Measurement-Based Probabilistic Timing Analysis: Lessons from an Integrated-Modular Avionics Case Study

Franck Wartelψ, Leonidas Kosmidis†,* , Code Lo*, Benoit Triquetψ, Eduardo Quiñones†, Jaume Abella†, Adriana Gogonel*, Andrea Baldovinψ, Enrico Mezzettiφ, Liliana Cucu*, Tullio Vardanegaφ, Francisco J. Cazorla†,†

ψAirbus, France, †Barcelona Supercomputing Center, Spain, *Universitat Politècnica de Catalunya, Spain
φINRIA, France, φUniversity of Padua, Italy, †Spanish National Research Council (IIIA-CSIC), Spain

Abstract—Probabilistic Timing Analysis (PTA) in general and its measurement-based variant called MBPTA in particular can mitigate some of the problems that impair current worst-case execution time (WCET) analysis techniques. MBPTA computes tight WCET bounds expressed as probabilistic exceedance functions, without needing much information on the hardware and software internals of the system. Classic WCET analysis has information needs that may be costly and difficult to satisfy, and their omission increases pessimism. Previous work has shown that MBPTA does well with benchmark programs. Real-world applications however place more demanding requirements on timing analysis than simple benchmarks. It is interesting to see how PTA responds to them. This paper discusses the application of MBPTA to a real avionics system and presents lessons learned in that process.

I. INTRODUCTION

The size of software programs used in aircraft on-board processors, reported in Figure 1 courtesy of Airbus [1], has increased exponentially in the past, causing an equivalent increase in the demand for processing power. This trend is predicted to continue in the future owing to the use of new, complex and hard real-time functionalities for flight control, collision avoidance, auto-piloting. A similar trend has been observed in other application domains. The strict real-time nature of several software programs in those domains causes a growing demand for guaranteed performance, which is vital to assert before deployment.

The timing behaviour of numerous hardware and software components seen in isolation can be abstracted into simple deterministic worst-case models. This quality is much desired by state-of-the-art static timing analysis (STA) techniques, which can thus be applied to them with good cost-benefit ratio. The situation changes when those individual components are assembled into larger units. As long as the quantity of and the interaction among the involved components remain small, the resulting system can continue to be submitted to STA with good-quality results. However, when the balance changes, as it is likely the case for future systems, modelling system behaviour as it results from the interaction of multiple independent components becomes exceedingly complex: it is then that state-of-the-art STA hits the wall [2][3]. Any loss of accuracy in, or plain lack of, knowledge on the state of inner components with bearing on the timing behaviour of interest yields substantial pessimism: this reduces the guaranteed performance of the system. This challenge presents industry with a lose-lose choice: either increase the number of hardware units to compensate for the lesser guaranteed performance per unit, which breaks economy of scale, or endure the possibly prohibitive costs of seeking the missing information needed to mitigate pessimism in computing bounds.

Probabilistic timing analysis (PTA) [4][5][6] has recently emerged as an attractive alternative. PTA considers timing bounds in the same manner as the embedded safety-critical systems domain addresses system reliability, which is expressed as a compound function of the probabilities of hardware failures and software faults. PTA extends this probabilistic notion to timing correctness. PTA seeks WCET bounds for arbitrarily low probabilities, so that even if violation events may in principle occur, they would only do with a probability well below the one specified by the system safety requirements.

In this paper we focus on the measurement-based variant of PTA, called MBPTA [6]. With MBPTA, end-to-end runs of the program are carried out on the target hardware. The information obtained from the measurement runs is processed to determine the upper extreme execution time distribution of the program (hence not a number as with STA, but a distribution profile, also known as probabilistic WCET, pWCET). For this reason, MBPTA has low information cost, while being able to achieve sound results so long as the processor hardware has a probabilistically characterisable timing behaviour [4][6]: this requirement is not met by conventional processors but, in other ongoing work, we argue it can be met without excess cost and complexity.

PTA has been shown to work well with reference benchmarks such as EEMBC [7] and Mälardalen [8], in the first natural step to assess the goodness of a new analysis technique. However, the challenge that those benchmarks pose in terms of code size and complexity does not level with real-world applications, and the gap can be large and frustrating for the industrial users.

This paper makes the following distinct contributions:
1) it demonstrates that the requirements posed by PTA can be met for real industrial systems by showing that an unmodified avionics application binary can be run on a standard processor modified only in the timing behaviour of some of its internal resources;

2) it presents the individual steps of the MBPTA technique adopted in this work and how they collectively produced the required pWCET estimates;

3) it discusses issues arising in the analysis process with regard to the control of the essential parameters; and

4) it relates the pWCET estimates obtained from MBPTA to the way industrial practice determines WCET bounds, also studying the effect exercised by the modifications required in the processor internals.

