What if we would degrade LO tasks in mixed-criticality systems?

Marcus Völп
School of Computer Science, Logical Systems Lab
Carnegie Mellon University
Pittsburgh, PA, USA
mvoelp@cs.cmu.edu

I. INTRODUCTION

Mixed-criticality (MC) systems [1] allow tasks of different importance (or criticality) to be consolidated into a single system. Consolidation facilitates resource sharing (even across criticality levels) and hence bears the potential to reduce the overall amount of resources needed. However, there is a common misconception that recurs in literature about Vestal’s model: the false believe that low criticality tasks are degraded to soft real-time or even best effort tasks. In this work, we not only wish to clarify this misconception but also ask ourselves what would happen if we degrade LO tasks. Revisiting Quality Assurance Scheduling (QAS) [2], [3], our goals are stochastic guarantees for LO completion in addition to and replacing the hard MC guarantees if LO tasks are soft real-time. In this WIP report, we focus on properties of dropped LO tasks (while keeping hard MC guarantees) such as: “What is the likelihood of lower criticality jobs being dropped because higher criticality jobs exceed their low WCET estimates?”, “What is the likelihood of dropped jobs to still make their deadline?”, and “What is the expected time / Q-percentile for dropped jobs to catch up with their execution?” Part of our future work will be to extend these guarantees and to develop MC schedulers for a combination of hard and soft real-time tasks. Notice though, that the assumptions in this report still limit the applicability of our results. We indicate how we plan to relax them in the future. Most notably, we assume that jobs arrive in their synchronous arrival sequence and that execution-time distributions are known. The latter we plan to replace with a higher criticality task with \( l_h > l_i \) exceeds its low WCET estimate \( C_h(l_i) \).

There are two important points to notice:

1) If no higher-criticality job \( J_h \) exceeds its low estimate \( C_h(l_i) \), lower criticality jobs \( J_i \) receive the same hard real-time guarantees as in a classical system. That is, provided all WCET estimates are safe, they receive sufficient time to complete before their deadlines; and

2) No guarantee is given to these jobs once a higher criticality job exceeds its low WCET estimate. Notice also that dropped jobs merely lose their real-time guarantees (and possibly their high prioritized budget). There is no necessity to terminate these jobs. The question about low-criticality guarantees now boils down to whether WCET estimates are safe or whether in some rare situations the actual execution time may exceed these estimates. Either way, MC schedulers convey no guarantee about the subset of deadlines low criticality tasks meet, which is essential for soft real-time.

II. MIXED-CRITICALITY SCHEDULING

Let \( T \) be a set of sporadic tasks. As usual, we characterize \( \tau_i \in T \) by tuples \((l_i, D_i, T_i, C_i)\) where \( l_i \) is a criticality level in the ordered set of criticality levels \( L \) (e.g., \( L = \{LO, HI\} \) with \( LO < HI \)), \( D_i \leq T_i \) is the relative deadline and \( T_i \) the minimal interrelease time. We subject tasks to execution time analyses suitable for the individual levels. The result is a vector of increasingly more pessimistic WCET estimates \( C_i(l) \). The set \( L \) and the requirements for considering an analysis suitable may be drawn from evaluation criteria such as DO-178C [4] but other metrics are also conceivable. We assume the system enforces budgets \( C_i(l_i) \) and subject \( \tau_i \) only to the WCET analyses for all \( l \leq l_i \). The feasibility criteria and hence the guarantee given to all admitted tasks is:

\[ \text{Definition 1 (Feasibility):} \text{ The set } T \text{ is feasible if every job } J_i = \tau_{i,t} \text{ receives } C_i(l_i) \text{ time in between its release time } r_i \text{ and its absolute deadline } r_i + D_i \text{ provided no job } J_h \text{ of higher criticality task with } l_h > l_i \text{ exceeds its low WCET estimate } C_h(l_i). \]

QAS [2] offers stochastic guarantees for imprecise computations [5] where jobs are composed of mandatory parts, which must execute to completion, and optional parts, which improve the final result. For the first, a safe WCET estimate \( P[X_i \leq C_i] = 1 \) is anticipated whereas for the latter QAS assigns budgets \( b_i^k \) resulting from requested \( Q_i \)-percentiles of the execution time distribution \( P[X_i \leq b_i^k] = Q_i \). Here and in the following \( X_i \) and \( Y_i^k \) denote non-negative random variables capturing actual execution times and \( P[X_i \leq c] \) stands for the probability that \( X_i \leq c \). QAS aborts parts at their assigned budgets but considers the actual execution time distribution to determine the likelihood of lower prioritized jobs meeting their deadlines. The \( k^{th} \) optional part \( o_i^k \) is executed (possibly at a different priority) only if all prior optional parts completed in time. For the special case where tasks shares a common release time \( r \) and deadline \( D \), the job \( J_i \) completes \( o_i^k \) with probability \( p \) if

\[ P[Y_i^k < b_i^k \wedge \left( \sum_{j \in \text{hp}_i} X_j + \sum_{i} \min(Y_i^m, b_i^m) \right) < D] = p \]

