Run-time Check of AMC-rtb Analysis

First Author Second Author Third Author

***Abstract*—This document summarizes the checks.**

I. Introduction

The use of machine-learning (ML) for scheduling of hard real-time systems is challenging. Indeed, the main, or may be the only, kind of systems that use hard real-time scheduling are safety-critical systems, and these systems are subject to strict certification requirements, including timeliness guarantees. However, current ML algorithms cannot provide such guarantees.

Thus our approach is different. We use ML algorithms to adjust, at run-time, parameters of scheduling algorithms amenable to certification, in order to improve their QoS, while ensuring that such adjustments do not break the schedulability analysis developed for certification.

As an example of the application of this approach we consider the Adaptive Mixed Criticatility - response time bound (AMC-rtb) algorithm.

II. AMC-rtb

This algorithm assumes a model that is an extension of the Vestal model. Like in the original model, it assumes that each task belongs to one of two classes: LO and HI, with different degrees of "importance". However, only HI-tasks have two WCET estimates. I.e. each HI-task has one conservative WCET, HI-WCET, which we assume that is an upper-bound of the execution time of every job of that task, and a more agressive, smaller, WCET, LO-WCET, which we assume that holds for most jobs, but that can be exceeded only rarely. On the other hand, LO-tasks have only one WCET, LO-WCET, which can be exceeded occasionally as well. Furthermore, the system operates in two-modes: LO and HI. In LO-mode the system schedules all tasks for running, whereas in HI-mode the system executes only HI-tasks, i.e. LO-tasks are suspended. The system starts in LO-mode, and switches to HI-mode when the execution time of a job exceeds its WCET-L. The system switches back to HI-mode when the CPU is idle.

The purpose of this model is to increase the utilization of computing resources. Indeed, by using different modes of execution, each of which with a different set of tasks, the algorithm allows to execute both LO- and HI-tasks in LO-mode, while ensuring, by an appropriate schedulability analysis, that all jobs of all HI-tasks never exceed their deadlines, even if in LO-mode the job of some task exceeds its LO-WCET.

Note that in the original AMC-rtb paper the mode switch occurs when any task, HI- or LO-, exceeds its LO-WCET. In our model, we advocate aborting LO-tasks that exceed their LO-WCET, but not switching to HI-mode. The reason is that this action affects only the offending task, not other LO-tasks that have not exceeded their LO-WCET. Furthermore, this action affects only the current job of the offending task, rather than potentially many jobs (until the system switches back to LO-mode).

Regardless, both aborting LO-tasks or switching to HI-mode may affect the QoS of the system. So, we would like to avoid those actions as much as possible. The approach we propose is to use a deep reinforcement learning (DRL) agent to adapt the parameters used by AMC-rtb so as to make those events as rare as possible. More specifically, our goal is to change at run-time the LO-WCET of the different tasks. However, in LO-mode, when the current LO-WCET of a LO-task is exceeded the offending job will be aborted, and when the LO-WCET of a HI-task is exceeded there is a mode switch and all LO-tasks are suspended. By tuning the different LO-WCET estimates we expect to be able to reduce the frequency of the events that reduce the system QoS.

Now the LO-WCET of all tasks is used by the schedulability analysis. So, we need to make sure that whatever changes are applied do not invalidate that analysis.

A. AMC-rtb Schedulability Tests

AMC-rtb is a fixed-priority scheduling algorithm. I.e. each task is assigned a priority that does not change over time. Therefore its schedulability tests rely on response time analysis.

In LO-mode, we can apply the standard response time analysis. I.e. the response time of task τ_i, R_i^{LO}, is given by:

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

where C_i^{LO} is the LO-WCET of task τ_i, T_j is the minimum inter-arrival time, or period, of task τ_j, and $hp(i)$ is the set of all tasks (regardless of their type) with higher priority than τ_i.

Thus the task set is schedulable in LO-mode, if for every task τ_i, $R_i^{LO} \leq D_i$.

We can use a similar recurrence for computing R_i^{HI} in HI-mode, for HI-tasks. However this analysis is subsumed by the analysis of HI-tasks upon mode switch. In AMC-rtb, the response time of HI-task τ_i upon mode switch R_i^* is given by the following recurrence:

$$R_i^* = C_i^{HI} + \sum_{j \in hpH(i)} \left\lceil \frac{R_i^*}{T_j} \right\rceil C_j^{HI} + \sum_{j \in hpL(i)} \left\lceil \frac{R_i^{LO}}{T_j} \right\rceil C_j^{LO} \quad (2)$$

where C_i^{HI} is the HI-WCET of HI-task τ_i, and $hpH(i)$ is the set of HI-tasks with higher priority than τ_i and $hpL(i)$ is the set of LO-tasks with higher priority that τ_i.

Again, the task set is schedulable in switch mode, if for every HI-task τ_i, $R_i^* \leq D_i$.

To argue the safety of (2) we observe that the RHS of (2) assumes that: 1) the job of the task under analysis, τ_i executes for its HI-WCET, which we assume is never exceeded; 2) that all jobs of each higher-priority HI-task that may arrive while τ_i executes, i.e. R_i^*, execute for the respective HI-WCET; since $C_l^{LO} \leq C_i^{HI}$ this is clearly an upper bound of the interference by HI-tasks; 3) all jobs of each higher-priority LO-task that may arrive until the latest possible instant of a mode switch that may affect a job of τ_i, R_i^{LO}, execute for the respective LO-WCET.

