobs of τ_i, the first ℓ^L of which take their L-WCET:

$$g^*(\tau_i,\ell^L,\ell^H) = \begin{cases} g^H(\tau_i,\ell^H) & \text{if } \ell^L = 0 \\ g^L(\tau_i,\ell^L) & \text{if } \ell^L \neq 0 \wedge \ell^H = 0 \\ \max\limits_{0\le j<F_i}\{ \sum\limits_{k=j}^{j+\ell^L-1} C^L_{i,(k \bmod F_i)} & \\ \quad + \sum\limits_{k=j+\ell^L}^{j+\ell^L+\ell^H-1} C^H_{i,(k \bmod F_i)}\} & \text{if } 1{\le}\ell^L{<}F_i \wedge 1{\le}\ell^H{<}F_i \\ q^L \cdot g^L(\tau_i,F_i) + g^*(\tau_i,r^L,r^H) + q^H \cdot g^H(\tau_i,F_i) & \text{otherwise} \end{cases} \tag{10}$$

where $q^L{=}(\ell^L \text{ div } F_i)$, $r^L = (\ell^L \bmod F_i)$, $q^H = (\ell^H \text{ div } F_i)$ and $r^H = (\ell^H \bmod F_i)$.

Additionally, in order to simplify the WCRT recurrence, [3] defines:

$$G^{L+}(\tau_i,t) = g^L\left(\tau_i, \left\lfloor \frac{t}{T_i} \right\rfloor + 1\right) \tag{11}$$

With these definitions, [3] transformed the AMC-max response time recurrence for the mode switch, (5), to a recurrence for computing the response time of job j of multiframe task τ_i when there is mode switch at time s:

$$R^*_{i,j}(s) = C^H_{i,j} + \sum_{k\in hpL(i)} G^{L+}(\tau_k,s) + \sum_{k\in hpH(i)} g^*\left(\tau_k, \ell^L(k,R^*_{i,j}(s),s), \ell^H(k,R^*_{i,j}(s),s)\right) \tag{12}$$

$$\ell^L(k,t,s) = \left\lceil \frac{t}{T_k} \right\rceil - \ell^H(k,t,s) \tag{13}$$

$$\ell^H(k,t,s) = M(k,s,t) \tag{14}$$

where $M(k,s,t)$ is given by (6). Therefore, $R^*_{i,j} = max\{R^*_{i,j}(s) : 0 < s < R^L_{i,j}\}$. Again, it suffices to compute $R^*_{i,j}$ only for those values of s specified for the original AMC-max. To compute the WCRT of task τ_i, one needs to compute $R^*_{i,j}$ for every job j in a superframe and take the maximum.

$$R^*_i = max\{R^*_{i,j} : 0 \le j < F_i\} \tag{15}$$

Finally, [3] proposes trading-off computation cost for pessimism, by computing the WCRT of only the longest H-mode job in a superframe. Thus (12) becomes:

$$R^*_i(s) = g^H(\tau_i,1) + \sum_{j\in hpL(i)} G^{L+}(\tau_j,s) + \sum_{k\in hpH(i)} g^*\left(\tau_k, \ell^L(k,R^*_i(s),s), \ell^H(k,R^*_i(s),s)\right) \tag{16}$$

which must be evaluated only for values of s smaller than R^L_i, i.e.:

$$R^*_i = max\{R^*_i(s) : 0 < s < R^L_i\} \tag{17}$$

4.4. Stall Analysis

To quantify the interference from the sharing of memory and interconnect among cores, we build upon the schedulability analysis of Yao et al. [21], which provides stall analysis for memory-regulated multicore single-criticality systems. That stall analysis assumes that the task under analysis is the only task on its core. Therefore, an active task is either executing, accessing memory or stalled.

With memory regulation, a task may experience two kinds of stalls: 1) *contention stall*, arising from the sharing of the memory and the interconnect among the cores; 2) *regulation stall*, occuring when a core exhausts its budget. The worst-case stall experienced by a task depends essentially on two values: the fraction of the total memory bandwidth assigned to the core running the task, $b{=}\frac{Q}{P}$, and the fraction of the task's total execution time that is spent accessing memory, $r{=}\frac{C^m}{C}$, where $C = C^m + C^e$. Indeed, if $b \leq \frac{1}{K}$, where K is the number of cores, the worst-case stall occurs by maximizing the number of regulation stalls. Otherwise, the worst case occurs by maximizing the number of contention stalls. If there is enough computation, then all memory accesses can suffer the maximum contention stalls. However, if there is not enough computation, some regulation periods may experience regulation stalls. Thus, Yao's analysis yields 3 cases:

Case 1: If $b \leq \frac{1}{K}$, then the stall is maximum when the number of regulation periods with a regulation stall is maximum, and it can be upper-bounded by:

$$stall(C^e, C^m) = \begin{cases} \frac{C^m}{Q}(P-Q) + (K-1)Q & \text{if } C^m \bmod Q = 0 \\ \left\lceil \frac{C^m}{Q} \right\rceil (P-Q) + (K-1)(C^m \bmod Q) & \text{otherwise} \end{cases} \tag{18}$$

Case 2: If $b > \frac{1}{K}$ and $r = \frac{C^m}{C} < \frac{1-b}{(K-1)b}$, then the stall is maximum when every memory access suffers interference by all other cores and it can be bounded by:

$$stall(C^e, C^m) = (P-Q) + (K-1) \cdot C^m \tag{19}$$

Case 3: If $b{>}\frac{1}{K}$ and $r{=}\frac{C^m}{C} \geq \frac{1-b}{(K-1)b}$, then stall is bounded by

$$stall(C^e, C^m) = \begin{cases} (1+A)(P-Q) + r_1 & if\ C \leq (1+A)Q \\ \left(1 + \frac{C}{Q}\right)(P-Q) + r_2 & \text{otherwise} \end{cases} \tag{20}$$

$$\text{where, } A = \left\lfloor \frac{C^e}{Q - RBS} \right\rfloor,\ RBS = \frac{P-Q}{K-1},$$

$r_1 = \min\{P-Q, (K-1)(C^m - A \cdot RBS)\},\ r_2 = \min\{P-Q, (K-1)(C\, mod\, Q)\}$

All three cases consider an initial (regulation) stall of $(P-Q)$ because, in the worst case, when a task is scheduled to run on a core, the latter's memory access budget is already depleted.

To apply the stall to the WCRT analysis of preemptive fixed-priority scheduling, [21] uses the concept of a synthetic task with the following parameters:

$$\widehat{C_i^e(t)} = C_i^e + \sum_{j \in hp(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil C_j^e$$

Table 1: Example task set and WCRT computed by stall-oblivious AMMC-max analysis.