The application case we chose to that end handles data concentration and maintenance of the flight control computers, and it is built on top of an ARINC 653 Operating System. We applied MBPTA to selected application functions as well as to the partition switch procedure, which resides in the operating system space. Our results show that the cumulative cost of performing PTA on the selected programs is very low: providing a pWCET estimate for each of the case-study programs took less than 1 hour between preparation, measurement run and processing of the observations in our experimental environment (and would take less than 1 minute in a real platform). The procedure requires virtually no information on the hardware and software internals, which makes it very attractive for industrial users. The resulting pWCET bounds are sound by construction and much tighter than the previous bounds used by the application owners.

The remainder of this paper is organised as follows. Section II presents the Integrated Modular Avionics system selected as case study. Section III introduces PTA and MBPTA. Section IV describes the experimental set-up used for the MBPTA observation runs. Section V presents the results obtained with MBPTA on the case study. Section VI reviews some related work. Section VII draws some conclusions.

II. AVIONICS CASE STUDY

In the past, conventional avionics systems were based on the federated architecture paradigm. In those systems each computer is a fully dedicated unit, which allows local optimisations. However, since most of the federated computers perform essentially the same functions (input acquisition, processing and output generation), a natural optimisation of resources is to share the development effort by identifying common subsystems, standardising interfaces and encapsulating services; in other words, adopting a modular approach. That is the intent of the Integrated Modular Avionics (IMA) concept, whose integration refers to the sharing of (platform) resources for use by multiple subsystems. Federated architectures have thus been replaced by Integrated Architectures in which the same computer can host multiple applications, potentially operating at distinct criticality levels.

Although this paradigm shift comes from the IMA concept inbred to the avionics community, it is much applicable to other application domains. In the automotive sector, for example, software components can be supplied from multiple sources, integrated on the same hardware platform or physically distributed and possibly moved from one CPU to another without loss of functional and time correctness, while also providing a guaranteed level of reliability.

A. IMA Concepts

The experiments we run were made on an avionics application that performs data concentration and maintenance of the flight control computers. The application runs on top of an ARINC 653 Operating System [9]. The system architecture pursues temporal partitioning and spatial partitioning:

- Temporal partitioning allows partitions to execute without affecting one another temporally. It warrants strict allocation of CPU time to partitions, according to a static schedule computed offline, where a partition is the scheduling unit. The MA jor Frame is the hyper-period of all partitions and it is broken down in an integral number of MI nor Frames of equal period and duration, in which partition scheduling occurs. A Partition Time Window represents an integral time duration during which the Operating System exclusively schedules processes belonging to a given partition. Each partition is granted at least one Partition Time Window in every MA jor Frame. Processes belonging to a given partition can only be scheduled in the Partition Time Windows of that partition.

- Spatial partitioning allows partitions to execute without affecting one another spatially. This is achieved by prohibiting (Read, Write or Execute) memory accesses from a partition outside of the memory areas statically allocated to it.

B. Case Study

The application program considered in the case study is illustrated in Figure 2. Its activity consists of five functional blocks, each of which is a high-level procedure, root of a finite acyclic call graph which changes at MI jor Frame boundary:

- Functional blocks FUNC1 acquire raw data from the I/O ports. Data are acquired through the A653 READ_SAMPLING_MESSAGE API service.

- Functional blocks FUNC2 deserialise raw data. Raw payload data are parsed and allocated to global variables used in the subsequent steps.

- Functional blocks FUNC3 perform data processing. These functions are automatically generated from a SCADE [10] model.

- Functional blocks FUNC4 serialise the processed data.

- Functional blocks FUNC5 post the output data to external physical I/O ports.

The real application is hosted on a computer board embedding a MPC 755 processor, with L1 data cache operating in copy-back mode. State-of-the-art STA could not be performed on it within sustainable bounds of engineering effort and containment of pessimism. The greatest difficulties were caused by the caches operating in copy-back mode, and by the massive presence of interfering Operating System code in the application space – typical for ARINC 653 implementations –, which are overly difficult to analyse by application-level state-of-the-art STA tools.

The results presented in this paper were obtained by running the application on a processor simulator composed of a PowerPC-family MPC755 instruction set and a pipeline
emulator attached to a time-accurate cache, modified to meet PTA requirements. The original application was run on a top of an ARINC 653 compliant operating system prototype that provided zero-disturbance constant-time services [11], which proved a good facilitator to application-level timing analysis.

The processing in FUNC4 and FUNC5 is analogous to that in FUNC2 and FUNC1 respectively, so is their timing behaviour. For this reason we do not discuss FUNC4 and FUNC5 further in this paper. We focus instead on FUNC1, FUNC2, and FUNC3 for the application, and on partition switch, from the perspective of the operating system.

