Optimization in Heterogeneous Distributed Real-Time Systems based on Partitioning

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

[1]Ikerlan Technology Research Centre, Arrasate-Mondragón, Spain
[2]Universidad de Cantabria, Santander, Spain

***Abstract*—In this work, a solution that can be applied to the RTSS 2022's Industry Challenge is proposed. It relies on a real-time system model and a set of schedulability analysis and optimization tools, enabling the design of safety-critical applications compliant with timing requirements. The presented toolchain is enhanced with a novel task allocation technique, which leverages sensitivity analysis and that can be applied to heterogeneous systems, to provide promising solutions that improve state-of-the-art algorithms' performance.**

***Index Terms*—MAST, Schedulability analysis, Optimization, Partitioning**

I. Introduction

Nowadays, the design of safety-critical systems is becoming more and more complex, as the need of high-performance computing platforms spreads all over the industry. Complex and concurrent software functions, each of them with its own timing constraints and safety requirements, need to be executed within resource-constrained embedded systems, for instance, in Advanced Driver-Assistance Systems (ADAS) [11] similar to the visual-based navigation system devised in this Industry Challenge. Object and obstacle detection or image classification and segmentation duties must be performed within a bounded time or deadline, as human lives may be jeopardized in case of any error.

In this context, we propose to use partitioning techniques [1] [5] [14] in order to isolate software functions that share a computing platform. As a solution to the proposed challenge, in this paper we suggest the use of a set of modeling, analysis and optimization tools, with the aim of providing the means for guaranteeing that safety critical applications meet their deadlines even in the worst-case scenario, as part of the safety certification process. We show that the system model captures the complexity that these systems exhibit, and we propose using several schedulability analysis and optimization techniques. We also propose a novel method to allocate tasks to processors that produces promising solutions.

After a brief introduction in Section I, Section II presents the system model, the schedulability analysis technique and the optimization algorithms. Then, the new task allocation algorithm is described in Section III, including a description the sensitivity analysis, the algorithm's design and its evaluation. Finally, Section IV draws some conclusions and discuses future research lines.

II. Modeling, analysis and optimization

A. System model

This system model captures the time-related features and complexity of safety critical applications. It is compliant with the MAST model. The second version of MAST's metamodel, MAST 2 [7], adds several novel scheduling policies and modeling elements, such as time partitioning.

The main element is *the end-to-end (e2e) flow*, which consists of a sequence of activities with precedence relations executed in response to a periodic or sporadic workload event, 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 processor or a network). Processors are denoted as CPU_y. Each step is activated by an input event, and after its execution it generates an output event. The j-th step in the e2e flow Γ_i is denoted as τ_{ij}, and it has a worst-case and a best-case execution time, C_{ij} [*** anadir CPU y] and C^b_{ij} respectively. These execution times may be different depending on the processor they are allocated to, so that heterogeneous computing platforms can be modeled. Each step represents a utilization of the processing resource of $U_{ij} = C_{ij}/T_i$, and therefore, the utilization of an e2e-flow (U_i) is the sum of the utilization of all its steps. Workload events that activate e2e flows and also the internal events that activate handlers may exhibit a release jitter, so any step τ_{ij} may suffer a release jitter up to a maximum of J_{ij}. Steps can also have an initial offset Φ_{ij}.

This model includes other event handlers that do not have runtime effects and enable the modeling of complex event combinations like the multipath e2e flows addressed here: 1) Fork: It generates one event in each of its outputs each time an input event arrives. 2) Join: It generates an output event when all of its input events have arrived.

Hierarchical schedulers are composed of a primary scheduler and a secondary scheduler. A table-driven scheduling policy is considered as primary scheduler in every processor, where temporal partitions are scheduled in a cyclic manner within a Major Frame (MAF). A temporal partition P_x is composed of one or more partition windows Win_{xk}, defined as follows: Win_{xk} = { S_{xk} , L_{xk} } where S_{xk} is the start time relative to the start of the MAF, and L_{xk} is its length. The secondary scheduler is based on preemptive fixed priorities,

where $Prio_{ij}$ is the priority of the step τ_{ij}, and where the highest number represents the highest priority.

B. Response-time analysis

In order to determine whether safety critical applications meet their deadlines, we need to calculate the steps' worst-case response times. To do so, we will make use of the technique developed in [2]. This is an offset-based schedulability analysis technique [12] [13], which has been extended in [2] to support multipath e2e flows. It is the most accurate technique available to calculate worst-case response times, as it is demonstrated that the results of the holistic analysis [6] for multipath flows are notably improved. This technique stands out from the state-of-the-art approaches in some key features: on the one hand, it can be applied to event-triggered multipath applications whose deadlines may be greater than their periods, and on the other hand, it supports both time-partitioned distributed real-time systems as well as general distributed real-time systems (without partitioning) based on fixed priorities. Readers are encouraged to read the aforementioned references for specific details on the schedulability analysis and optimization tools.