Task	κ	$\vec{C}_i^L$	$\vec{C}_i^H$	$T = D$	WCRT	
					L-mode	Mode switch
τ_1	L	{(1,2),(2,2),(6,1),(4,3)}	{-}	20	7	NA
τ_2	H	{(3,2)(5,1)(2,1)}	{(6,4),(10,2),(4,2)}	30	13	19
τ_3	H	{(1,3),(2,1)}	{(2,6),(4,2)}	40	17	27

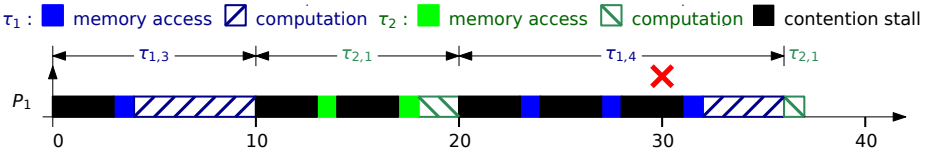

Figure 1: L-mode execution scenario with stalls for the task set in Table 1 showing that τ_2 misses its deadline, even though the AMMC-max stall-oblivious analysis deems it schedulable. Assuming a four core platform with a regulation period of 10. P_1's memory budget is 3.

$$\widehat{C_i^m(t)} = C_i^m + \sum_{j \in hp(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil C_j^m$$

I.e. it treats the job under analysis plus the jobs of interfering tasks arriving in the response time interval as a single job of a synthetic task. Thus, the classic WCRT equation becomes:

$$R_i = C_i + \sum_{j \in hp(\tau_i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j + stall(\widehat{C_i^e(R_i)}, \widehat{C_i^m(R_i)})$$

4.5. Example

We present an example of how the stall-oblivious analysis for AMMC-max presented in Section 4.3 may be unsafe in the presence of memory regulation.

Consider a quad-core system using memory regulation based on MemGuard, with a regulation period of 10 time units (worst-case memory access latency). Assume that cores P_1 to P_4 have memory budgets of 3, 3, 2 and 2, respectively.

P_1 runs three multiframe tasks whose characteristics are shown in Table 1. The two right-most columns in this table show the worst-case response times of all tasks computed by the stall-oblivious AMMC-max analysis (Section 4.3). Since no WCRT exceeds the respective task deadline, the stall-oblivious AMMC-max analysis deems the task set schedulable.

However, this may not be the case as illustrated by the execution of P_1's tasks in Figure 1, where blue is used for τ_1, green for τ_2 and black for contention stalls. For each task, we use a solid pattern to represent memory accesses, and a diagonal line pattern for computation. In this example, we assume that the system is operating in L-mode and that τ_1's third frame and τ_2's first frame arrive at time 0, the beginning of a regulation period. (The $\tau_{i,j}$ labels above P_1's execution time line denote that P_1 is executing frame j of τ_i.) Assuming deadline monotonic scheduling, P_1's scheduler decides to run τ_1, which incurs a cache miss. Further assume that at the time P_1 issues a memory request, the

other 3 cores also issue memory requests and that the memory controller, which executes a round-robin scheduler, determines that P_1's request is served only after the requests of the other 3 processors. Hence, τ_1 incurs a stall for 3 time units, after which it performs its memory access. Next, τ_1 performs computation for 6 time units, and completes at time 10. The scheduler now decides to run τ_2. Again, we assume that, before any computation, τ_2 suffers two last-level cache misses, both of which result in memory requests that incur 3-time-unit memory stalls. At time 18, when the second memory access completes, τ_2 starts performing computation. However, after 2 time units, at time 20, τ_1's fourth frame arrives, and the scheduler preempts τ_2, and assigns P_1 to τ_1. Again, τ_1 first performs 3 memory accesses, incurring a contention stall of 3 time units each. After the third memory access terminates, τ_1 executes for 4 time units completing at time 36. The scheduler then resumes τ_2; it executes for one time unit and completes at time 37. Since τ_2's relative deadline is 30 time units, in this execution τ_2 misses its deadline (represented by the red cross in Figure 1), thus showing that with memory regulation this task set is not schedulable.

5. Memory-aware WCRT Analysis

We now extend the AMC-max based WCRT analysis for mixed-criticality multi-frame tasks [33], AMMC-max, summarized in Section 4.3, to systems using memory regulation. Our approach is based on Yao's analysis [21], summarized in Section 4.4 and it is safe, simple and efficient, but somewhat pessimistic. Section 6 presents tighter analysis whose run-time is significantly larger.

The simplicity of the analysis arises essentially from our approach to upper bound the stall. This approach and the use of a single recurrence per task, see (16) and (17), lead to a rather efficient analysis.

As a first step, we extend some g and G functions described in Section 4 to compute the maximum cumulative computation time and the maximum cumulative memory access time required to compute the stall. More specifically, we define functions $g^{e|L}/g^{m/L}$ and $G^{e|L}/G^{m/L}$, based on g^L, (8), and G^L, (9), respectively. For example, we define:

$$g^{e|L}(\tau_i, k) = \begin{cases} 0 & \text{if } k = 0 \\ \max\limits_{0 \le j < F_i} \sum\limits_{\ell=j}^{j+k-1} C^{e|L}_{i,(\ell \bmod F_i)} & \text{if } 1 \le k \le F_i \\ q \cdot g^{e|L}(\tau_i, F_i) + g^{e|L}(\tau_i, r) & \text{otherwise} \\ \quad \text{where } q = k \text{ div } F_i \text{ and } r = k \bmod F_i & \end{cases} \tag{21}$$

$$G^{e|L}(\tau_i, t) = g^{e|L}\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil\right) \tag{22}$$

Functions $g^{e|H}/g^{m|H}$ and $G^{e|H}/G^{m|H}$ are derived similarly from g^H and G^H.

Since we consider adaptive mixed-criticality with only two criticality levels, high (H) and low (L), the system operates in two modes. In L-mode all tasks execute, while just the H-tasks in H-mode. Hence, we must consider WCRT

in L-mode, in (stationary) H-mode and in transient H-mode, i.e. when a task is caught by a mode transition. In all cases, we take into account the effect of memory regulation, by using the approach in [21] (summarized in Section 4.4), i.e. by adding a term that upper bounds the stall to the respective WCRT equation derived in [33]. For example, in L-mode, the WCRT becomes:

$$R_i^L = g^L(\tau_i, 1) + \sum_{j \in hp(i)} G^L(\tau_j, R_i) + stall(\widehat{C_i^{e|L}(R_i^L)}, \widehat{C_i^{m|L}(R_i^L)}) \tag{23}$$

The *stall* term is computed using the algorithm in [21] and summarized in Section 4.4. The crux is what values to use for arguments $\widehat{C_i^{e|L}(R_i^L)}$ and $\widehat{C_i^{m|L}(R_i)}$. In the multiframe task model, the computation time and the memory access time of a sequence of jobs of a given length depends on the first job in that sequence. Furthermore, the sequence with the maximum computation time may be different from the sequence with the maximum memory access time. To ensure safety, $\widehat{C_i^{e|L}(R_i^L)}$ and $\widehat{C_i^{m|L}(R_i^L)}$ must be upper bounds of the computation and memory access time of a synthetic task composed by the largest L-job of the task under analysis, τ_i, and all jobs of higher-priority tasks that arrive in the response time interval. Thus we bound each of these components independently:

$$\widehat{C_i^{e|L}(t)} = g^{e|L}(\tau_i, 1) + \sum_{j \in hp(i)} G^{e|L}(\tau_j, t) \tag{24}$$

$$\widehat{C_i^{m|L}(t)} = g^{m|L}(\tau_i, 1) + \sum_{j \in hp(i)} G^{m|L}(\tau_j, t) \tag{25}$$

This is safe, but may be pessimistic as, for any task τ_j, the sequence of jobs with the largest computation demand, $G^{e|L}(\tau_j, R_i^L)$, may differ from the sequence of jobs with the largest memory access time, $G^{m|L}(\tau_j, R_i^L)$.

