

# Towards a General Framework to Model, Analyze and Optimize Real-time Systems with GPUs

Iosu Gomez<sup>a,b</sup>, Juan M. Rivas<sup>b</sup>, J. Javier Gutiérrez<sup>b</sup>, Jorge Parra<sup>a</sup>, Unai Díaz de Cerio<sup>a</sup>

<sup>a</sup>*Distributed and Connected Intelligence Department, IKERLAN Research Centre, Basque Research and Technology Alliance (BRTA), Arrasate-Mondragón, Spain.*

<sup>b</sup>*Software Engineering and Real-Time Group, Universidad de Cantabria, Spain.*

---

**Keywords:** real-time, GPU, modeling, analysis, optimization

---

## 1. Extended Abstract

Although the computational power and efficiency of GPUs would clearly benefit emerging real-time applications such as smart mobility, the adoption of such accelerators is being hindered by the poorly documented nature of their internal scheduling mechanisms. There is presently an intense research effort to propose solutions to enable the safe usage of GPUs in real-time applications [1] [2], although their applicability remains a challenge.

In this extended abstract we present our methodology to safely incorporate GPUs into real-time systems. We propose a comprehensive framework that builds upon existing and validated tools and techniques, based on three main aspects: (1) an extension of an industry relevant meta-model, called MAST-2, to support GPUs, (2) leveraging time partitioning to control the access to the GPUs, which enables the application of existing WCRT (Worst-Case Response Time) analysis techniques, and (3) an optimization framework that takes advantage of the previous meta-model and analysis, to construct optimized time partitions. These aspects are described in the following sections. This work is the continuation of the Work-in-Progress presented in the ECRTS 2023 Industrial Challenge [3], and is developed in the context of an industrial collaboration with Ikerlan Research Centre [4] and the railway vehicles manufacturer CAF [5].

### 1.1. System Model: Extending MAST 2

MAST (Modeling and Analysis Suite for real-Time applications) [6] is a modeling technique together with a set of analysis tools. The MAST meta-model allows describing the timing properties of a real-time system, focusing on the relevant aspects for the analysis of the timing behavior. The model describes the software architecture of the application, as well as its deployment on the hardware platform. The second version of the meta-model, called MAST 2 [7], added support to ARINC 653 time partitioning and ARINC 664 (AFDX) communication networks, and is aligned with the UML profile for Modeling and Analysis of Real-Time Embedded Systems (MARTE) [8]. In its current version, MAST 2 does not incorporate the concept of GPUs. As a necessary step towards a general framework to exploit GPUs in real-time systems, we extend MAST 2 to support two different paths to model such systems, depending on the level of detail required and/or supported.

For a coarse grained modeling of GPUs, we add a new type of task (*step* in MAST 2), called GPU\_Step, that models a section of code that is executed in a GPU. In contrast to a standard Step, a GPU\_Step is not assigned to any scheduler or processor, and just includes an execution time that is assumed to be static and known. A GPU\_Step can be used in those situations in which the GPU is considered a black box, with no attempt to analyze its internal contentions.

For a more detailed modeling of GPUs, we add a new type of processor, called GPU\_Processor, which aims to model the whole GPU, including all the aspects that could influence its internal scheduling, such as its number of cores or other discrete resources (e.g. copy engine). The access to the GPU is governed by a new abstract scheduling policy called GPU\_Policy, which should be extended as the scheduling policies governing the GPU become known. These detailed model elements are included to future-proof MAST 2, even if a precise analysis that takes into account such information remains currently outside the state-of-the-art.

### 1.2. Response-time Analysis of Real-time Systems with GPUs

We propose to use a two-level scheduling scheme: a higher level that implements time partitioning, and a priority-based lower level to schedule the workload inside each partition. Time partitioning is usually deployed to achieve time isolation among different functionalities in the system. With our approach, it can also be leveraged to pre-establish at which instants it is allowed to offload kernels into the GPU. For instance, all the CPU tasks that offload workload into the GPU can be mapped to a unique time partition, effectively resulting in a serialization of the accesses to the GPU. By limiting by construction the amount of concurrent kernels, the contentions that may occur inside the GPU are simplified, and therefore simple techniques such as directly measuring the execution times of the kernels inside the GPU in isolation become viable.

Given the scheme described above, we can model the execution of the kernels as static delays, which is supported by the "coarse grained" GPU\_Step included in MAST 2. These delays must accommodate the maximum execution times that are expected from the kernels, which can be established via measurements. There exist several worst-case response-time analysis techniques that can be used to analyze these workloads that contain delays and time partitioning [9], which are based on the Offset-Based Analysis [10, 11].

### 1.3. Optimization Framework

The execution scheme described in the previous section relies on the construction of a proper time partition plan. This represents a complex optimization problem, as it involves several inter-dependent sub-problems: (1) definition of the time partitions, (3) mapping of partitions to processors, (4) mapping of tasks to partitions, and (5) assignment of priorities to tasks. Furthermore, this optimization process should also take into account that the WCETs can not be considered static or known in advance. In particular, each plan can produce different interference patterns in the access to shared resources, including GPUs, that may lead to different WCETs.

