Execution time budget assignment for mixed criticality systems

Mohamed Amine Khelassi, Yasmina Abdeddaïm

Execution time budget assignment for mixed criticality systems

Mohamed Amine Khelassi, Yasmina Abdeddaïm
Univ Gustave Eiffel, CNRS, LIGM, F-77454 Marne-la-Vallée, France

***Abstract*—In this paper we propose to quantify execution time variability of programs using statistical dispersion parameters. We show how the execution time variability can be exploited in mixed criticality real-time systems. We propose a heuristic to compute the execution time budget to be allocated to each low criticality real-time task according to its execution time variability. We show using experiments and simulations that the proposed heuristic reduces the probability of exceeding the allocated budget compared to algorithms which do not take into account the execution time variability parameter.**

***Index Terms*—Mixed criticality systems, execution time variability, statistical dispersion parameters.**

I. Introduction

In a mixed criticality system, programs of different criticality are executed on the same processor. The challenge is that low criticality tasks do not disturb the good functioning of the high criticality ones. In real-time scheduling, since the original Vestal's model [1], a classical model has emerged, see [2] for a complete survey. In this model, tasks have several execution times budgets, one budget per possible criticality. If a task does not signal its termination after the execution of its allocated budget at a certain criticality level, the system moves to the next criticality level. In every system criticality level, only tasks of criticality equal or higher to the criticality of the system have to respect their deadlines. Therefore, low criticality tasks can be suspended to allow higher criticality tasks to meet their deadlines. Much research in real-time mixed criticality scheduling assumes that the execution time budgets in every criticality level are provided and focus on the challenge of maximizing the execution of low criticality tasks while guaranteeing that all high criticality tasks always meet their deadlines. The execution time budgets are often defined as estimates of the worst-case execution time at different certification requirements but few studies have focused on how to determine these budgets.

In this paper our aim is to determine the execution time budget of low criticality tasks so that all the tasks always respects their deadlines. If a job does not complete within its assigned budget, the job is suspended. We propose a heuristic to determine the execution time budgets, this heuristic uses an estimation of the variability of execution time of the task to determine the budgets. Many definitions have been proposed for the quantification of execution time variability, in [3] and [4] the execution time variability is quantified as the ratio between run-time measurements and either the best, worst or mean execution time. In [5], [6], the execution time variability is considered to be the factor between the worst-case execution time computed in isolation and along with other workload. These approaches give a quantification that does not give information on how the execution times are distributed as we would like to have. In this work we use statistical dispersion parameters to quantify the execution time variability of a task. The idea is to characterize the variability of execution times more accurately, so that we can use it to determine the budgets to be allocated to tasks. Compared to probabilistic approaches [7] that rely on the computation of the probabilistic response time to determine if the probability of a deadline miss respects a given threshold, our objective is to use an estimate of the variability of execution times without using the complete distribution. Indeed, the methods that use the full distribution to compute the probabilistic response time have a high complexity for exact methods or have to make assumptions on the shape of the distributions for analytical methods. The contributions of the paper are: (1) We propose a definition of execution time variability and a method for its quantification using statistical dispersion parameters, (2) We propose a heuristic that uses the execution time variability to solve the scheduling problem of a mixed criticality system, (3) We evaluate our approach using simulations and benchmarks executed on an ARM-Cortex A53.

II. Related work

Concerning the use of statistical dispersion parameters in real-time systems, in [8], [9] the authors use statistical dispersion parameters to measure the variability in the experimentation part to compare different execution time distributions of different modules of a system. The difference with our proposition is that in our work statistical dispersion parameters are part of the task model and not a measure used to estimate the efficiency of an approach. Decreasing the computation times of LO-criticality tasks in order to avoid sacrificing all LO criticality tasks when the system is in high criticality has been considered in [10]–[12]. In these papers, a new model is used for mixed criticality systems in which the budget of low criticality tasks is smaller when the system is in high criticality than their budget when the system is in low criticality. Our approach differs mainly in the following aspects. Firstly, in our model, all non-critical tasks have a single budget assigned to them. This budget has a practical significance and is linked to the distribution of task execution times. Secondly, our aim is to propose an approach that is agnostic to the scheduling algorithm used. Thirdly, the only