In steady H-mode, the WCRT of an H-task τ_i resembles that for the L-mode, except that there is interference only by higher-priority H-tasks and H-mode WCET estimates are used:

$$R_i^H = g^H(\tau_i, 1) + \sum_{\tau_j \in hpH(i)} G^H(\tau_j, R_i^H) + stall(\widehat{C_i^{e|H}(R_i^H)}, \widehat{C_i^{m|H}(R_i^H)}) \tag{26}$$

where the arguments of *stall*, $\widehat{C_i^{e|H}(R_i^H)}$ and $\widehat{C_i^{m|H}(R_i^H)}$, are given by (27) and (28), which again upper-bound the two components independently:

$$\widehat{C_i^{e|H}(t)} = g^{e|H}(\tau_i, 1) + \sum_{k \in hpH(i)} G^{e|H}(\tau_k, t) \tag{27}$$

$$\widehat{C_i^{m|H}(t)} = g^{m|H}(\tau_i, 1) + \sum_{k \in hpH(i)} G^{m|H}(\tau_k, t) \tag{28}$$

In *transient H-mode* the WCRT equation, (16), becomes:

$$R_i^*(s) = g^H(\tau_i, 1) + \sum_{j \in hpL(i)} G^{L+}(\tau_j, s) + \sum_{k \in hpH(i)} g^*(\tau_k, \ell^L(k, R_i^*(s), s), \ell^H(k, R_i^*(s), s)) + stall(\widehat{C_i^{e*}(R_i^*(s))}, \widehat{C_i^{m*}(R_i^*(s))}) \quad (29)$$

where ℓ^L and ℓ^H are given by (13) and (14), respectively, and the arguments of *stall*, $\widehat{C_i^{e*}(R_i^*(s))}$ and $\widehat{C_i^{m*}(R_i^*(s))}$, are given by:

$$\widehat{C_i^{e*}(t)} = g^{e|H}(\tau_i, 1) + \sum_{j \in hpL(i)} G^{e|L+}(\tau_j, s) + \sum_{k \in hpH(i)} g^{e|*}(\tau_k, \ell^L(k, t, s), \ell^H(k, t, s))$$

$$\widehat{C_i^{m*}(t)} = g^{m|H}(\tau_i, 1) + \sum_{j \in hpL(i)} G^{m|L+}(\tau_j, s) + \sum_{k \in hpH(i)} g^{m|*}(\tau_k, \ell^L(k, t, s), \ell^H(k, t, s))$$

where the $g^{e|*}$ and $g^{m|*}$ functions are similar to the g^* function, (10), except that they use the respective component of the WCET. Similarly, $G^{e|L+}$ and $G^{m|L+}$ are similar to G^{L+} function, (11), except that they use the respective component of WCET. The WCRT is the maximum of all the responses computed for all mode switch instants s less than R_i^L:

$$R_i^* = max\{R_i^*(s) : 0 < s < R_i^L\}$$

Actually, rather than considering all time instants less than R_i^L it is enough to consider only the release time instants of higher-priority L-tasks, assuming that they are released at the same time as the task under analysis as often as possible. Indeed, the original arguments for this approach in [35] are still applicable. By analysing, the RHS of equation (29), we observe that the interference by higher-priority H-tasks decreases with s, whereas the interference by higher-priority L-tasks is a step function that increases at the instant these tasks are released. This is also true for their WCET components, i.e. the computation time and the memory access time, and therefore for the stall.

6. Exhaustive analysis

The analysis in Section 5 is pessimistic because it uses (i) Baruah's g/G functions in accounting the interference by higher-priority tasks and also (ii) independent upper bounds on the accumulated computation time and the accumulated memory access time of each job sequence of a higher-priority task. In this section, we address this pessimism with an exhaustive analysis, which builds on that of Zuhily and Burns [14]. In Section 7 we reduce its complexity.

6.1. Notation and underlying principles

Fixed-priority WCRT analysis relies on upper-bounding the number of job releases by each higher-priority task until the task under analysis completes. For

multiframe tasks, this is not enough. For any given number of interfering jobs by a given higher-priority task, there are as many potential interfering sequences as frames in a superframe: one sequence per initial frame. Because each frame may have different WCET, each such sequence may cause different interference.

To enumerate all interfering higher-priority frame sequences of a given length, we adapt the notation of [14]. Let $L_i = \{0, \ldots, F_i\text{-}1\}$ denote the set of (the indexes of) initial frames of all job sequences of task τ_i. Assuming, without loss of generality, that tasks are numbered from higher to lower priority, each element of the Cartesian product $V_i = \prod_{j=1}^{i-1} L_j$ denotes a tuple of initial frames, one per higher priority task, of all the sets (combinations) of interfering job sequences, consisting of one such sequence per task in $hp(i)$. For example, we have three tasks with rate-monotonic priority:$\tau_1 = \{(3,4,7,7),(),20,20,L\}$, $\tau_2 = \{(5,6,3),(),30,30,L\}$ and $\tau_3 = \{(4,3),(8,6),40,40,H\}$. Then, $V_3 = \{(0,0),(0,1),(0,2),\ldots,(3,0),(3,1),(3,2)\}$ is the Cartesian product of $L_1 = \{0,1,2,3\}$ and $L_2 = \{0,1,2\}$. $(1,0) \in V_3$ denotes two sequences of interfering frames, one by τ_1, starting with frame 1, and another by τ_2, starting with frame 0. Note that $(1,0)$ just specifies the initial frames of the interfering sequences. The lengths of the sequences are specified differently. Moreover, to upper-bound the interference of every frame sequence, [14] defines the cumulative function:

$$g(\tau_i, j, k) = \sum_{\ell=j}^{j+k-1} C_{i,(\ell \ mod \ F_i)} \tag{30}$$

for k=1,2,.... I.e., $g(\tau_i, j, k)$ computes the cumulative WCET of a sequence of k frames of τ_i, starting with frame j. We generalize this cumulative function to:

$$g^{\kappa}(\tau_i, j, k) = \begin{cases} \sum_{\ell=j}^{j+k-1} C^{L}_{i,(\ell \bmod F_i)} & \text{if } \kappa = L \\ \sum_{\ell=j}^{j+k-1} C^{H}_{i,(\ell \bmod F_i)} & \text{if } \kappa = H \end{cases} \tag{31}$$

for mixed-criticality and define (32) to compute cumulative computation and

$$g^{c|\kappa}(\tau_i, j, k) = \begin{cases} \sum_{\ell=j}^{j+k-1} C^{e|L}_{i,(\ell \bmod F_i)} & \text{if } c = e \text{ and } \kappa = L \\ \sum_{\ell=j}^{j+k-1} C^{m|L}_{i,(\ell \bmod F_i)} & \text{if } c = m \text{ and } \kappa = L \\ \sum_{\ell=j}^{j+k-1} C^{e|H}_{i,(\ell \bmod F_i)} & \text{if } c = e \text{ and } \kappa = H \\ \sum_{\ell=j}^{j+k-1} C^{m|H}_{i,(\ell \bmod F_i)} & \text{if } c = m \text{ and } \kappa = H \end{cases} \tag{32}$$