To solve this, we are working on extending existing time partitioning optimization algorithms, such as [12], with the capability to automatically deploying testing code on real hardware platforms to empirically determine the WCETs of the tasks. An overview of the algorithm is depicted in Figure 1, which is composed of the following steps:

1. The input model is a description of the real-time system (timing properties and platform), with an initial estimation of WCETs.
2. A plan is constructed (partitions, mapping and priorities), applying the current estimations of the WCETs.
3. Testing code is constructed and deployed in the real hardware platform. The objective of these tests is to obtain realistic measurements of the WCETs for the interferences expected with the current plan.
4. WCETs are obtained from the measurements.
5. The WCETs are integrated into the model, and a WCRT analysis is launched. This analysis could be the one described in section 1.2, which considers the GPU kernels as delays.
6. WCRTs are obtained, which can be compared with the deadlines to determine the schedulability of the system.
7. If the current plan is determined to be schedulable, the algorithm returns the model enhanced with the optimized plan and associated WCETs.

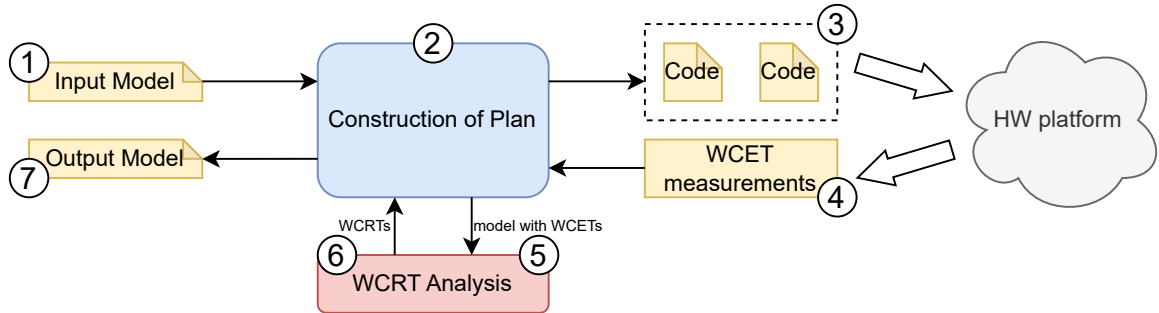


Figure 1: Optimization framework main components and flow

## Acknowledgements

This work was partially supported by MCIN/ AEI /10.13039/501100011033/ FEDER “Una manera de hacer Europa” under grants PID2021-124502OB-C42 and PID2021-124502OB-C44 (PRESECREL).

## References

- [1] J. Perez-Cerrolaza, J. Abella, L. Kosmidis, A. J. Calderon, F. Cazorla, J. L. Flores, GPU Devices for Safety-Critical Systems: A Survey, *ACM Computing Surveys* 55 (12 2022). doi:10.1145/3549526.
- [2] I. Gomez, U. D. de Cerio, J. Parra, J. M. Rivas, J. J. Gutiérrez, Using GPUs in Real-Time Applications - A Review of Techniques for Analyzing and Optimizing the Timing Parameters, *Revista Iberoamericana de Automática e Informática industrial* 21(1) (2023) 1–16. doi:doi.org/10.4995/riai.2023.20321.
- [3] I. Gomez, U. Díaz-de Cerio, J. Parra, J. M. Rivas Concepción, J. J. Gutiérrez García, Analysis and optimization of real-time applications running on heterogeneous hardware, in: 35th Euromicro Conference on Real-Time Systems (ECRTS 2023), Industrial Challenge (Early Stage Proposal), 2023.  
URL <https://hdl.handle.net/10902/31005>
- [4] Ikerlan research center, (Last update Jan 2024), url: <https://www.ikerlan.es/en/> (2024).
- [5] Grupo CAF (Construcciones y Auxiliar de Ferrocarriles), (Last update Jan 2024), url: <https://www.caf.net/en> (2024).
- [6] M. G. Harbour, J. J. Gutiérrez, J. C. Palencia, J. M. Drake, MAST: Modeling and Analysis Suite for Real Time Applications, in: *Proceedings of 13th Euromicro Conference on Real-Time Systems*, IEEE Computer Society Press, 2001, pp. 125–134.
- [7] M. G. Harbour, J. J. Gutiérrez, J. M. Drake, P. L. Martínez, J. C. Palencia, Modeling distributed real-time systems with MAST 2, *Journal of Systems Architecture* 59 (2013) 331–340. doi:10.1016/j.sysarc.2012.02.001.
- [8] OMG, An OMG UML Profile for MARTE TM publication UML Profile for MARTE TM : Modeling and Analysis of Real-Time Embedded Systems. Version 1.2. (2019).
- [9] J. C. Palencia, M. G. Harbour, J. J. Gutiérrez, J. M. Rivas, Response-time analysis in hierarchically-scheduled time-partitioned distributed systems, *IEEE Transactions on Parallel and Distributed Systems* 28 (2017) 2017–2030. doi:10.1109/TPDS.2016.2642960.
- [10] J. C. Palencia, M. G. Harbour, Schedulability Analysis for Tasks with Static and Dynamic Offsets, in: *Proceedings of 19th IEEE Real-Time Systems Symposium*, IEEE, 1998, pp. 26–37. doi:10.1109/REAL.1998.739728.
- [11] J. C. Palencia, M. G. Harbour, Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems, in: *Proceedings of 20th IEEE Real-Time Systems Symposium*, 1999, pp. 328–339. doi:10.1109/REAL.1999.818860.
- [12] A. Amurrio, J. J. Gutiérrez, M. Aldea, E. Azketa, Partition window assignment in hierarchically scheduled time-partitioned distributed real-time systems with multipath flows, *Journal of Systems Architecture* 130 (9 2022). doi:10.1016/j.sysarc.2022.102671.