	Merge sort	Quick sort	Insert sort
VWCET	7,911	0,459	4,75
sKw	9,559	-0,399	-4,186

TABLE I
VWCET AND SKW OF MERGE SORT, QUICK SORT AND INSERT SORT

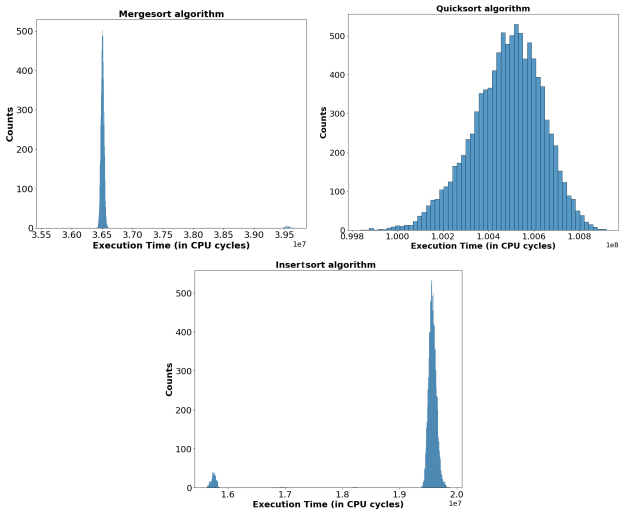

Fig. 1. Execution times (cycles) of programs executed in an ARM Cortex 53

monitoring we need to perform during execution is to stop jobs if they do not terminate at their assigned budget.

Our model has some similarities with the approaches using probabilistic tasks models for mixed criticality systems [13]–[15] as we also consider that we have a distribution of execution times of the tasks. However, we don't use the distribution to compute the probability of deadlines misses, but we use it to guide us in the choice of budgets to assign to tasks so that deadlines are always met.

III. TIME VARIABILITY QUANTIFICATION

In this section we present our method to quantify the execution time variability of a program P. Let $S(P)$ be a set of execution times of program P. This set can be generated for example using repeated executions of the program in a specific configuration of the processor and/or using estimates of execution times for example in the worst and/or in the best case, however the way this set is computed is out of the scope of this paper. We note C_P the random variable taken its values in $S(P)$ with $p(C_P = x)$ being the probability of occurrence of x in $S(P)$. Note that we do not make any assumption concerning the distribution function of this random variable.

We define the execution time variability as being quantified by a statistical dispersion parameter. In Definition 1 we deliberately do not specify which statistical dispersion parameter to use as we want to remain flexible in the choice of the parameter to be used.

Definition 1 (Execution time variability): The execution time variability of a program P is defined by TV a statistical dispersion parameter of C_P the execution time random variable of program P.

Dispersion parameters are used in statistics to measure the tendency of the values of a distribution to be scattered on either side of a value. For example the coefficient of variation measures the dispersion of a set of data around the mean. The higher the value of the coefficient of variation, the greater the dispersion of the values around the mean. The skewness parameter [16] describes which side of the distribution has a longer tail. For unimodal distributions, symmetric distributions should have a skewness value near zero, negative skewness values means that data are more shifted towards the maximum value, positive skewness values means that data are more shifted towards the minimum value of the distribution [17].

In this work we propose our own dispersion parameter in Definition 2. Our goal is to provide a dispersion parameter that is not sensitive to the type of distribution the random variable follows. This parameter is inspired from the classical coefficient of variation. In our case, it measures the dispersion of a set of data around the largest value instead of the mean.

Definition 2 (Coefficient of variation to the maximum): The coefficient of variation to the maximum $VWCET$ of X a random variable with n occurrences $\{X_1, \dots X_n\}$ is:

$$VWCET = \frac{\sqrt{\frac{\sum_{i=1}^{n}(M-X_i)^2}{n}}}{M} * 100 \text{ with } M = \max_{1 \leq i \leq n} X_i \quad (1)$$

Given that the VWCET is based on the sum of deviations between the WCET and the other data values, the smaller it is the less likely the data is far away from the WCET.