6.2. AMMC-max Exhaustive Analysis

To perform an exhaustive analysis, we compute the response time of every frame of a multiframe task, considering all possible interfering sequences by higher priority tasks. Let $v \in V_i$ denote a set of interfering sequences, one per task with higher priority than τ_i, such that the first frame of each sequence is the corresponding element of v.

L-mode analysis. The L-mode WCRT of frame j of a task τ_i subjected to v is:

$$R^{v|L}_{i,j} = C^{L}_{i,j} + \sum_{k \in hp(i)} g^{L}\left(\tau_k, v(k), \left\lceil \frac{R^{v|L}_{i,j}}{T_k} \right\rceil\right) + stall\left(\widehat{C^{e|L}_{i,j}(R^{v|L}_{i,j})}, \widehat{C^{m|L}_{i,j}(R^{v|L}_{i,j})}\right) \tag{33}$$

where $v(k)$ denotes element k of tuple v, and arguments $\widehat{C_{i,j}^{e|L}(R_{i,j}^{v|L})}$ and $\widehat{C_{i,j}^{m|L}(R_{i,j}^{v|L})}$ of the stall function are given by:

$$\widehat{C_{i,j}^{e|L}(t)} = C_{i,j}^{e|L} + \sum_{k \in hp(i)} g^{e|L}\left(\tau_k, v(k), \left\lceil \frac{t}{T_k} \right\rceil\right)$$

$$\widehat{C_{i,j}^{m|L}(t)} = C_{i,j}^{m|L} + \sum_{k \in hp(i)} g^{m|L}\left(\tau_k, v(k), \left\lceil \frac{t}{T_k} \right\rceil\right)$$

As usual, (33) is solved by recurrence. All iterations use the same value for v, i.e., the same first job for each interfering task. To compute the L-mode WCRT of frame j of task i, $R_{i,j}^L$, we consider all possible sequences of interfering tasks:

$$R_{i,j}^L = max\left\{R_{i,j}^{v|L} : v \in V_i\right\} \tag{34}$$

Finally, for the WCRT of task i in L-mode, we must consider all frames of τ_i:

$$R_i^L = max\left\{R_{i,j}^L : j = 0, ..., F_i - 1\right\} \tag{35}$$

Steady H-mode analysis. The WCRT of H-task τ_i in (steady) H-mode is computed similarly. The response time recurrence is obtained from (26), by applying the transformations that allow us to derive (33) from (23). The arguments of the stall function can be derived from the execution time terms in the RHS of the response time recurrence just like for L-mode. The WCRT of each frame of task i in H-mode is then obtained by taking the maximum of the WCRT for all sequences of interfering tasks, like in (34). Finally, the WCRT of task i is obtained by taking the maximum of the WCRT of all of its frames, as in (35).

Transient H-mode analysis. We now focus on the WCRT analysis of H-tasks caught by mode-switch. Such tasks can suffer the interference of sequences containing not only H-jobs but also L-jobs, i.e jobs that complete before the mode switch, of higher priority H-tasks. To bound this interference, we combine the cumulative function, (30), with the g^* function, (10), and define the cumulative WCET of a sequence of $\ell^L + \ell^H$ frames of task τ_i, starting with frame j, such that ℓ^L frames complete in L-mode and the remaining in H-mode:

$$g^*(\tau_i, j, \ell^L, \ell^H) = g^L(\tau_i, j, \ell^L) + g^H(\tau_i, (j + \ell^L) \bmod F_i, \ell^H) \tag{36}$$

Likewise, we define $g^{e|*}$ and $g^{m|*}$ to compute the cumulative computation and memory access time of frame sequences.

$$g^{e|*}(\tau_i, j, \ell^L, \ell^H) = g^{e|L}(\tau_i, j, \ell^L) + g^{e|H}(\tau_i, (j + \ell^L) \bmod F_i, \ell^H) \tag{37}$$

$$g^{m|*}(\tau_i, j, \ell^L, \ell^H) = g^{m|L}(\tau_i, j, \ell^L) + g^{m|H}(\tau_i, (j + \ell^L) \bmod F_i, \ell^H) \tag{38}$$

Let $R_{i,j}^{v|*}$, with $v \in V_i$, denote the worst-case response time of frame j of task τ_i caught by a mode switch when subjected to the interference of a set of frame sequences, one per higher priority tasks, such that the first frame of each of these sequences is the corresponding element in v. Then, with AMMC-max, the WCRT recurrence for a task caught by a mode switch becomes:

$$\begin{aligned} R_{i,j}^{v|*}(s) = \; & C_{i,j}^{H} + \sum_{k \in hpL(i)} g^{L}\left(\tau_k, v(k), \left\lfloor \frac{s}{T_k} \right\rfloor + 1\right) \\ & + \sum_{k \in hpH(i)} g^{*}(\tau_k, v(k), \ell^{L}(k, R_{i,j}^{v|*}(s), s), \ell^{H}(k, R_{i,j}^{v|*}(s), s)) \\ & + stall\left(C_{i,j}^{e*}(\widehat{R_{i,j}^{v|*}}(s)), C_{i,j}^{m*}(\widehat{R_{i,j}^{v|*}}(s))\right) \end{aligned} \tag{39}$$

where $\ell^{L}(k, R_{i,j}^{v|*}(s), s)$ and $\ell^{H}(k, R_{i,j}^{v|*}(s), s)$ are appropriate values for the number of interfering jobs that complete before/after the mode change, respectively. We discuss the selection of these values in this section's last paragraph.

As for other modes, the arguments of *stall* are easily derived from the RHS of (39), with the help of component-wise cumulative functions (37) and (38).

Note that (39) provides the response time of a frame j of task i when subject to the set of interfering sequences, $v \in V_i$, as a function of s. Thus:

$$R_{i,j}^{v|*} = max\{R_{i,j}^{v|*}(s) : 0 < s < R_{i,j}^{v|L}\}$$

As argued in Section 5, it suffices to compute $R_{i,j}^{v|*}(s)$ only for values of s equal to release times of higher priority L-tasks. Like for the other modes, from $R_{i,j}^{v|*}$ the response time of frame j of task i is computed by taking the maximum for all sequences of interfering tasks, and from that the response time of task i, by taking the maximum for all frames.