C. Priorities and partition windows optimization

With the aim of minimizing the worst-case response times of safety critical applications, we propose a set of scheduling optimization techniques, that are compliant with the system model previously described. In this section, we address both scheduling levels: we show a partition window assignment algorithm that can be applied to time partitioned distributed real-time systems [4], and then a collection of 8 non-iterative priority assignment algorithms, which can be applied to the secondary scheduler of hierarchical systems and also to general fixed-priority based distributed real-time systems [3]. Readers are encouraged to read the corresponding references for further details on their design, implementation and results.

Partitioning is a key technique for isolating different functions that share the same computing platform. The algorithm presented in [4] can be used in this challenge in order to perform an optimized partition window assignment that allows applications based on hierarchical scheduling to meet their deadlines. It is shown that schedulable solutions can be found for a wide range of deadline requirements, taking into account the overheads produced by partition context switches.

Regarding the priority assignment algorithms, we will rely on the priority assignment techniques developed in [3], which can be applied to multipath e2e flows and time-partitioned distributed real-time systems. These algorithms produce a single solution by distributing the end-to-end deadline through all the steps in the e2e flow (called Virtual Deadlines or VDs), and then transforming such VDs into priorities following the deadline monotonic algorithm. The main conclusion in [3] is that there is no algorithm that stands out from the others in the tested scenarios, which reinforces the idea of evaluating all of them and choosing the one that produces the best solution. Notice that there are two algorithms (NPD_Global and NPD_Local) that rely on a given step-to-processor allocation, so they are not going to be applicable in the strategy proposed for this challenge.

III. Slack-Based Allocation algorithm

A. Sensitivity analysis (Slack)

The algorithm developed in this work is based on the Slack as defined in MAST[1] for sensitivity analysis. Thus, a Slack Factor (SF) is defined as the factor by which the worst-case execution times of a step or a set of steps may be increased while keeping the system schedulable (SF is then a positive value higher than 1), or decreased, in order to make the system schedulable (SF is then a positive value lower than 1). If SF is 1, the system is just schedulable. This definition of SF can be applied to different sets of steps, thus obtaining the following parameters that allow a comparison between different elements of the system or even the whole system:

- System Slack Factor (SSF), if all the steps in the systems are considered.
- Processor Slack Factor (CPUSF), if the SF is calculated by modifying only the steps of a particular processor.

These parameters enable the determination of how close a particular element is to schedulability, and they are obtained by a binary search algorithm in which execution times are successively increased or decreased. It is assumed that, when the SF calculation reaches a near-zero threshold, the calculation will stop, meaning that the system cannot be scheduled even if this element is removed.

B. Algorithm design

The design of the Slack-Based Allocation (SBA) algorithm, proposed in Algorithm 1 for step-to-processor allocation in distributed real-time systems, is inspired by traditional bin-packing algorithms [9], where tasks are allocated to different processors depending on their utilization. Its input is a distributed system based on preemptive FP scheduling and composed of steps that are not allocated to any processor. Its output is the step-to-processor allocation together with a priority assignment for each step. Instead of considering processors' utilization as a reference parameter, with the aim of capturing the effect on the worst-case response times of the decisions that the allocation algorithm takes in each stage, the System Slack Factor (SSF) and the Processor Slack Factor will be used. These parameters give an idea of the goodness (in terms of schedulability) of the decision of allocating a step to a certain processor.

In each stage of the algorithm, a $Candidate\_List$ is created, which gathers the set of processors where the step under assignment can be allocated. First, steps are sorted according to their priority (Line 2), and in each allocation stage one of them (the first one in the sorting) is allocated into a processor. As will be explained later, the proposed algorithm has to calculate the slacks at each allocation stage, which implicitly means that response-time analysis should be carried out. This kind of analysis has to be executed over different subsets

[1] https://mast.unican.es/

of steps which must be compliant with the e2e flow model. Therefore, step-to-processor allocation must be performed in such a way that steps' precedence relations are preserved. To achieve this, priorities must be assigned in a non-decreasing order from the first step until the end. This particular priority assignment can be achieved by a subset of the 8 algorithms mentioned in Section II.C.

