How windows size and number can influence the schedulability of hierarchically-scheduled time-partitioned distributed real-time systems

Andoni Amurrio[1,2], Ekain Azketa[1], Mario Aldea[2], and J.Javier Gutiérrez[2]

[1]Ikerlan, Memeber of Basque Research and Technology Alliance
[2]Software Engineering and Real-Time Group, University of Cantabria

July 16, 2021

Abstract

Partitioning techniques are implemented in the development of safety-critical applications to ensure isolation among components. An adequate scheduling of the execution of such partitions is a key challenge so that applications meet the hard deadlines imposed. In this work, we study the effect of different partition window configuration parameters, with the aim of analyzing their impact in the worst-case response times of system tasks. This is the first step in the development of an algorithm for optimizing partition windows in hierarchically-scheduled time partitioned distributed systems.

Keywords: Time-partitioning, Partition-scheduling, Schedulability-analysis, Safety-critical applications.

1 Introduction

Although scheduling real-time systems has been addressed for many years, novel design, architectural and execution paradigms enforce real-time researchers to continuously develop new strategies to guarantee that deadlines imposed in software are met even in the worst-case scenario. Partitions are strictly independent execution environments protected from each other, and they are used when application components have different criticality levels; it is crucial to avoid that low criticality components jeopardize the execution of high criticality ones. Recent works [1] [2] remark the interest on virtualization/partitioning techniques at many industrial domains, such as automotive or railway, as key issues to be addressed when developing safety-critical applications, and in [3] the will of

train manufacturers to re-factor their applications designs is shown, in order to allow the execution of applications with different criticality levels.

In this work, we focus on a railway signalling application. Up to now, this application has been executed in a cyclic scheduler where all functions are executed even if they do not have any useful work to do. With the aim of substituting this architecture by a novel partitioned execution environment, in [4] a schedulability analysis technique was presented, and based on this, we can propose a partition scheduling optimization algorithm. A more detailed description of this application can be found in [4], and from now on we will construct a simplified (yet representative) system that captures the most relevant features of such safety-critical applications. The experiments conducted in this work will serve as a basis and guide for the development of an optimization algorithm that shall produce schedulable solutions for time-partitioned distributed real-time systems.

The rest of the paper is organized as follows. In Section 2 the system model addressed in this work is described, including an overview of the response time analysis technique. In Section 3 the experiments conducted are presented, and in Section 4 we draw the conclusions and the future works.

2 System Model

In this work we follow a system model compliant with MAST (Modeling and Analysis Suite Tool for Real-Time Applications) [5] [6], which is a GPL open source model and also a set of scheduling, analysis and simulation tools developed by the University of Cantabria.

2.1 Logical Architecture

The main element of this model is the distributed *end-to-end flow* (hereafter e2e flow) as the one depicted in Figure 1, which consists of a sequence of activities with precedence relations executed in response of a periodic or sporadic workload event (e_{in}), with a minimum inter-arrival time (T_i). The main component of an e2e flow is the event handler called *step*, which represents an operation being executed by a schedulable resource (a task or a message) in a processing resource (a computer or a network). Each step is activated by an input event, and after its execution it generates an output event. *Fork* and *Join* event handlers are also allowed, they do not have runtime effects and enable modeling multipath e2e flows. The j-th step in a Γ_i e2e flow is denoted as τ_{ij}, and it has a worst-case and a best-case execution time, C_{ij} and C_{ij}^b respectively, and a deadline D_{ij} relative to the workload event. Each step represents a utilization of the processing resource of $U_{ij} = C_{ij}/T_i$.