Quantification of $\ell^{L}(k, R_{i,j}^{v|*}(s), s)$ *and* $\ell^{H}(k, R_{i,j}^{v|*}(s), s)$. Equation (39) is to be solved via a recurrence relation, as in classic AMC-max. However, for the recurrence to converge, appropriate values for $\ell^{L}(k, R_{i,j}^{v|*}(s), s)$ and $\ell^{H}(k, R_{i,j}^{v|*}(s), s)$ must be selected. The quantification of interfering jobs completing before/after the mode change, does *not* carry over from lassic AMC-max, because it may prevent the recurrence from converging, as evidenced by the following counter-example, that uses classic AMC-max values for ℓ^{L} and ℓ^{H} (see (5) and (6)):

Consider the computation of the response time of a frame of H-task τ_i upon mode switch, using (39). Assume that the L- and H-WCET estimates of higher priority H-task τ_j are $\{20, 5, 1\}$ and $\{40, 10, 2\}$, respectively. Assume that frame 0 is the first one for τ_j. Assume that, in iteration k, there are two interfering H-jobs; then the cumulative interference is 50. Assume that in iteration $k+1$, the number of interfering jobs increases to 3, the first of which is an L-job and the othes are H-jobs, the cumulative interference will decrease to 32. As a result, the response time computed in iteration $k+1$ may be smaller than in iteration

k. Such non-monotonicity in the cumulative interference with respect to time may prevent the recurrence from converging. To address this issue we define:

$$\ell^L(k,t,s) = max\left(\left\lfloor \frac{s}{T_k} \right\rfloor - 1, 0\right) \tag{40}$$

$$\ell^H(k,t,s) = \left\lceil \frac{t}{T_k} \right\rceil - \ell^L(k,t,s) \tag{41}$$

There are three key properties of the quantification of $\ell^L(k,t,s)$ and $\ell^H(k,t,s)$ via (40) and (41) respectively that ensure that (39) can be solved using a recurrence and that its solution is safe:

Lemma 1. (40) *provides a lower bound on the number of jobs of a periodic task τ_k with period T_k that execute completely in a time interval, I, of length s.*

Proof sketch: The minimum number of job releases of τ_k in I is $\lfloor \frac{s}{T_k} \rfloor$. If the release of a job is inside I and the release of the following job is also inside I, then the former job executes completely in I. Therefore, the number of jobs that execute completely in I is one less than the number of job releases in I.

Lemma 2. (41) *is an upper bound on the number of jobs of sporadic task τ_k with minimum inter-arrival time T_k released in a time interval I, of length t, that complete after time instant s, relative to the beginning of I. Also, the sum of* (40) *and* (41) *is equal to the maximum number of job releases of τ_k in I.*

Proof sketch: The maximum number of job releases of τ_k in I is $\lceil \frac{t}{T_k} \rceil$. From Lemma 1, at most $\ell^l(k,t,s)$ of those complete before s.

The third property is that (40) is independent of t. This is trivial to check, but it is crucial as it ensures that in the solution of (39) by a recurrence, the interference of jobs of higher priority H-task τ_k before the mode switch has the same value in all iterations. Furthermore, the first H-job of the interfering job sequence of τ_k is always the same frame, and therefore the interference by the H-job sequence is monotonically non-decreasing. (As shown, for the memory agnostic case in Lemma 1 of [3].) Therefore, in the solution of the recurrence (39), $R_{i,j}^{v|*}(s)$ is monotonically non-decreasing, thus ensuring termination, either by convergence or by deadline violation.

7. Pruning

The exhaustive analysis of Section 6 is tighter than the analysis of Section 5 because it exhaustively computes the WCRT of (i) every frame of all tasks and (ii) for all possible frame sequences of interfering tasks. Zuhily and Burns [14] use *pruning* to reduce running times. We now adapt that pruning approach to our model of Section 3.

The work in [14] considered a single core platform whereas our work targets a multi-core platform with memory regulation. Therefore, we need to account for the two components of the WCET, because, as summarized in Section 4.4,

in a multicore platform with memory regulation, a job may incur a stall which depends on those components. As an extreme example, two jobs with the same WCET but one with a zero-valued memory access time, and the other with a zero-valued compute time, will incur different stalls: whereas the first job will incur no stall the latter may incur a significant stall, i.e. comparable or even higher than the job's WCET. Thus, whereas [14] relies on the $<$ relation in $\mathbb{R}$ to prune jobs, our approach relies on the relation $<$ between elements of $\mathbb{R}^2$, the WCET pairs, defined as follows:

Definition 1. *Let* $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$. *Then,* $(x_1, y_1) < (x_2, y_2)$ *iff* $(x_1 < x_2 \wedge y_1 \leq y_2 \vee x_1 \leq x_2 \wedge y_1 < y_2)$

Pruning frames: To compute the WCRT of a task, rather than consider every frame of that task, Zuhily and Burns [14] consider only the *peak frame* of each task, i.e., the one with the maximum WCET among its frames. In our model, since $\mathbb{R}^2$ is a poset under the $<$ relation defined above, there may not be a total order among the WCET pairs of a task's frames. So, for each task, we must compute the WCRT for each frame whose WCET pair is maximal among the set of WCET pairs of the frames. If multiple frames have the same maximal WCET pair, analyzing one of them suffices.

With AMC, we must compute WCRTs for H-tasks in both modes. Thus, we perform pruning independently for both modes, using the respective WCETs. I.e., for H-task τ_i, we define $\hat{F}_i^L$ and $\hat{F}_i^H$ – the sets of frames of τ_i whose WCET pairs are maximal in L- and H-mode, respectively. With these sets, to compute the L-mode WCRT of τ_i, we compute the response time for each of the frames in $\hat{F}_i^L$ and take the maximum. Similarly for the H-mode and mode switch, we consider only the frames in $\hat{F}_i^H$. For an L-task τ_i, we only define $\hat{F}_i^L$.

Pruning frame sequences: The analysis in Section 6 solves one recurrence for every element of the cartesian product of the L-sets of higher-priority tasks, i.e. the V-sets. In [14], to reduce the number of response time recurrences, Zuhily and Burns prune elements from the L sets, thus reducing the cardinality of their cartesian products and ultimately the number of recurrences.

The pruned set, $\hat{L}_i$, is obtained from set L_i by removing every frame j that fulfills the following condition: There exists a frame $k \neq j$ such that, for every sequence of ℓ frames (with $1 \leq \ell < F_i$), the cumulative WCET of the sequence starting with frame j is smaller than that of the same-length sequence that starts with frame k.

Our approach differs from that in [14] in two aspects: 1) For each task, we define set $\hat{L}^L$, furthermore for each H-task, we define also sets $\hat{L}^H$ and $\hat{L}^*$, since we must consider each mode; 2) We use the $<$ relation on $\mathbb{R}^2$, not the $<$ relation on $\mathbb{R}$, since we must ensure that the stall caused by the interfering task is also taken into account.