To illustrate the VWCET and skewness parameters, Figure 1 represents histograms of execution times (in terms of number of cycles) of three sorting programs, merge sort, quick sort and insert sort of the Mälardalen [18] benchmark. The execution times of the sorting algorithms were measured in a bare-metal environment using the Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit. For every algorithm we use the same input and all the algorithms are executed in the same configuration of the processor i.e. the caches are disabled and one core executes the sorting program and all other cores execute the same program. We can see that the shape of the histograms are are left-biased, centered, or right-biased. For every program we compute the VWCET and skewness statistical parameters.

We can see in Table I that the the sKw value is close to zero for quick sort, negative for insert sort and positive for merge sort confirming the shapes of Figure 1. The $VWCET$ value of merge sort is the larger, then follows the $VWCET$ value of insert sort and the $VWCET$ quick sort is the smallest one.

IV. MIXED CRITICALITY BUDGET ASSIGNMENT PROBLEM

In this section we show how execution time variability can be exploited in the real-time scheduling problem of mixed criticality systems. The goal is to define the execution time budgets to be assigned to tasks in such a way that: (1) the real-time task set is schedulable, i.e. all the tasks meet their deadlines whatever their criticality (2) critical tasks terminate at the latest at their allocated budget and (3) less critical tasks maximize the instances where they terminate the latest at their

allocated budget. We consider that if a job of a task does not terminate after the execution of its assigned budget, the job is stopped. In our model the jobs of the less critical tasks stopped only if during the execution they exceed their budget, otherwise they are executed and are guaranteed to meet their deadlines. We consider a task set Γ of n mixed criticality real-time tasks executed using a scheduling algorithm $Sched$. We do not specify whether the execution platform is uniprocessor or multiprocessor, however the scheduling algorithm must be sustainable with respect to task execution times, i.e. if the task set is schedulable using algorithm Sched, the task set is still schedulable using algorithm Sched if the task execution times are smaller. Each task $\tau_i \in \Gamma$ is a tuple $(Budget_i, TV_i, L_i, D_i, T_i)$ with:

- $Budget_i$: set of possible budgets to be assigned to τ_i with $p_i(b)$ is the probability that the budget b is not exceeded
- TV_i: execution time variability of τ_i
- $L_i \in \{LO, HI\}$: τ_i criticality, LO less critical than HI
- $D_i \in \mathbb{N}^*$: relative deadline of τ_i
- $T_i \leq D_i$: task period i.e., its minimum inter-arrival time.

We consider that TV_i, the execution time variability, has been computed using given execution time random variables C_i. We suppose that the random variables C_i of all the tasks are independent. The independence is necessary for the computation of the scores of Equation 2. The set $Budget_i$ is a subset of possible budgets of execution times selected from the possible values of C_i. Each set of budgets of a task τ_i contains m possible budgets noted $\{b_{1,i}, \dots b_{m,i}\}$ ordered in a decreasing order of budgets with $b_{1,i} = WCET_i$ is the worst-case execution time of task τ_i with $p(b_{1,i}) = 1$.

Given a task set Γ of n mixed criticality real-time tasks, a budget assignment for Γ is a vector $B = (B_1, B_2, \dots, B_n)$ with $\forall i \in 1 \dots n, B_i \in Budget_i$ is the execution time budget assigned to task τ_i. We define Γ_B as the task set where every task τ_i is defined by (B_i, L_i, D_i, T_i) with B_i, L_i, D_i and T_i are the execution time, the criticality, the deadline and the period of τ_i respectively.

Given a task set Γ of n mixed criticality real-time tasks the score of a budget assignment $B = (B_1, B_2, \dots, B_n)$ computes the probability that all the tasks do not exceed their assigned budget and is defined as

$$Score(B) = \prod_{i \in 1 \dots n} p_i(B_i) \tag{2}$$

The score $Score(B_{LO}) = \prod_{\substack{i \in 1 \dots n \\ L_i \in LO}} p_i(B_i)$ is the score of the budget assignment of LO criticality tasks and $Score(B_{HI}) = \prod_{\substack{i \in 1 \dots n \\ L_i \in HI}} p_i(B_i)$ is the score of the budget assignment of HI criticality tasks.