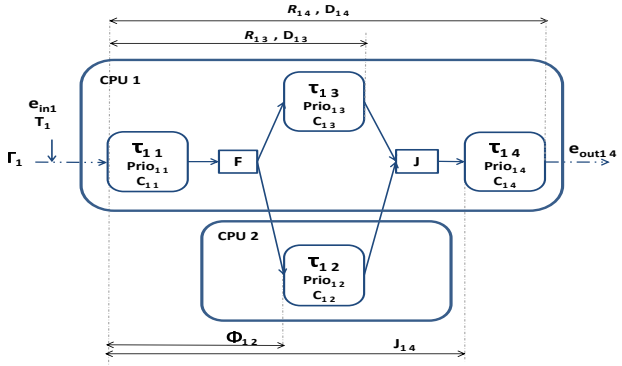


Figure 1: Distributed multipath e2e flow

2.2 Partitioning model

In this work, hierarchically scheduled and time-partitioned systems are addressed. A timetable driven scheduling policy is considered as primary scheduler in every processor, where temporal partitions are scheduled in a cyclic manner. A temporal partition P_x is composed of one or more partition windows Win_{xk} within a periodic Major Frame (MAF). Partition windows start at a time S_{xk} relative to the start of the MAF, and their length is L_{xk}. Hence, partition windows within a temporal partition are defined as follows: Win_{xk} = { S_{xk} , L_{xk} }. Inside each partition, the secondary scheduler is based on preemptive fixed priorities, where $Prio_{ij}$ is the priority of step τ_{ij}, and where the highest number represents the highest priority. The partition utilization of P_x is the sum of the utilization of all the steps contained in that partition: $U_{P_x} = \sum_{\forall \tau_{ij} \in P_x} U_{ij}$. We also define the term Available Utilization (AU_{P_x}) as the processing time allocated to the partition P_x in each processor, which is: $AU_{P_x} = \sum_{\forall Win_{xk} \in P_x} L_{xk}/MAF$.

We are taking into account the overheads provoked by context switches at the primary scheduler. This overhead is the time CS that CPUs need to load a partition context at the beginning of a partition window and saving it after execution terminates. In other words, it can be understood as a non-available CPU time whenever a partition window executes. For response time analysis purpose, this effect can be modelled by gathering this unavailable time at the beginning of every partition window and substracting this amount to the available CPU-time for that window, as shown in the example of Figure 2, where $MAF = 40ms$. In this example a time partition is composed of two partition windows, and the effect of substracting the time for switching the context at each window provokes that the effective partition's (P'_i) length is 2 ms lower than the original P_i. Therefore, the effective partition window is defined as follows: $Win'_{ik} = \{ S_{ik} + CS, L_{ik} - CS\}$.

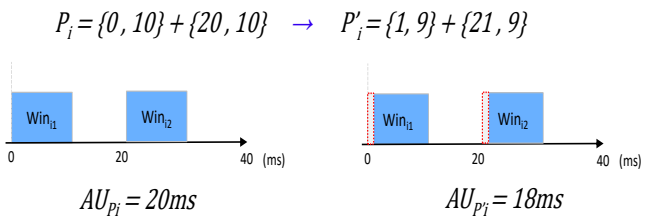


Figure 2: Partition and effective partition with $CS = 1ms$

2.3 Response time analysis

The analysis of a given partition is performed independently: the rest of time-partitions, as well as the unused CPU time within the MAF, are modeled as a single high priority e2e flow, called *unavailability flow*, which has to be analyzed along with the steps hosted in the partition under analysis. To do so, we use the offset-based analysis techniques [7] [8], which was extended in [4] to support the analysis of multipath flows like those addressed in our use-case. Readers are encouraged to read the aforementioned references for a deeper understanding of the schedulability analysis of these systems.

3 Study of the influence of partition windows in schedulability

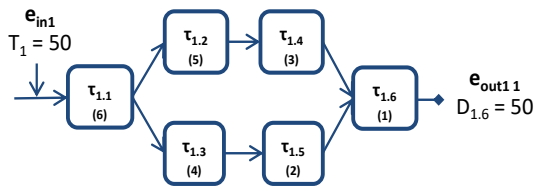


Figure 3: Guiding application example

To perform the proposed study, we will evaluate different partition scheduling schemes. Thus, we are going to construct a simple application example, which gathers the most relevant features that characterize our motivating railway use-case. This example is composed of a single multipath e2e flow activated periodically every 50 ms which is composed of six steps (τ_{11} to τ_{16}), as shown in Figure 3. The flow is mapped within a single partition (P_1), and we assume, for the sake of simplicity, the the execution time of each step is fixed and equals 2 ms. The priority of each step is shown between brackets. When referring to the schedulability of the application, it regards to the worst-case response time of the 6-th step (R_{16}) in comparison with its deadline. The MAF considered for

the whole experiment set is 50 ms.