Thus, sets $\hat{L}_i^L$ and $\hat{L}_i^H$ are obtained from set L_i using the L-WCET and the H-WCET estimates, respectively, and by using the relation $\leq$ on $\mathbb{R}^2$, to compare the cumulative WCET pairs of sequences of ℓ jobs, with $1 \leq \ell < F_i$.

Set $\hat{L}_i^*$, used for mode switch analysis, is obtained from L_i, by removing every frame j for which there is a frame $k \neq j$ such that, for every sequence of ℓ^L

Table 2: Overview of Parameters

Parameters	Values	Default
Number of cores (K)	$\{2, 4, 6, 8\}$	2
Memory Intensity (γ)	$\{.1 : .1 : .9\}$	0.4
H-WCET scale-up factor (ξ)	$\{2 : 0.5 : 6\}$	2
Task-set size ($\lvert\tau\rvert$)	$\{6, 8, 10, 12\}$	10
Fraction of H-tasks in τ (ζ)	$\{0.05 : 0.05 : 0.6\}$	0.4
Upper bound on # of frames (α)	$\{3 : 1 : 8\}$	5
Lower bound on L-WCET (β)	$\{0.1 : 0.1 : 0.8\}$	0.2
Inter-arrival time (T_i)	$10msec$ to $1sec$	N/A

L-mode frames followed by a sequence of ℓ^H H-mode frames, with $0 \leq \ell^L \leq F_i$ and $0 < \ell^H \leq F_i$, the cumulative WCET pair of the sequence starting with frame j is $<$ than that of the sequence starting with frame k.

With the definitions above, rather than solve one response time recurrence per element of $V_i = \prod_{j=0}^{i-1} L_j$, we need only solve one recurrence per element of the Cartesian product of the appropriate "pruned" sets. For example, to compute the WCRT of an H-task τ_i upon mode switch, we solve one recurrence per element of $\prod_{j=0}^{i-1} \hat{L}_j^*$.

8. Evaluation

We implemented both analyses for performance evaluation. Our simulator has two modules. One module generates parameterizable synthetic workloads. Another one assigns bandwidth and tasks to cores, and tests for schedulability.

Unbiased task utilizations in L-mode are generated using the UUnifast-discard [36, 37] algorithm and varied within $[0.1,\ 1] \times K$. Task periods (10 msec to 1 sec) are log-uniform-distributed. Deadlines are implicit (T_i=D_i), though analyses hold for constrained deadlines ($D_i \leq T_i$). Task-set size ($\lvert\tau\rvert$) is an input parameter. The number of H-tasks is $\lfloor \zeta \times \lvert\tau\rvert \rceil$, where $\zeta \in (0,1)$ is a user defined fraction of H-tasks and $\lfloor\cdot\rceil$ denotes rounding to the nearest integer. The number of frames of each task is randomly selected over $[1,\ \alpha]$, where upper-bound α is an input parameter. The L-WCET of the first frame is generated by multiplying the period of a task with its utilization, i.e., $C_{i,1}^L = T_i * U_i$. The L-WCETs of other frames are randomly generated within $[\beta \times C_{i,1}^L,\ C_{i,1}^L]$, where $\beta \in (0,1]$ is an input parameter[2]. For any frame, its H-WCET is computed by linear scaling its L-WCET (i.e., $C_{i,j}^H = \xi \times C_{i,j}^L$), where ξ is an input parameter. (Griffin's [38] more sophisticated H-WCET generation approach came too recently for us to consider.) The total WCET is divided between memory access time and the CPU computation time using an input parameter γ. The memory access time of a frame in mode $x \in \{L, H\}$ is randomly selected within $[0,\ \gamma \times C_{i,j}^x]$; what remains is the CPU computation time, i.e., $C_{i,j}^{e|x} = C_{i,j}^x - C_{i,j}^{m|x}$. Audsley's priority

[2]For convenience, without loss of generality (since shift-rotating the order of the frames results in an equivalent multiframe task), the first frame has the biggest L-WCET. Note that our analyses make no assumptions regarding the WCET of the first frame being the greatest.

assignment [39] is used. For frame-agnostic analyses, we use for each task τ_i the parameters $g^{e|\kappa}(\tau_i, 1)$, $g^{m|\kappa}(\tau_i, 1)$ and $g^{e|\kappa}(\tau_i, 1) + g^{m|\kappa}(\tau_i, 1)$ for its worst-case CPU computation time, worst-case memory access time and total WCET.

Table 2 summarizes the parameters. The triples represent minimum value, increment size and maximum value for a parameter. Each random parameter uses its own pseudo random number generator, which is seeded in the first replication [40].

For every parameter in this table, except the inter-arrival time, we run a two-factor experiment in which one of the factors is that parameter and the other factor is the task set L-mode utilization. In each of these experiments, the remaining parameters take their default values.

For each combination of input parameters, 100 task sets are generated. As in [21], a memory access takes 40 nsec and the regulation period is 100 μsec. Task-to-core assignment and bandwidth allocation are performed using memory-fit [11]. This heuristic assigns a task to the core that requires the least memory budget increase to accommodate it, as quantified via binary search.

Because we use a two-factor experimental design, we use weighted schedulability [41, 42] as evaluation metric, making it possible to show the results in two dimensional plots rather than in three dimensional plots.

Let $S_y(\tau, p)$ denote the result of schedulability test y for task set τ for an input parameter p. The weighted schedulability $W_y(p)$ is:

$$W_y(p) = \frac{\sum_{\forall \tau}(\bar{U}^L(\tau) \cdot S_y(\tau, p))}{\sum_{\forall \tau} \bar{U}^L(\tau)} \tag{42}$$

where, $\bar{U^L}(\tau) = \frac{U^L(\tau)}{K}$ is L-mode system utilization normalized by number of cores K. The experiments ran on a system with 8 Xeon E5-2420 2.20 GHz dual cores, 32 GB of RAM and Linux Mint 18.3 Sylvia with open jdk 1.8.0_242.

8.1. Results

In our experiments, the simpler multiframe analysis (AMMC-max, Section 5) markedly outperforms the frame-agnostic AMC-max, AMC-rtb and SMC, and, surprisingly, almost matches the exhaustive analysis (AMMC-max-Z, Section 6).

The effect of the H-WCET scale-up factor (ξ) and the H-task fraction (ζ) are presented in Figures 2a and 2b. High values for either, increase the processing requirements of H-tasks, hence lowering the weighted schedulability (and performance difference among tests). The number of generated H-tasks is rounded up to the nearest integer when the target ζ yields a non-integer number. This explains the step-like behavior in Figure 2b. A higher memory intensity increases the overall stall, detrimentally for schedulability (Figure 2c). With more cores, the overall interference among cores increases, impacting the schedulability for all tests (Figure 2d). It is expected, since the memory bandwidth is not scaled up with the increase in the number of cores. The higher interference also subsumes the benefits of better analysis.

Many low-utilization tasks are easier to schedule than few high-utilization tasks, due to bin-packing fragmentation. Hence, the performance of all tests

Table 3: Maximum absolute difference in unweighted schedulability success ratio compared to AMC-max over all experiments for each parameter varied.

