Increasing Fixed-Priority Schedulability using Non-Periodic Load Shapers

Mitra Nasri
Max Planck Institute for Software Systems (MPI-SWS),
mitra@mpi-sws.org

Geoffrey Nelissen
CISTER-INESC TEC, ISEP, Portugal.
grrpn@isep.ipp.pt

I. Motivations

Many real-time systems use *fixed-priority scheduling* (FPS) for three main reasons: (i) it is easy to understand, configure and analyse, (ii) it has low runtime overheads, and (iii) it is widely implemented in commercial operating systems. Additionally, FPS is enforced by some industrial safety related standards such as Autosar. However, FPS is not optimal from a schedulability viewpoint as it may prioritise the execution of non-urgent high-priority tasks over tasks with urgent deadlines.

As an attempt to increase FPS schedulability, variations of the FPS scheduling scheme such as Preemption Threshold Scheduling (PTS) [1] and Dual-Priroity (DP) scheduling [2] have been introduced. PTS raises the priority of a task (possibly several times) based on its remaining execution time, while DP promotes the task's priority after a given time following its release. It was shown that PTS is not optimal [3] and designing a strategy for correctly assigning priorities and promotion times such that all deadlines are met is still an open problem for DP [4].

In this work, we introduce *load-shaping* (LS); a technique that shapes the workload of tasks to limit their impact on the system's schedulability while improving the system's safety. LS is implemented using a reservation server for each task. Each server has a budget that replenishes itself following a given pattern.

Example. Fig. 1-(a) shows an example of a periodic task set which is not schedulable with FPS. However, the system may become schedulable if the workload released by τ_1 is properly shaped. An example of such shaping is shown in Fig. 1-(b), where τ_1's execution is divided in two segments. τ_1 is allowed to execute 4 time units in the first segment and 8 time units in the second.

Note that load shaping is more generic then a simple period transformation [5]. Period transformation divides the tasks' periods into smaller segments of equal lengths, while load shaping may generate segments of different lengths (see example above). We claim that period transformation may result in having more segments than actually required for the system to become schedulable. For instance, for Fig. 1's example, period transformation would need to reduce τ_1's period to 10 (i.e., the greatest-common-divisor of the tasks' periods), hence, dividing τ_1 in 3 segments instead of 2 as with load-shaping.

II. System Model and Notations

In this work, we consider a uniprocessor system executing a periodic task set τ. Each task τ_i in τ is characterised by a period T_i, a worst-case execution time (WCET) C_i, a relative deadline $D_i \leq T_i$, and a priority p_i (a lower value denotes a higher priority). Tasks are indexed such that $p_1 \leq p_2 \leq \ldots \leq p_n$. We assume that tasks are independent, do not share resources and do not self-suspend. The task set utilisation is denoted by $U = \sum_{i=1}^{n} C_i/T_i$.

A *shaper task* (or simply shaper) τ_i^S for an original task $\tau_i \in \tau$ is defined as a sequence of segments $\langle S_{i,1}, S_{i,2}, \ldots, S_{i,m_i} \rangle$, where each segment $S_{i,j} = (r_{i,j}, c_{i,j}, d_{i,j})$ is defined by a relative release time $r_{i,j}$ w.r.t. the release time of a job of τ_i, a relative deadline $d_{i,j}$ w.r.t. $r_{i,j}$, and a budget $c_{i,j}$ that limits the time during which τ_i can execute within segment $S_{i,j}$.

III. Open Problems and Potential Solution

As explained earlier, parameters of the shaper tasks are needed in order to use load-shaping. In addition, a schedulability test is needed to ensure that all timing constraints will be met at run-time. The following problem summarises our main goal.

Problem 1. Given a task set τ and a schedulability test S, find a set of parameters for shapers such that τ becomes schedulable with S.

Note that a solution to Problem 1 always exists. Indeed, we know that by using period transformation and breaking the periods of the original tasks into segments that are equal to the greatest-common-divisor (GCD) of all periods and deadlines, then the shaped task set becomes harmonic. Hence, by using Rate Monotonic (RM) it is possible to schedule the shaped task set with no deadline miss regardless of the original value of periods. However, this solution significantly increases the number of preemptions. Therefore, the actual open problem is the following.