A very common early-design decision regards to the CPU time allocated for the execution of each partition, i.e. AU_{P_1} in our model. Depending on this time, response times may vary significantly as shown in Figure 4. Even if this effect may result obvious, it gives us an idea about the effect that not having all the processor time dedicated for the execution of applications produces on the worst-case response times calculated by the analisis technique. The longer the gaps between partition windows within the MAF are (where P_1 is not allowed to be executed), the higher is the worst-case response time. In this example the partition utilization U_{P_1} represents 24% of the CPU time, and worst-case response times vary from 580 ms to 12 ms when the available CPU goes from the initial utilization of 24% to the 100%.

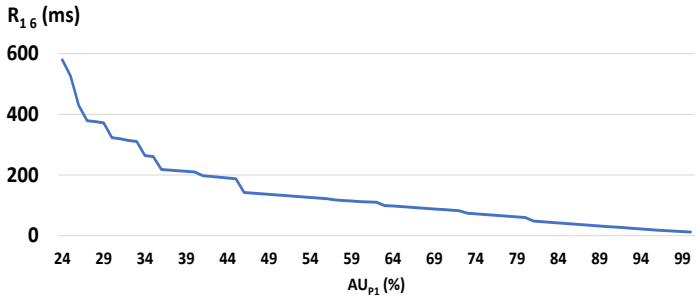


Figure 4: Worst-case response time as a function of P1's available CPU time (in %)

Once the available time has been fixed, the next design decision to take would be to distribute this given time along the MAF. At a first approach, we will consider a harmonic distribution of partition windows, although this might be subject for optimization when more partitions conform the MAF. Figure 5 shows the worst-case response times obtained when the number of partition windows varies from 1 to 100, for three different values of AU_{P_1}. As can be seen, increasing the number of windows produces in fact a reduction in the unavailable gaps in the MAF, making a remarkable reduction of the response times obtained.

Experiments conducted until now have not considered the effects of the context switch overheads that are present when a partition is activated, and which have been modeled in the previous section. In general, context switch overheads depend on the operating system/hypervisor where applications are executed. In [9] the measured context switch overheads are 17μs, and in [10] they are 27μs. Therefore, in our experiments we will consider a value of $CS = 20\mu$s and also a higher order of magnitude representing a slower processor, i.e. $CS = 200\mu$s. The maximum CPU time that can be dedicated for context switch overheads is the difference between the available partition utilization (AU_{P_1}) and the partition utilization (U_{P_1}). With this, we determine the limit of the number of partition windows that can be set without overloading the partition as follows:

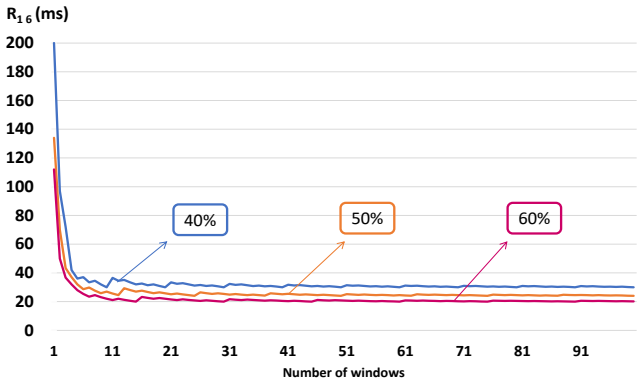
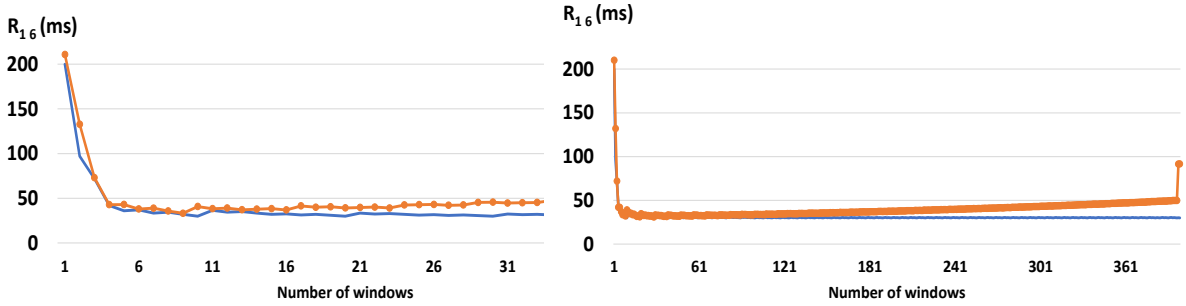