Analysis	ζ	$\lvert\tau\rvert$	ξ	α	β	γ	K
AMMC-max-Z	39%	45%	69%	54%	47%	**72%**	39%
AMMC-max	37%	41%	**66%**	50%	44%	62%	37%

improves with task set size (Figure 2e). With more tasks, there is also more potential to leverage frame information, hence the greater performance difference between frame-agnostic and multiframe analyses with large task set sizes.

Parameter β lower-bounds the ratio between the smallest and the biggest L-WCET, among a task's frames. As β tends to 1, the generated task sets approximate single-frame tasks, thereby reducing the potential benefits from leveraging the frame information. This effect is evident for all tests (Figure 2f). Varying the number of frames shows a roughly ascending trend for the multiframe analyses (Figure 2g). Indeed, having more frames per task means a higher percentage of frames not exhibiting the worst-case computation and/or memory access time – which multiframe analyses can leverage. The trend is opposite for the frame-agnostic approaches, because the single-frame representation of a multiframe task combines the worst-case computation and the worst-case memory access time over all of the latter's frames (possibly, different ones). The pessimism from that is more pronounced, the higher the number of frames is.

The test running time of the simpler multiframe analysis (AMMC-max) is of the same order of magnitude as that of the frame agnostic analyses, whereas the exhaustive analysis (AMMC-max-Z) is several orders of magnitude higher(Figure 2h).

Finally, Table 3 provides the maximum absolute difference in *unweighted* schedulability success ratio of each multiframe max analysis (AMMC-max-z and AMMC-max) from the single frame analysis (AMC-max), when varying different parameters. The maximum difference for each analysis is highlighted in Table 3. In the best case, AMMC-max-Z achieves up to 72% higher success ratio than AMC-max. The results for the simpler analysis is close to those of the exhaustive analysis, whereas its running time is much shorter.

9. Conclusions

We formulated schedulability analysis for multiframe mixed-criticality tasks with memory regulation, that incorporates the related memory stalls. The state-of-the-art previously only offered analysis for systems with at most any two of the following three characteristics: (i) mixed criticalities; (ii) multiframe tasks; and (iii) memory access regulation. By covering all three of those our work allows for safely leveraging frame information of mixed-criticality tasks, for better schedulability, also on COTS multicore architectures; no longer just single cores. As our experiments with synthetic tasks demonstrated, the frame-aware and stall-aware schedulability analysis substantially outperforms its frame-agnostic

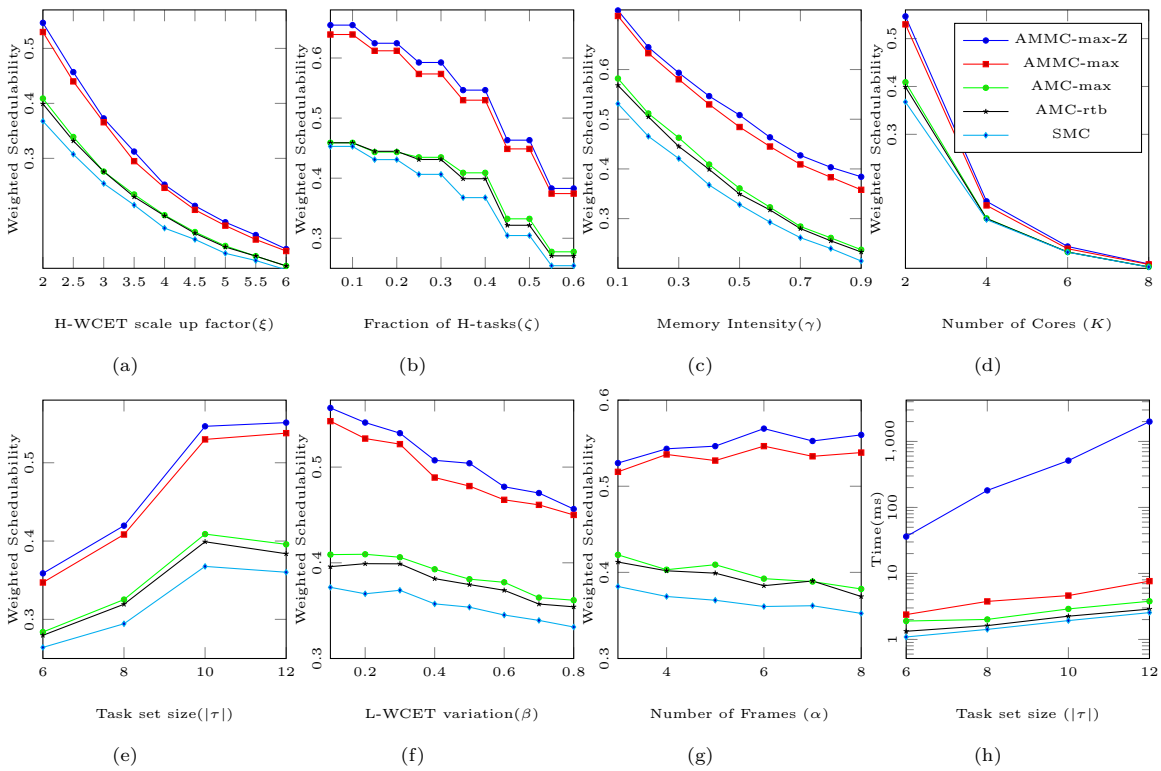

Figure 2: Weighted schedulability for different factors variation a) Effect of H-tasks share in a taskset b) Effect of H-WCET variation c) Effect of number of cores d) Effect of memory intensity e) Effect of tskset set size f) Lower bound on L-WCET variation g) Effect of number of frames h) Average runing time

(but still stall-aware) counterparts. This validates our approach as a tool for safely harnessing the power of COTS multicores for critical workloads.

Acknowledgements

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDP/UIDB/04234/2020); by the Operational Competitiveness Programme and Internationalization (COMPETE 2020) under the PT2020 Partnership Agreement, through the European Regional Development Fund (ERDF), and by national funds through the FCT, within project PREFECT (POCI-01-0145-FEDER-029119); by FCT through the European Social Fund (ESF) and the Regional Operational Programme (ROP) Norte 2020, under grant 2020.08045.BD.

References

[1] S. Baruah, A. Burns, Implementing mixed criticality systems in Ada, in: 16th Ada-Europe Conference, 2011, pp. 174–188.

[2] A. K. Mok, D. Chen, A multiframe model for real-time tasks, Trans. Softw. Engin. 23 (10) (1997) 635–645.

[3] I. Hussain, M. A. Awan, P. F. Souto, K. Bletsas, B. Akesson, E. Tovar, Response time analysis of multiframe mixed-criticality systems with arbitrary deadlines, J. Real–Time Syst. (2020). doi:10.1007/s11241-020-09357-w.

[4] D. Le Gall, Mpeg: A video compression standard for multimedia applications, CACM 34 (4) (1991) 46–58. doi:10.1145/103085.103090.

[5] Aeronautical radio, Inc., ARINC specification 818-2 Avionics Digital Video Bus (ADVB) High Data Rate, 818th Edition (2013).