Definition 3 (Mixed criticality schedulability): Given a task set Γ of n mixed criticality real-time tasks, the task set Γ is schedulable w.r.t. the budget assignment $B = (B_1, \dots, B_n)$ and the scheduling algorithm Sched if and only if:

1) The task set Γ_B is schedulable according to the scheduling algorithm $Sched$
2) $Score(B_{HI}) = 1$

V. Budget assignment heuristic

We consider the problem of finding the budget assignment of a task set Γ such that Γ is mixed criticality schedulable and the probability of respecting the budget assignment for low criticality tasks is maximized. An optimal solution to the problem is to assign to all the high criticality tasks τ_i their largest value as a budget and find the optimal assignment for low criticality tasks among m^{nbLO} possible assignments where $nbLO$ is the number of low criticality tasks. In this case the schedulability test of the scheduling algorithm is executed in the worst-case $O(m^{nbLO})$ times. The more the number of low criticality tasks grows, the longer the computation time is, the latter is also impacted by the complexity of the used scheduling test. For this reason, we propose a non optimal greedy heuristic, Algorithm 1, where the schedulability test is executed in the worst-case $O(m \times nbLO)$ times. In this algorithm the low criticality tasks are handled in a decreasing order of their execution time variability parameter. The idea is to start by reducing the budget of the tasks that are the less shifted towards their worst execution time. In Algorithm 1, first in (lines 2-5) if the task set with the minimal assignment budget is not schedulable the problem is not schedulable.

Otherwise a low criticality task with the highest execution time variability is selected. The budget for this task is decreased until the task set is schedulable (lines 8-14). If the task set is still not schedulable, the next task with the highest execution time variability parameter is selected.

Algorithm 1: Time variability heuristic

Input : $\Gamma = \{\tau_i, i \in 1..n\}$
Output: system is not schedulable or $B = (B_1, \dots B_n)$

1 $Set =$ set of all low criticality tasks
2 $\forall \tau_i$, if $\tau_i \in Set, B_i = b_{m,i}$ else $B_i = b_{1,i}$;
3 **if** Γ_B *not schedulable using algorithm Sched* **then**
4 return system is not schedulable
5 **end**
6 $\forall \tau_i \in Set, B_i = b_{1,i}$
7 **while** Γ_B *not schedulable using algorithm Sched* **do**
8 i is the index of τ_i with $\forall \tau_j \in Set, TV_i \geq TV_j$
9 **for** $r \in 2..m$ **do**
10 $B_i = b_{r,i}$,
11 **if** Γ_B *is schedulable using algorithm Sched* **then**
12 return $B = (B_1, \dots B_n)$;
13 **end**
14 **end**
15 τ_i is removed from Set;
16 **if** $Set = \emptyset$ **then**
17 return system is not schedulable ;
18 **end**
19 **end**
20 return $B = (B_1, \dots B_n)$;

We illustrate our budget assignment algorithm using the following example. Let τ_1, τ_2 be two LO criticality tasks and τ_3 a HI criticality task scheduled in a mono-processor according to rate monotonic scheduling algorithm. For every task τ_i, $S(\tau_i)$ is a set of 100 executions times of τ_i with:

- $S(\tau_1)$ is composed of 10 occurrences of value 1, 20 occurrences of value 2 and 70 occurrences of value 3.
- $S(\tau_2)$ is composed of 40 occurrences of value 1, 50 occurrences of value 2 and 10 occurrences of value 3.
- $S(\tau_3)$ is composed of 10 occurrences of value 1, 10 occurrences of value 2 and 80 occurrences of value 3.

The VWCET of τ_1 and τ_2 computed using Formula 1 are respectively $VWCET_1 = 0.258$, $VWCET_2 = 0.48$. We don't compute $VWCET_3$ because the assigned budget of HI criticality task do not depend on the variability of the task.

Let Γ be a mixed criticality task set, as defined in Section IV, composed by $\tau_1 = ((1,2,3), 0.258, LO, 6, 6)$, $\tau_2 = ((1,2,3), 0.48, LO, 9, 9)$ and $\tau_3 = ((1,2,3), VWCET_3, HI, 12, 12)$.