Figure 5: Evolution of worst-case response times when the number of windows is increased, for different % of AU_{P_1}

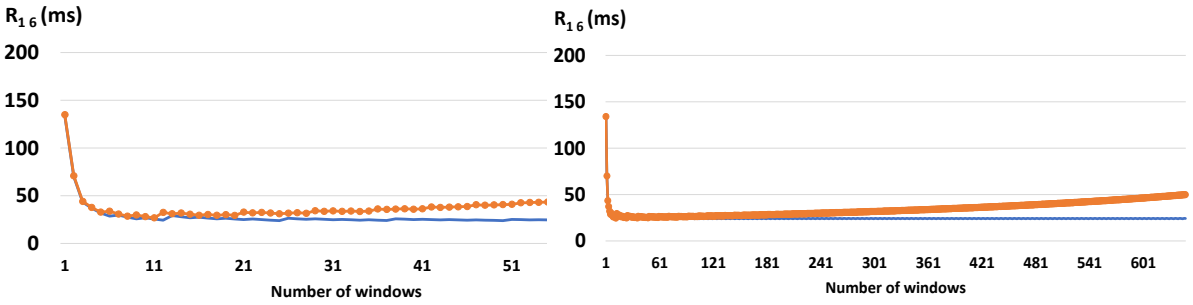
$$NW_{P_1} = \lfloor (AU_{P_1} - U_{P_1}) * \frac{MAF}{CS} \rfloor \tag{1}$$



(a) $CS = 200\mu$s (b) $CS = 20\mu$s

Figure 6: Worst-case response time as a function of the number of partition windows - $AU_{P_1} = 40\%$

In Figures 6 to 8 the worst-case response times obtained in different scheduling schemes are shown. For different AU_{P_1} values we calculate the response times when increasing the partition window number. Qualitatively, worst-case response times vary in the same way regardless the CPU availability, i.e. the maximum response times are obtained when P_1 is scheduled in a single window, and as the number of windows increases response times reduce fast, up to a point. After that point, notice that in the ideal scenario ($CS = 0$, blue plot) the response-time curve remains constant, while if $CS > 0$ (orange plot) the curve increases again.



(a) $CS = 200\mu$s (b) $CS = 20\mu$s

Figure 7: Worst-case response time as a function of the number of partition windows - $AU_{P_1} = 50\%$

4 Conclusions and future work

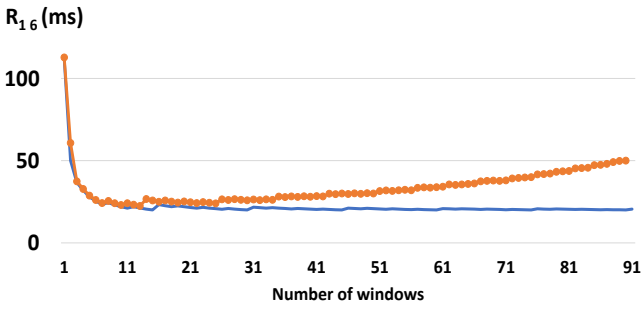
In this work we have presented a study of the behaviour that time-partitioned real-time systems exhibit when worst-case analysis is applied on them, including the effect of context switch overheads on the primary scheduler. The knowledge of this performance will be exploited for developing a partition-scheduling optimization algorithm.