[6] C. Bailey, A. Burns, A. Wellings, C. Forsyth, Keynote paper: A performance analysis of a hard real-time system, Control Engineering Practice 3 (4) (1995) 447–464.

[7] J. P. Lehoczky, L. Sha, Y. Ding, The rate monotonic scheduling algorithm: Exact characterization and average case behavior, in: 10th RTSS, 1989, pp. 166–171.

[8] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, L. Sha, Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms, in: 19th RTAS, 2013, pp. 55–64. doi:10.1109/RTAS.2013.6531079.

[9] R. Mancuso, R. Pellizzoni, M. Caccamo, L. Sha, H. Yun, Wcet (m) estimation in multi-core systems using single core equivalence, in: 27th ECRTS, IEEE, 2015, pp. 174–183.

[10] M. A. Awan, P. Souto, K. Bletsas, B. Akesson, E. Tovar, Mixed-criticality scheduling with memory bandwidth regulation, in: 55th DATE, 2018.

[11] M. A. Awan, P. F. Souto, K. Bletsas, B. Akesson, E. Tovar, Memory bandwidth regulation for multiframe task sets, in: 25th RTCSA, 2019, pp. 1–11.

[12] S. K. Baruah, A. Mok, Static-priority scheduling of multiframe tasks, in: 11th ECRTS, 1999, pp. 38–45.

[13] S. Baruah, D. Chen, S. Gorinsky, A. Mok, Generalized multiframe tasks, J. Real–Time Syst. 17 (1) (1999) 5–22.

[14] A. Zuhily, A. Burns, Exact scheduling analysis of non-accumulatively monotonic multiframe tasks, J. Real–Time Syst. 43 (2) (2009) 119–146.

[15] A. Burns, R. I. Davis, A survey of research into mixed criticality systems, Comput. Surveys 50 (6) (2017) 82:1–82:37.

[16] S. K. Baruah, A. Burns, R. I. Davis, Response-time analysis for mixed criticality systems, in: 32nd RTSS, 2011, pp. 34–43.

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

[18] A. Burns, R. I. Davis, Response time analysis for mixed criticality systems with arbitrary deadlines, in: 5th WMC, York, 2017.

[19] T. Fleming, A. Burns, Extending mixed criticality scheduling, in: Proc. WMC, RTSS, 2013, pp. 7–12.

[20] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, R. Rajkumar, Bounding memory interference delay in COTS-based multi-core systems, in: 20th RTAS, 2014, pp. 145–154.

[21] G. Yao, H. Yun, Z. P. Wu, R. Pellizzoni, M. Caccamo, L. Sha, Schedulability analysis for memory bandwidth regulated multicore real-time systems, IEEE Transactions on Computers 65 (2) (2016) 601–614. doi:10.1109/TC.2015.2425874.

[22] R. Pellizzoni, H. Yun, Memory servers for multicore systems, in: 22nd RTAS, 2016, pp. 97–108.

[23] K. Lampka, G. Giannopoulou, R. Pellizzoni, Z. Wu, N. Stoimenov, A formal approach to the WCRT analysis of multicore systems with memory contention under phase-structured task sets, J. Real–Time Syst. 50 (5) (2014) 736–773.

[24] R. Mancuso, R. Pellizzoni, N. Tokcan, M. Caccamo, WCET Derivation under Single Core Equivalence with Explicit Memory Budget Assignment, in: 29th ECRTS, Vol. 76 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017, pp. 3:1–3:23.

[25] A. Agrawal, G. Fohler, J. Freitag, J. Nowotsch, S. Uhrig, M. Paulitsch, Contention-aware dynamic memory bandwidth isolation with predictability in COTS multicores: An avionics case study, in: 29th ECRTS, 2017, pp. 2:1–2:22.

[26] M. A. Awan, P. F. Souto, B. Akesson, K. Bletsas, E. Tovar, Uneven memory regulation for scheduling IMA applications on multi-core platforms, J. Real–Time Syst. 55 (2) (2019) 248–292. doi:10.1007/s11241-018-9322-y.

[27] L. Sha, M. Caccamo, R. Mancuso, J.-E. Kim, M.-K. Yoon, R. Pellizzoni, H. Yun, R. Kegley, D. Perlman, G. Arundale, R. Bradford, Single core equivalent virtual machines for hard real—time computing on multicore processors, Tech. rep., University of Illinois (2014).

[28] M. A. Awan, K. Bletsas, P. F. Souto, B. Akesson, E. Tovar, Mixed-criticality scheduling with dynamic memory bandwidth regulation, in: 24th RTCSA, 2018.

[29] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, L. Sha, Memory access control in multiprocessor for real-time systems with mixed criticality, in: 24th ECRTS, 2012, pp. 299–308.

[30] A. Kritikakou, O. Baldellon, C. Pagetti, C. Rochange, M. Roy, F. Vargas, Monitoring On-line Timing Information to Support Mixed-Critical Workloads, in: 34rd RTSS, Vancouver, Canada, 2013, pp. 9–10.
URL `https://hal.archives-ouvertes.fr/hal-01015455`

[31] A. Kritikakou, C. Rochange, M. Faugère, C. Pagetti, M. Roy, S. Girbal, D. G. Pérez, Distributed run-time wcet controller for concurrent critical tasks in mixed-critical systems, in: 22nd RTNS, ACM, 2014, p. 139.

[32] A. Kritikakou, T. Marty, M. Roy, Dynascore: Dynamic software controller to increase resource utilization in mixed-critical systems, TODAES 23 (2) (2018) 13.

[33] I. Hussain, M. A. Awan, P. F. Souto, K. Bletsas, B. Akesson, E. Tovar, Response time analysis of multiframe mixed-criticality systems, in: 27th RTNS, 2019, pp. 8–18.

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

[35] S. K. Baruah, H. Li, L. Stougie, Mixed-criticality scheduling: Improved resource-augmentation results., in: 25th Int. Conf. Computers & Their Applic.(CATA), 2010, pp. 217–223.

[36] E. Bini, G. C. Buttazzo, Measuring the performance of schedulability tests, J. Real–Time Syst. 30 (1-2) (2005) 129–154.

[37] R. I. Davis, A. Burns, Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems, in: 30th RTSS, 2009, pp. 398–409.

[38] D. Griffin, I. Bate, R. I. Davis, Generating utilization vectors for the systematic evaluation of schedulability tests, in: 41st RTSS, IEEE, 2020, pp. 76–88.

[39] N. C. Audsley, On priority assignment in fixed priority scheduling, Inform. Processing Lett. 79 (1) (2001) 39–44.

[40] R. Jain, The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling., Wiley professional computing, Wiley, 1991.

[41] A. Bastoni, B. Brandenburg, J. Anderson, Cache-related preemption and migration delays: Empirical approximation and impact on schedulability, 6th OSPERT 10 (2010) 33–44.

[42] A. Burns, R. Davis, Adaptive mixed criticality scheduling with deferred preemption, in: 35rd RTSS, 2014, pp. 21–30.