Using response time analysis [19] the worst-case probability deadline miss of τ_3 is 0.204 which is quite high for a high criticality task. We need to reduce LO criticality task execution times to ensure the schedulability. Algorithm 1 applied to Γ first assigns to all the tasks the budget 3. Then the task τ_2 is selected as it is the task with the largest time variability among the set of low criticality tasks. The budget of τ_2 is decreased until the budget 1 as the task set is schedulable with this budget. Algorithm 1 returns the set of budgets $B_1 = 3$, $B_2 = 1$ and $B_3 = 3$ for the tasks τ_1, τ_2 and τ_3 respectively. The score $Score(B_{HI}) = 1$ because the probability of not exceeding the budget $B_3 = 3$ is equal to 1 and $Score(B_{LO}) = 1 \times 0.4 = 0.4$ with 1 is the probability of not exceeding the budget $B_1 = 3$ and 0.4 is the probability of not exceeding the budget $B_2 = 1$. Note that the budgets computed with Algorithm 1 are the budgets with the optimal score.

VI. Simulation-based evaluation

In this section, we present simulation-based experiments to evaluate the proposed heuristic. We consider uniprocessor earliest deadline first algorithm. We can consider without loss of generality that all the tasks are of low criticality because all the algorithms we are comparing assign to the high criticality tasks WCET as budget. For each task we generate a maximal random utilization that ranges from 1 to 1.45 using [20]. We generate periods using a uniform distribution in the range $[4, 102]$ and a deadline using a uniform distribution in the range $[T_i/2, T_i]$. We generate the minimal random utilizations by reducing the maximal utilizations by a value that ranges between 1 and 45 percent. We use a truncated normal distribution, where the mean is a random (uniform) value ranging from the Best case execution time BCET to the WCET of the task, and the standard deviation is a random value proportional to the span of execution times $(WCET - BCET)/(x)$ where x follows a uniform distribution in the range $[2, 40]$. We generate 1000 task sets composed of 6 tasks. For each task τ_i, $Budget_i = \{WCET_i, b_1, b_2, Median_i\}$ where b_1 and b_2 are the $80th$ and $60th$ percentiles respectively and $Median_i$ is the median value. We discard task sets with $\sum_{i=1...n} \frac{BCET_i}{T_i} > 1$, and task sets which do not produce a solution with at least one algorithm in the list of the algorithms that we compare.

We generate three categories of task sets:

- Scenario 1: the first 80% of the task distributions have a skewness value greater than 2, the next 10% have a skewness value between 2 and -2, and the final 10% have a skewness value less than -2.
- Scenario 2: the first 10% of the task distributions have a skewness value greater than 2, the next 10% have a skewness value between 2 and -2, and the final 80% have a skewness value less than -2.
- Scenario 3: distributions are generated randomly.

We test Algorithm 1 in two cases:

- VWCET: Algorithm 1 with $\forall \tau_i$, $TV_i = VWCET(C_i)$,
- sKw: Algorithm 1 with $\forall \tau_i$, TV_i is the skewness of C_i.

We compare Algorithm 1 to the following algorithms that don't consider variability parameter as a criteria to select the task to reduce it's time budget:

- Medians: low tasks are assigned the median as a budget.
- Opt: optimal solution by testing all the configurations and applying the schedulability test $O(m^{nbLO})$ times.
- Periods: similar to Algorithm 1 but tasks are selected in an ascending order of periods in line 8.
- Deadlines: similar to Algorithm 1 but tasks are selected in an ascending order of deadlines in line 8.
- Random: similar to Algorithm 1 but tasks are selected randomly in line 8.

Figures 3, 4 and 5 depict the scores obtained by the different algorithms in the scenarios 1, 2 and 3 respectively suing box plot representation. The mean score is also represented in each figure. We conclude that:

1) Assigning all the budgets of all the tasks to their median value budget result in a score close to 0,
2) Using the execution time variability gives better score than using the periods, or deadlines or a random,
3) VWCET gives better scores than the skewness parameter
4) The difference between the scores of the optimal solution and the ones obtained by the heuristics have the same order of magnitude in all the scenarios. This suggests that our approach is not highly affected by the shape of execution times distribution.

In Figure 2, we compare the time needed to compute the set of assigned budgets of the optimal algorithm Opt and the Algorithm 1. We can see that from task sets with 8 tasks, the execution time of Opt increases exponentially.