We have observed that increasing the number of partition windows has a positive effect on the reduction of response times, which is outweighed by the negative effect of context switch overheads. Finding the window configuration that gets the turning point on response times is essential for a partition window-optimization algorithm.

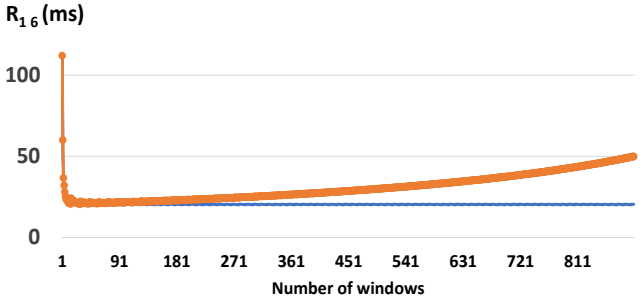
When the partition utilization is very low in comparison to the available utilization, and also when context switch overheads are very low, the range of windows to explore $(1..NW_{P_x})$ is very high. We have noticed that analyzing a very big window number (> 500) has a negative impact in the execution time of the analysis tool. That is why the strategy we shall follow is to explore first the minimum number of windows and increasing them until the turning point is found.

Acknowledgements

Work supported in part by "Doctorados Industriales 2018" program from the University of Cantabria and by the Spanish Government and FEDER funds (AEI/FEDER, UE) under Grant TIN2017-86520-C3-3-R (PRECON-I4).



(a) $CS = 200\mu$s



(b) $CS = 20\mu$s

Figure 8: Worst-case response time as a function of the number of partition windows - $AU_{P_1} = 60\%$

References

[1] C. Donnarumma, A. Biondi, F. De Rosa, and S. Di Carlo, "Integrating online safety-related memory tests in multicore real-time systems," In Proceedings of the 41st IEEE Real-Time Systems Symposium (RTSS), 2020.

[2] Z. Jiang, S. Zhao, P. Dong, D. Yang, R. Wei, N. Guan, and N. Audsley, "Re-thinking mixed-criticality architecture for automotive industry," in *Proceedings of the IEEE 38th International Conference on Computer Design (ICCD)*, pp. 510–517, 2020.

[3] H. Fang and R. Obermaisser, "Execution environment for mixed-criticality train applications based on an integrated architecture," in *2017 International Conference on Promising Electronic Technologies (ICPET)*, pp. 1–7, IEEE, 2017.

[4] A. Amurrio, E. Azketa, J. J. Gutierrez, M. Aldea, and M. G. Harbour, "Response-time analysis of multipath flows in hierarchically-scheduled time-partitioned distributed real-time systems," *IEEE Access*, vol. 8, pp. 196700–196711, 2020.

[5] M. González Harbour, J. J. Gutiérrez, J. C. Palencia, and J. M. Drake, "Mast: Modeling and analysis suite for real time applications," in *in Proceedings of the 13th Euromicro Conference on Real-Time Systems*, pp. 125–134, IEEE, 2001.

[6] M. G. Harbour, J. J. Gutiérrez, J. M. Drake, P. López, and J. C. Palencia, "Modeling distributed real-time systems with mast 2," *Journal of Systems Architecture*, vol. 59, no. 6, pp. 331–340, 2013.

[7] J. C. Palencia and M. González Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279)*, pp. 26–37, IEEE, 1998.

[8] J. C. Palencia, M. González Harbour, J. J. Gutiérrez, and J. M. Rivas, "Response-time analysis in hierarchically-scheduled time-partitioned distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2017–2030, 2016.

[9] E. Hamelin, M. A. Hmid, A. Naji, and Y. Mouafo-Tchinda, "Selection and evaluation of an embedded hypervisor: application to an automotive platform," Proceedings of the 10th Embedded Real-Time Systems International Congress (ERTS 2020).

[10] M. Masmano, I. Ripoll, A. Crespo, and J. Metge, "Xtratum: a hypervisor for safety critical embedded systems," Proceedings of the 11th Real-Time Linux Workshop 2009, pages 263-272.