Algorithm 1 Slack-based Allocation Algorithm

```
1: Input: Set of steps τ_ij, set of Processors CPU_y
2: Priority Assignment
3: Sort all τ_ij according to Prio_ij
4: for each τ_ij do
5:    MaxSSF = 0 , Clear Candidate_List
6:    for each CPU_y do
7:       CPU_y ← τ_ij
8:       Calculate System Slack
9:       if MaxSSF < System_Slack then
10:         MaxSSF = System_Slack
11:         Clear Candidate_List
12:         Candidate_List ← CPU_y
13:      else if System_Slack = MaxSSF then
14:         Candidate_List ← CPU_y
15:      end if
16:   end for
17:   if CPU_y ∈ Candidate_List >1 then
18:      MaxCPUSF = 0
19:      for each CPU_y ∈ Candidate_List do
20:         CPU_y ← τ_ij
21:         Calculate Processor_Slack at CPU_y
22:         if Processor_Slack >MaxCPUSF then
23:            CPU_y ← τ_ij
24:         end if
25:      end for
26:   else
27:      CPU_y ∈ Candidate_List ← τ_ij
28:   end if
29: end for
```

At each stage (Line 4), the algorithm allocates the step in all processors and calculates the resulting system slacks (Line 8). If there is a single processor where the maximum SSF (*MaxSSF*) is achieved, the step is allocated to that processor (Line 27), whereas if there is more than one, all of them are added to the $Candidate\_List$ and their CPUSFs are calculated (Line 21). The step will be allocated to the processor where the maximum CPUSF (*MaxCPUSF*) is achieved, and if there is still a tie at processor SF, the allocation is decided randomly, subject to future optimization strategies to be evaluated.

As said before, a partition window assignment algorithm can be applied as part of the proposed solution to this challenge, even if partitioned systems are not going to be evaluated in this paper. We propose to apply this partition optimization algorithm right after the allocation is performed, as the algorithm developed in [4] requires a given step-to-processor allocation as input.

C. Evaluation

This novel algorithm is conceived for applications compliant with the system model described before, that is, heterogeneous platforms based on partitioning. In this work, we show a set of experiments where the slack is applied to allocate tasks in homogeneous systems without partitioning. This allows to characterize the performance of the algorithm on basic systems. The influence of partitioning and heterogeneous processors will require further experiments and is left for future work.

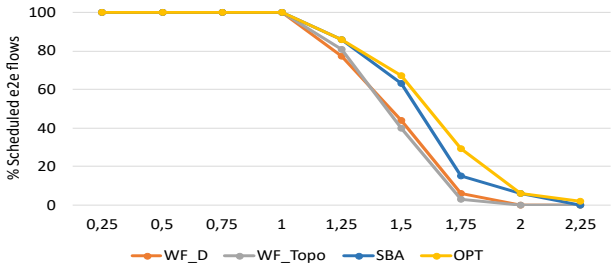
The experiments are based on randomly generated synthetic e2e flows, generated following the principles in [10]. A special purpose generation tool has been developed and it has also been made publicly available[2]. It is based on the well-known DAG model [8] [15], which can be directly transformed to the multipath flow model described in this work. The utilization of the generated e2e flows can be provided as input parameters to the tool, as well as the number of e2e flows and the steps within them. Test cases consist of different sized e2e flows, in terms of the number of steps, which are allocated individually to a varying number of processors. A test case will be developed through the execution of the proposed algorithm to allocate a single e2e flow into a set of processors. A hundred different e2e flows are tested for incremental utilizations in a specific range with a fixed increasing step. In all experiments, the SBA algorithm is compared against the well-known Worst-Fit algorithm [9] considering two variations, one where steps are sorted in decreasing utilization order (WF_D) and the other where steps are sorted in a topological arbitrary order (WF_Topo). If the opposite is not stated, the priority assignment algorithm used to sort the steps at the beginning of the algorithm will be PD_Global [3].

The first test case consists of small sized e2e flows, where the number of steps in each e2e flow is in the range [7,10]. The utilization of the e2e flows has been set from 0.25 to 2.5, with an increasing step of 0.25, and they will be allocated to 3 and 4 processors. The optimal solution, i.e the lowest worst-case response time among all the possible allocation solutions, will also be evaluated, in order to see how close the solutions obtained are to the optimal one.