VII. Benchmark-based evaluation

We performed our experiments on the Zynq UltraScale+ MPSoC ZCU104 platform, featuring a quad-core Arm Cortex A53 processor. We use FreeRTOS [21], an open source, small, scalable and well documented real-time OS, along with ESFree [22] as the scheduling library in order to incorporate advanced features that are not nativaly proposed by freeRTOS, such as

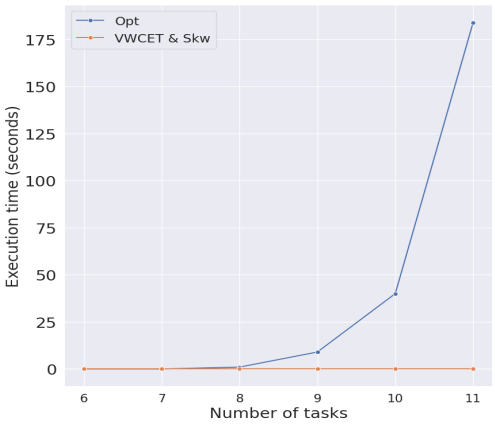

Fig. 2. Computation time of Opt compared to Algorithm 1

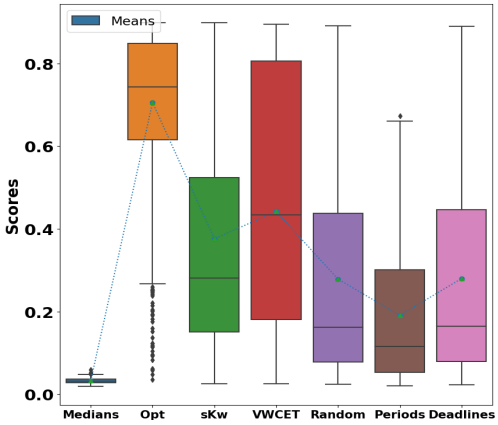

Fig. 3. Algorithms comparison for Scenario 1

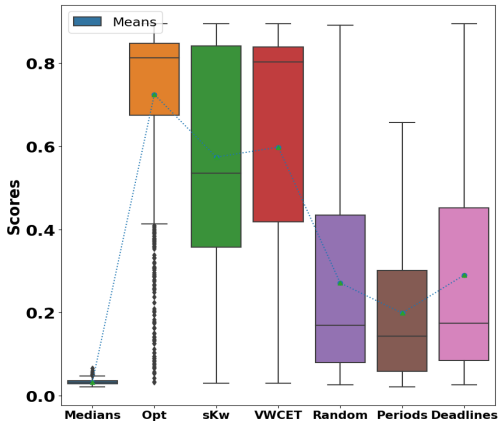

Fig. 4. Algorithms comparison for Scenario 2

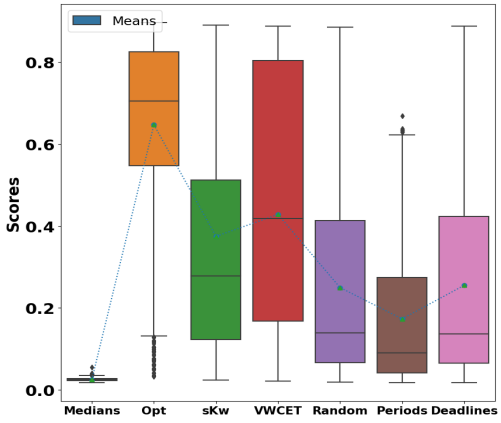

Fig. 5. Algorithms comparison for Scenario 3

timing error detection and handling mechanisms for scenarios involving task time budget excess and deadline misses. ESFree also provides an extended version of the FreeRTOS task control block (TCB) that includes additional information for managing periodic tasks such as current execution time, WCET and deadline of the task. FreeRTOS measures time using a tick count variable. The RTOS tick interrupt increments the tick count with strict temporal accuracy allowing the kernel to measure time to a resolution of the chosen timer interrupt frequency. In our case we choose a resolution of 1000 Hz, this translates into a time resolution of 1 ms, consequently execution time smaller than 1 ms cannot be measured. The tick interrupt calls an application defined hook function where we check which task is currently under execution and increase it's execution time variable by 1 ms. We generate a set of execution times by executing 6 programs from TACLeBench [23] and Mälardalen [18] benchmarks under the following configuration: (1) we consider asymmetric multiprocessing with FreeRTOS, meaning each core of a multicore processor runs its own independent instance of FreeRTOS (2) we do not vary the inputs of the programs (3) caches are disabled (4) cores 1, 2 and 3 execute [24] a program that iterates over a large array (5) core 4 executes the 6 programs and perform measurements to collect a set of execution times (6) the scheduling algorithm is preemptive rate monotonic.