Note that a job cannot be executed in LO-mode for longer than its R_i^{LO}. By the LO-mode analysis, (1), we know that, if every job of every higher priority task executes for at most its LO-WCET, then the response time of a job of task τ_i is given by R_i^{LO}. Thus, if a job executes for longer than R_i^{LO}, then some higher-priority job must have executed for longer than its LO-WCET, in which case there will be a mode switch.

In our variation of the original AMC algorithm, we ensure that no job of any LO-task ever executes for longer than its LO-WCET by terminating that job if it executes for more than LO-WCET. Thus, only jobs of HI-tasks can execute for longer than their LO-WCET, but this event will trigger a mode switch.

B. Validity of the Analysis

As mentioned above, the idea is to adjust the LO-WCET of the different tasks, without invalidating the analysis of HI-tasks. We might also consider the invalidation of LO-tasks, but this is likely to lead to too many invalid corrective actions, determined by our DRL agent.

By looking at (2), it is clear that the response time in mode switch, R_i^*, depends on LO-WCET of LO-tasks in $hpL(i)$, but it also depends on R_i^{LO}. From (1), we conclude that R_i^{LO} depends on the LO-WCET of all tasks in $hp(i)$.

Thus we must ensure that:

$$C_i'^{LO} + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{LO}}{T_j} \right\rceil C_j'^{LO} \leq R_i^{LO} \quad (3)$$

where $C_j'^{LO}$ is the LO-WCET determined by the DRL agent.

Note that not only we do not recompute R_i^{LO}, because it would require to solve a recurrence, but we also do not recompute the ceilings in the LHS summation of (1). This makes the checking less time consuming.

Finally we must ensure that the RHS of (2) using the LO-mode WCETs computed by the DRL agent do not exceed the R_i^* computed in the analysis:

$$C_i^{HI} + \sum_{j \in hpH(i)} \left\lceil \frac{R_i^*}{T_j} \right\rceil C_j^{HI} + \sum_{j \in hpL(i)} \left\lceil \frac{R_i^{LO}}{T_j} \right\rceil C_j'^{LO} \leq R_i^* \quad (4)$$

Note that the first two terms of the RHS can be computed at design time. Furthermore, the ceilings in the second summation can also be computed at design time. Therefore, checking inequality (4) should be much faster than solving a recurrence.

Actually, rather than inequality (4), we can use the inequality:

$$C_i^{HI} + \sum_{j \in hpH(i)} \left\lceil \frac{D_i}{T_j} \right\rceil C_j^{HI} + \sum_{j \in hpL(i)} \left\lceil \frac{R_i^{LO}}{T_j} \right\rceil C_j'^{LO} \leq D_i \quad (5)$$

I.e. rather than checking that the response time at mode switch is lower than R_i^*, we check that it is lower than D_i. It is not clear which inequality leads to better results. True that $D_i \geq R_i^*$, so the upper-bound is greater, but the interference by higher priority HI-tasks, i.e. the second term on the RHS, may also increase. We can also check (4) first, and if it fails, then try (5). If any of these checks succeed, then we can safely use the newly computed LO-WCETs.

Note that for this to be safe, the newly computed LO-WCETs can be changed only when the system is **idle**, i.e. when there is no job running in the system. Otherwise, it will raise some issues. E.g., assume that for some τ_j we decrease its C_j^{LO} and increase the C_k^{LO} of some other task. It is not clear that the increase in the LO-WCET of τ_k will be compensated by the decrease of the LO-WCET of τ_j, because, in the current busy period of some task τ_i with lower priority, these tasks may have already executed some jobs using different LO-WCETs.

This restriction may have some implications in the responsiveness of the agent, and ultimately on the success of this approach. If the system is highly loaded, the DRL agent may be able to intervene only upon the switch of HI- to LO-mode, which occurs only when the system is idle, for reasons similar to the ones that prevent us from changing the LO-WCET while there is some job running.

This suggests that in this case, it may be better to use the approach proposed in the original AMC paper: to switch to HI-mode upon exceedance of the LO-WCET, regardless of the class of the task of the job that exceeded that bound, i.e. upon exceedance of the LO-WCET not only of HI-jobs but also of LO-jobs. It would be interesting to run some simulations to evaluate this.

III. Open Issues

A. Penalties for DRL

As we have discussed the goal of the DRL agent is to prevent the occurrence of events that may degrade the QoS of the system: 1) abnormal termination of LO-jobs, caused by the exceedance of their LO-WCET 2) switches from LO- to

HI-mode, caused by the exceedance of the LO-WCET by a HI-job

Thus, we will have to assign some penalty to each of these events. A simple approach would be to give a fixed penalty, say of 1, for each task affected by these events. Thus 1) would lead to a penalty of 1, whereas 2) would have a penalty of $|\Gamma^{LO}|$, where Γ^{LO} is the set of LO-tasks.

An enhanced version of this approach is to consider the number of jobs affected by the event rather than the number of tasks. Again, using a penalty per job of 1, event 1) would lead to a penalty of 1, whereas event 2) would have a variable penalty, depending on the number of LO-jobs that arrive during HI-mode. Since the DRL-agent does not execute in HI-mode, we can wait for the mode switch from HI- to LO-mode, to compute the penalty. The main issue I have with this approach is that the penalty of a given job exceeding its LO-WCET may depend on the instant when that event occurs. Anyway, let's hope that the agent will be able to learn how to best proceed.

B. Task-set generation

Find Hamman's (?) paper. (Recall that he is Dakshina's boss.)

IV. System Model

System Model

A. Literature Survey

In 2007, Vestal [**?**] introduced the mixed-criticality notion of the classical real-time systems to the research community.