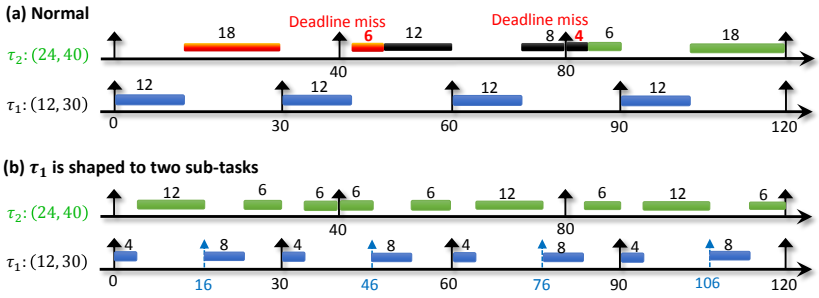

Fig. 1. An example of load shaping. In a normal execution (a), task τ_2 is not schedulable. Using a load shaper (b) with two segments of length 4 and 6 for τ_1, task τ_2 becomes schedulable. Note that deadlines are implicit in this example.

Problem 2. Given an unschedulable task set τ with $U \leq 1$, how can we assign parameters to shapers such that the system becomes schedulable and the number of preemptions is minimised.

Since each segment of a shaper task has a fixed release time w.r.t. to the release time of their job, we can see segments as independent tasks that share the same priority and period but have different release offsets and deadlines. Therefore, we can use existing schedulability tests (e.g., based on a busy-interval) for a set of tasks with release offsets and constrained deadlines as the underlying response-time analysis.

An efficient way to do both parameter assignment and schedulability analysis is to perform these steps together, and hence, find a set of parameters that guarantees schedulability by design. An iterative approach can be used as follows: **(i)** Start by assuming that each task τ_i has a default shaper $\tau_i^S = \langle S_{i,1} \rangle$, where $c_{i,1} = C_i$, $d_{i,1} = D_i$, and $r_{i,1} = 0$, **(ii)** Perform a response-time analysis (RTA) using busy-interval until an unschedulable task τ_i is found. **(iii)** Find a set of candidate high priority tasks and break their shaper tasks into more segments such that the supply-bound-function (SBF) becomes larger than the request-bound-function (RBF) in at least one time instant in $[0, D_i]$. **(iv)** Continue with step (ii) for the next low priority task.

As mentioned in step (iii), the next open problem is to find a time instant t (or maybe a set of such instants) at which the RBF is larger than the supply, and try to reduce RBF(t) by $\delta(t) = \max\{0, \text{RBF}(t) - \text{SBF}(t)\}$. For example, a potential stating point for t is a time instant at which $\delta(t)$ is minimized because in that case, a smaller amount of high-priority workload must be postponed. However, in general, we can perform this step for any $t \in [0, D_i]$ in which $\delta(t)$ changes its value. For each time instant t, there will be a set of high priority tasks that are not finished before t. Among them, those that have their deadline later than t are the candidates whose workload can be postponed.

Another potential application of load-shaping can be found in multi-constrained systems, where the timing behavior of a task might be constrained by other non-functional properties. Tasks that might accept different timing behaviors, and hence use different *set of shaper tasks*, depending on their requirements or system needs. For example, the shaper of a task τ_i can be different according to two different constraints: reduced response-time and reduced processor temperature or energy consumption. In the former mode, the shaper task assigns more budget at the release time of the task while in the latter the shaper task tries to force the task to be scheduled with a particular (and potentially) lower rate leading to a better temperature profile for the system. The open problem therefore becomes how to find the shapers' parameters under additional constraints. Further, if a system has several modes of execution for the same set of tasks (e.g., power saving and high performance mode), one should understand how to switch between modes of one or more shapers without causing any deadline miss for any task in the system.

ACKNOWLEDGMENT

This work is supported by a fellowship from Alexander von Humboldt Foundation. This work was also partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the Portugal2020 Program, within the CISTER Research Unit (CEC/04234).

REFERENCES

[1] Y. Wang and M. Saksena, "Scheduling fixed-priority tasks with preemption threshold," in *International Conference on Real-Time Computing Systems and Applications (RTCSA)*, 1999, pp. 328–335.

[2] A. Burns and A. J. Wellings, "Dual Priority Assignment: A Practical Method for Increasing Processor Utilisation," in *Euromicro Workshop on Real-Time Systems (EWRTS)*, 1993, pp. 48–55.

[3] R. J. Bril, M. M. van den Heuvel, and J. J. Lukkien, "Improved feasibility of fixed-priority scheduling with deferred preemption using preemption thresholds for preemption points," in *International conference on Real-Time Networks and Systems (RTNS)*. ACM, 2013, pp. 255–264.

[4] A. Burns, "Dual Priority Scheduling: Is the Processor Utilisation bound 100%," in *International Real-Time Scheduling Open Problems Seminar (RTSOPS)*, 2010, pp. 3–5.

[5] C. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves: an abstraction for managing processor usage," in *Workshop on Workstation Operating Systems (WWOS)*, 1993, pp. 129–134.