Table II gives for every program its deadline and its period. All the programs are considered of LO criticality. For each program i, the budgets are $Budget_i = \{WCET_i, b_1, ..., b_7, Median_i\}$ where b_1, ... , b_7 are the $99th$, $97th$, $95th$, $90th$, $80th$, $70th$, $60th$ percentiles respectively and $Median_i$ is the median value.

Using the generated execution times we apply:

- VWCET: Algorithms 1 with $\forall \tau_i$, $TV_i = VWCET(C_i)$,
- sKw: Algorithms 1 with $\forall \tau_i$, TV_i is the skewness of C_i.

A response time analysis [19] is used as schedulability test. We compare Algorithm 1 to the same algorithms used in the simulation-based evaluation i.e. Medians, Opt, Periods, Deadlines and Random. Table III depicts the scores obtained for every program using every algorithm. The score Π is the score computed using Formula 2. We can see that Algo-

Prog	epic	fft	iSort	ludcmp	matrix	sha
T(ms)	450	430	70	440	677	460
D(ms)	152	18	10	52	390	247

TABLE II
PROGRAM'S PARAMETERS

Prog	Medians	Random	Periods	sKw	VWCET	Opt
epic	0.51	0.73	0.73	1	1	0.97
fft	0.50	0.99	0.71	0.80	0.71	0.99
iSort	1	1	1	1	1	1
ludcmp	0.51	0.72	0.72	0.72	0.72	0.98
matrix	0.50	0.72	1	1	0.80	0.90
sha	0.53	0.72	0.99	0.72	1	0.97
Π	0.03	0.27	0.37	0.42	0.41	0.83

TABLE III
ALGORITHMS SCORES

Program i	epic	fft	iSort	ludcmp	matrix	sha
$p((B_i)$ (VWCET)	1	0.71	1	0.72	0.80	1
$ratio_i$ (VWCET)	1	0.69	1	0.75	0.82	0.99
$p(B_i)$ (sKw)	1	0.80	1	0.72	1	0.72
$ratio_i$ (sKw)	0.99	0.84	1	0.75	1	0.72

TABLE IV
THE PROBABILITIES OF MEETING THE EXECUTION TIME BUDGET

rithm 1 using variability parameters sKw and VWCET yields better results than the other algorithms Medians, Random and Periods. When comparing the optimal solution computed using algorithm Opt to the proposed heuristic that uses time variability we can see that the optimal solution in order to maximize the overall scores of the tasks reduces the time budget of 5 of the 6 tasks, in difference to the proposed heuristic where 3 of the 6 tasks will always meet their time budget. Using algorithm Period only 2 of the 6 tasks will always meet their time budget. The greater the number of tasks whose jobs always meet the assigned budget, the fewer tasks we need to monitor during execution. Using FreeRTOS OS and the library ESFree, we executed the 6 programs during 10 minutes using rate monotonic preemptive scheduling algorithm. If a job do not terminate at its assigned budget computed using Algorithm 1, the job is stopped. We count for every task τ_i the number of stopped jobs and computes $ratio_i$ the ratio between the number of stopped jobs and the number of activated jobs of task τ_i. Table IV presents the comparison of the probability of meeting the execution time budget per task to the ratios obtained. For VWCET the observed task performance aligns well and sometimes slightly exceeds the theoretically computed time budgets, this conclusion holds true for skewness as well. We can deduce that the overhead induced when stopping a job is not high.

VIII. CONCLUSION

This paper introduces an approach for the scheduling problem of mixed criticality systems. We show that the time variability of execution times can be used to compute the execution time budget to be allocated to low-criticality tasks in order to prevent these tasks from disrupting the functioning of high-criticality tasks, while ensuring that low-criticality tasks are not always stopped. We show that the proposed heuristic outperforms the budget selection strategies that are agnostic to the shape of the distribution of execution time budgets. In the future, we plan to apply our approach in an online algorithm where task budgets are adjusted during the execution.

REFERENCES

[1] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *28th IEEE Real-Time Systems Symposium (RTSS 2007), Tucson, Arizona, USA*, pp. 239–243.