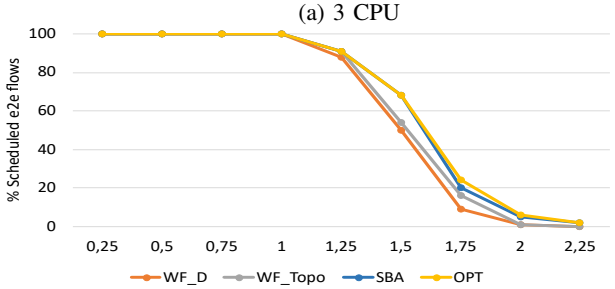
In the second test case, corresponding to medium-sized e2e flows, the input value in the generation tool is set so that the generated e2e flows are composed of a number of steps in the range [10,25]. For this experiment, the e2e flows will be allocated to 3, 4 and 5 processors. The e2e flows' utilizations are from 0.33 to 3.33 with an increasing step of 0.33. Due to the larger size of this experiment, the optimal solution's reference is no longer available for these results.

Finally, a large-sized e2e flow experiment is performed by generating e2e flows with a number of steps in the range [50-100], and with utilization values from 0.75 to 5 with an

[2]https://github.com/mive93/DAG-scheduling

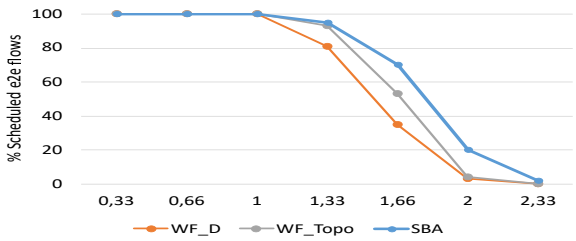


(a) 3 CPU

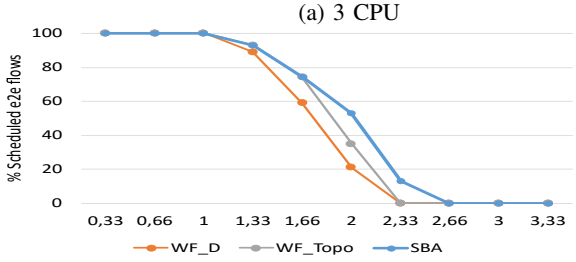


(b) 4 CPU

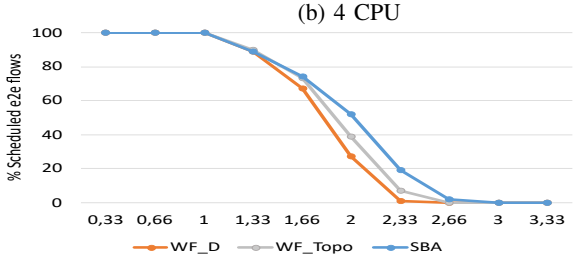
Fig. 1: Small-sized test



(a) 3 CPU



(b) 4 CPU

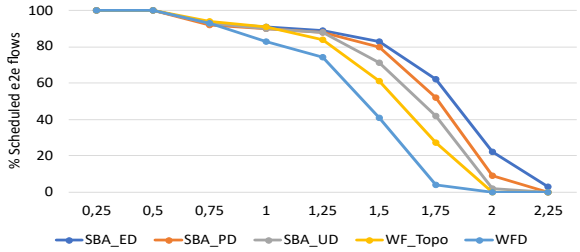


(c) 5 CPU

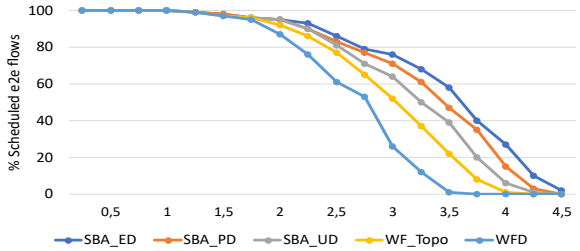
Fig. 2: Medium-sized test

increasing step of 0.25. They will be allocated to 4, 5 and 6 processors. In this test case, another feature of interest will be evaluated. As stated before, steps are initially sorted according to their priorities, which are assigned following different algorithms. In order to assess the impact of the priority assignment on the initial sorting, and therefore on the allocation algorithm, in this experiment the schedulability results will be presented for different step orderings. The evaluated priority assignment algorithms will be ED (SBA_ED), PD_Global (SBA_PD) and UD (SBA_UD) [3].

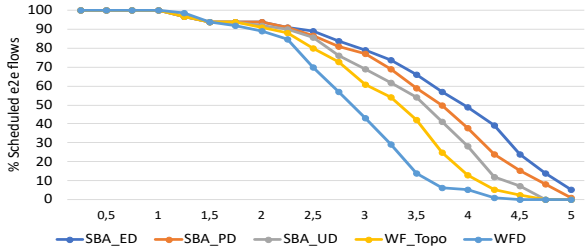
As said before, the SBA algorithm will be compared with two bin-packing algorithms, WF_D and WF_Topo, and also with the optimal solution in small-sized experiments. The results have been plotted in Figures 1, 2 and 3. They show, for each utilization value, the percentage of e2e flows that meet their deadlines after being allocated through the different methods. In all figures, each graph represents a different number of processors where the e2e flows have been allocated.