IV. OPTIONAL PARTS MEET MIXED CRITICALITY

Our goal is to translate MC scheduling into QAS to (1) reuse the feasibility and abortion results and (2) devise new algorithms to convey stochastic guarantees for LO jobs.
a) Likelihood to drop low criticality jobs: Fig. 1 shows an example of a standard (i.e., not QAS) adaptive MC algorithm. The algorithm is adaptive in that \( \tau_L \) must be dropped (i.e., the priority of \( \tau_L \) must change) to guarantee the completion of the two HI tasks in case \( J_1 \) exceeds \( C_{H}(LO) \). In other words, \( X_1 < C_{H}(LO) \) must hold for \( J_2 \) to not be dropped and in addition \( X_3 < C_{H}(LO) \) and \( X_4 < C_{H}(LO) \) to start executing \( J_5 \). But these are exactly the conditions for the execution of optional parts. By regarding the LO parts of HI jobs whose criticality decision point is before the worst-case response time \( R_{LO}^{1} \) of a job \( J_k \) as preceding optional parts of this job, QAS gives us the likelihood of this job being dropped as:

\[
1 - \prod_{\tau \in \mathcal{T} \mid R_{LO}^{1} \leq R_{LO}^{k}} P[X_1 < C_{i}(LO)]
\]

where \( R_{LO}^{i} = C_{i}(LO) + \sum_{\tau \in \mathcal{H}^{i}} T_{\tau} \) is the worst-case response time of \( J_i \) assuming all higher prioritized job \( J_j \) require no more than \( C_{j}(LO) \). Fig. 1 indicates this dropping dependency as dashed lines. The criticality decision point of a job \( J_k \) is the worst-case response time of its low part. At this point, we know whether \( J_k \) causes LO tasks to be dropped.

b) Likelihood of dropped jobs meeting their deadlines: The bottom part of Fig. 1 shows that it is possible to complete the dropped job \( J_2 \) at a priority level where it can no longer defer the execution of \( J_3 \), which could have caused a deadline miss. \( J_2 \) completes in time if after \( J_4 \) has exceeded \( C_{H}(LO) \) the combined execution time \( X_1 + X_2 + X_3 \) is less than or equal to \( D_L \). \( J_2 \) may even complete if \( C_{L}(LO) \) is an unsafe WCET estimate, for example if \( J_3 \) stops before time \( t_1 \) and if additional time is given to \( J_2 \) after \( t_2 \). More generally, if a HI job \( J_k \) does not complete, it receives an additional part with \( Y_k = \max(X_k - C_{k}(LO), 0) \) and the originally scheduled parts of \( LO \) jobs \( J_3 \) in dropping dependency with \( J_k \) are aborted. Instead of not executing dropped jobs \( J_1 \), we assume they receive a possibly larger budget at a priority level that is sufficiently low to not risk high completion. Priorities in a strictly lower band fulfill this condition, however, we expect to find less pessimistic setups in the future. The important constraint (in particular when considering more than two criticality levels) is to preserve the relative priority ordering of dropped jobs because then MC guarantees extend to the stochastic MC guarantees for dropped jobs. That is, dropped jobs with a higher criticality level than other dropped jobs complete more likely. As a preliminary result, the likelihood that a dropped job \( J_i \) meets its deadline under the condition that \( \pi(J_1) > \ldots > \pi(J_{i,1}) \) denotes the priority ordering of higher than \( J_i \) prioritized jobs after jobs have been dropped is:

\[
P[T_k = \max(T_{k-1} - r_k) + X_k + D_k, \quad \text{max}(T_{k-2} - r_k) + D_k]
\]

where \( T_k = \max(T_{k-1} - r_k) + X_k + D_k \) if \( \max(T_{k-1} - r_k) + X_k + D_k < X_k - r_k + D_k \) and \( \max(T_{k-2} - r_k) + D_k \) otherwise.

V. RELATED WORK

To our best knowledge, this is the first attempt to cast MC scheduling into an imprecise computation context. There is of course a large body of work on probabilistic analyses and scheduling of non-MC systems. Alahmad et al. [6] investigate probabilistic execution-behavior models and identify stochastic MC scheduling as an open problem [7]. In contrast to our work, they seek to optimize the feasibility of the MC schedule itself, not only of dropped tasks. Also they do not consider the possibility of unsafe WCET estimates for \( LO \) jobs.

VI. CONCLUSIONS

We present first results connecting quality assuring and mixed-criticality scheduling to give stochastic guarantees for dropped jobs.

REFERENCES


\[1\] Notice, there is no need to make this decision earlier because the completion of no high task is at risk.

\[2\] This work is partially funded through NSF Grant CNS-0931985 and by the DFG through the cluster of excellence Center for Advancing Electronics Dresden.