[2] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 82:1–82:37, 2018.

[3] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan, "Variability in the execution of multimedia applications and implications for architecture," *SIGARCH Comput. Archit. News*, vol. 29, no. 2, p. 254–265, may 2001.

[4] W. Koch, R. Mancuso, and A. Bestavros, "Neuroflight: Next generation flight control firmware," *CoRR*, vol. abs/1901.06553, 2019.

[5] V. Nélis, P. M. Yomsi, and L. M. Pinho, "The Variability of Application Execution Times on a Multi-Core Platform," in *16th International Workshop on Worst-Case Execution Time Analysis (WCET 2016)*, M. Schoeberl, Ed., vol. 55, Dagstuhl, Germany.

[6] J. Bin, "Controlling execution time variability using cots for safety-critical systems," Thesis, Université Paris Sud - Paris XI, Jul. 2014.

[7] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for real-time systems," *Leibniz Trans. Embed. Syst.*, vol. 6, no. 1, pp. 03:1–03:60, 2019.

[8] M. Alcon, H. Tabani, L. Kosmidis, E. Mezzetti, J. Abella, and F. J. Cazorla, "Timing of autonomous driving software: Problem analysis and prospects for future solutions," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020, pp. 267–280.

[9] F. Reghenzani, G. Massari, and W. Fornaciari, "Timing predictability in high-performance computing with probabilistic real-time," *IEEE Access*, vol. 8, pp. 208 566–208 582, 2020.

[10] A. Burns and S. Baruah, "Towards a more practical model for mixed criticality systems," in *Proc. WMC, RTSS*, 2013, pp. 1–6.

[11] S. K. Baruah, A. Burns, and Z. Guo, "Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors," in *28th Euromicro Conference on Real-Time Systems, (ECRTS) 2016, Toulouse, France*, pp. 131–138.

[12] X. Gu and A. Easwaran, "Dynamic budget management with service guarantees for mixed-criticality systems," in *IEEE Real-Time Systems Symposium, (RTSS), Porto, Portugal, 2016*, pp. 47–56.

[13] J. Singh, L. Santinelli, F. Reghenzani, K. Bletsas, D. Doose, and Z. Guo, "Mixed criticality scheduling of probabilistic real-time systems," in *Dependable Software Engineering. Theories, Tools, and Applications - 5th International Symposium, (SETTA) 2019, Shanghai, China, 2019*, N. Guan, J. Katoen, and J. Sun, Eds., vol. 11951, pp. 89–105.

[14] Y. Abdeddaïm and D. Maxim, "Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems," in *Design, Automation & Test in Europe Conference & Exhibition, (DATE), Lausanne, Switzerland, 2017*, D. Atienza and G. D. Natale, Eds., pp. 596–601.

[15] S. Draskovic, R. Ahmed, P. Huang, and L. Thiele, "Schedulability of probabilistic mixed-criticality systems," *Real Time Syst.*, vol. 57, no. 4, pp. 397–442, 2021.

[16] P. v. Hippel, *Skewness*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1340–1342.

[17] N. I. of Standards and Technology, *Measures of Skewness and Kurtosis*.

[18] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The Mälardalen WCET benchmarks – past, present and future," B. Lisper, Ed. Brussels, Belgium: OCG, Jul. 2010, pp. 137–147.

[19] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software engineering journal*, vol. 8, no. 5, pp. 284–292, 1993.

[20] P. Emberson, R. Stafford, and R. I. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pp. 6–11.

[21] R. Barry *et al.*, "Freertos," *Internet, Oct*, vol. 4, 2008.

[22] R. Kase, "Efficient scheduling library for freertos," 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:64508990

[23] H. Falk, S. Altmeyer, P. Hellinckx, B. Lisper, W. Puffitsch, C. Rochange, M. Schoeberl, R. B. Sørensen, P. Wägemann, and S. Wegener, "Taclebench: A benchmark collection to support worst-case execution time research," in *Workshop on Worst-Case Execution Time Analysis, (WCET), 2016, France*, M. Schoeberl, Ed., vol. 55, pp. 2:1–2:10.

[24] M. Bechtel and H. Yun, "Denial-of-service attacks on shared cache in multicore: Analysis and prevention," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019.