(a) 4 CPU



(b) 5 CPU



(c) 6 CPU

Fig. 3: Large-sized test

In Figure 1, the SBA algorithm (blue plot) outperforms the bin packing algorithms and obtains near-optimal solutions for the small-sized test case. As shown in Figure 2, the proposed algorithm still produces more schedulable results than the WF algorithms evaluated for all numbers of processors. Figure 3 shows that the SBA algorithm remains better than the bin packing algorithms for all the utilization values and number of processors evaluated. Moreover, it can be seen that the initial sorting of steps, performed via different priority assignment algorithms, has a paramount importance in the schedulability of applications. For any number of processors, the initial step

sorting produced by using the ED algorithm shows the best performance, so future research experiments should consider this as a reference.

IV. Conclusions and future work

In this paper, a solution applicable to the Industry Challenge at RTSS is presented. This solution is based on a rich real-time system model which includes partitioning as a key feature for safety-critical applications, and also several optimization algorithms that allow obtaining schedulable solutions for applications executed in heterogeneous platforms. Future work will be focused on a deeper characterization of the new task-to-processor allocation algorithm that has been proposed here, including thorough experimentation considering realistic scenarios.

Acknowledgements

This work was supported in part by MCIN/ AEI /10.13039/501100011033/ FEDER "Una manera de hacer Europa" under grant PID2021-124502OB-C42 (PRESECREL). The authors would like to thank Dr. Ignacio Sañudo, Dr. Micaela Verruchi and Dr. Marko Bertogna, from the University of Modena and Reggio Emilia, for their advice in the early stages of this research.

References

[1] Airlines Electronic Engineering Committee, Aeronautical Radio INC. Avionics application software standard interface. arinc specification 653-1. *AERONAUTICAL RADIO, INC*, 2551:21401–7435, 2010.

[2] 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*, 8:196700–196711, 2020.

[3] A. Amurrio, J. J. Gutiérrez, M. Aldea, and E. Azketa. Priority assignment in hierarchically scheduled time-partitioned distributed real-time systems with multipath flows. *Journal of Systems Architecture*, 122:102339, 2022.

[4] A. Amurrio, J. J. Gutiérrez, M. Aldea, and E. Azketa. Partition window assignment in hierarchically scheduled time-partitioned distributed real-time systems with multipath flows. *Journal of Systems Architecture*, 130:102671, 2022.

[5] A. Crespo, A. Alonso, M. Marcos, A. Juan, and P. Balbastre. Mixed criticality in control systems. *IFAC Proceedings Volumes*, 47(3):12261–12271, 2014.

[6] J. J. Gutiérrez, J. C. Palencia, and M. G. Harbour. Schedulability analysis of distributed hard real-time systems with multiple-event synchronization. In *Proceedings 12th Euromicro Conference on Real-Time Systems. Euromicro RTS 2000*, pages 15–24. IEEE, 2000.

[7] 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*, 59(6):331–340, 2013.

[8] C. Liu and J. H. Anderson. Supporting soft real-time dag-based systems on multiprocessors with no utilization loss. In *2010 31st IEEE Real-Time Systems Symposium*, pages 3–13. IEEE, 2010.

[9] E. C. man Jr, M. Garey, and D. Johnson. Approximation algorithms for bin packing: A survey. *Approximation algorithms for NP-hard problems*, pages 46–93, 1996.

[10] A. Melani, M. Bertogna, V. Bonifaci, A. Marchetti-Spaccamela, and G. C. Buttazzo. Response-time analysis of conditional dag tasks in multiprocessor systems. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 211–221. IEEE, 2015.

[11] I. S. Olmedo, N. Capodieci, and R. Cavicchioli. A perspective on safety and real-time issues for gpu accelerated adas. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 4071–4077. IEEE, 2018.

[12] 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)*, pages 26–37. IEEE, 1998.

[13] 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*, 28(7):2017–2030, 2016.

[14] S. Trujillo, A. Crespo, A. Alonso, and J. Pérez. Multipartes: Multicore partitioning and virtualization for easing the certification of mixed-criticality systems. *Microprocessors and Microsystems*, 38(8):921–932, 2014.

[15] M. Verucchi, M. Theile, M. Caccamo, and M. Bertogna. Latency-aware generation of single-rate dags from multi-rate task sets. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 226–238, 2020.