III. MBPTA

PTA has recently emerged as a viable alternative to state-of-the-art timing analysis techniques. PTA provides a probabilistic WCET (pWCET) estimate as a cumulative distribution function (CDF) that upper-bounds the highest execution times of the program under study. The Inverse CDF (ICDF or \(1 - \text{CDF}\)) computed from the CDF of the pWCET estimate guarantees that the timing behaviour of the program may only exceed the given bound with a probability lower than an exceedance threshold expressed as \(10^{-x}\) for some \(x\) per activation of that program or, equivalently, hour of operation.

The MBPTA [12][5][13][4][6] variant of PTA constructs the pWCET by collecting observations of the program’s execution time. In its simplest form, given a set of \(R\) runs, the ICDF is computed by providing the probability of occurrence of each of the observed execution times. However, the ICDF built in this way provides execution time estimates that only have associated probabilities down to \(1/R\). Techniques such as Extreme Value Theory (EVT) [14][6] are used to provide values with much smaller associated probabilities. MBPTA techniques that are based on EVT provide pWCET estimates for arbitrarily low target probabilities (e.g., \(10^{-20}\) per hour).

Figure 3 shows the pWCET generated from a collection of 1,000 execution time observation runs of a sample program. The solid line shows the ICDF function resulting from such an execution time collection, which yields probabilities up to \(10^{-3}\) per execution. The dashed line shows the pWCET estimate produced applying MBPTA with EVT. Unlike STA, which provides a single WCET bound for a given program, PTA provides a probabilistic WCET function whose tail the user can cut at the level required for the problem at hand; in Figure 3 this is shown as placing the exceedance probability at \(p_x\) which corresponds to a pWCET bound at \(WCET_x\).

A. MBPTA Requirements

Notably, the utilisation of EVT with MBPTA requires that the observed execution times can be described by random variables that are proven independent and identically distributed (i.i.d.) [6]. As a result, the applicability of MBPTA [6] rests on the key assumption that all sources of execution time variation in the system must be either statically upper-bounded or probabilistically characterised. As we showed in [6] for different benchmarks, this assumption holds by randomising the timing behaviour of the hardware resources whose latency is too high to upper-bound [4] and by upper-bounding those for which the incurred pessimism is acceptable. Consequently – short of timing anomalies, which we discuss in Section IV – the only events that can cause (local and global) variations in execution times are indeed random.

As we discuss in Section IV, in our processor architecture we only had to modify the instruction and data cache and the TLB (translation look-aside buffer) so that their timing behaviour was randomised. Designing “PTA-friendly”processors that meet the above requirements in a manner that avoids costly modifications to commercial off-the-shelf processors is a hot research topic: recent results are promising as set-associative time-randomised caches have been proven feasible [15] and software randomisation has been proven as a valid alternative for use on top of deterministic caches [16].
IV. Experimental Setup

The processor architecture considered in this case study fulfils the PTA requirements and avoids timing anomalies by construction [17]. In particular, our processor features a 4-stage in-order pipelined core architecture with separate TLB and two levels of caches for instructions and data (see Figure 4). With in-order issue and finalisation we prevent instructions from using resources out-of-order in the final stage of the pipeline, which could cause timing anomalies otherwise.

The cache system is composed of two separate first level instruction and data 8-way set-associative caches of 32 KB in size and 32-byte cache line each, with random placement and replacement policies, operating in write-through mode [15], as well as a unified second level of 8-way set-associative cache of 64 KB in size and 32-byte cache line, operating in copy-back mode, with random placement and replacement policies.

For memory we have 4 KB pages and two separate instruction and data 2-way set-associative TLB with 128 entries each, with random placement and replacement.

TLB and caches are accessed in parallel so that instructions are stalled in the corresponding stage until both cache and TLB can serve the request.

The use of time-randomised cache and TLB designs allows attaching a probability to any latency for each processor instruction at any stage of execution. That is, the latency of the fetch stage depends on whether the access hits or misses in the instruction caches and TLB: for instance, a cache and TLB hit has a 1-cycle latency, whereas TLB and cache misses have a 100-cycle latency. After the decode stage, memory operations access the data cache and TLB analogously to the fetch stage.

Consequently, the latency of a non-memory instruction depends on whether it hits or not in the instruction cache and TLB, and the particular (fixed) execution latency of the operation (e.g. integer additions take 1 cycle). The latency for memory instructions depends instead on the hit/miss behaviour in both instruction and data caches and TLB.

The use of random placement and replacement policies in caches and TLB attaches a distinct probability to hits and misses [4]. As a result, the observed execution times meet MBPTA requirements by construction.

V. Results

We can now present the results of applying MBPTA with EVT to the three functions (FUNC1, FUNC2 and FUNC3) outlined in Section II as well as to the Partition Switch procedure (PS) that is a key part of the software architecture. We also consider average execution time to study how the processor modifications affect average behaviour.

The measurements we report have been obtained on a cycle-accurate simulator based on SoCLib [18], with PowerPC binaries [19], modelling the processor architecture presented in Section IV.

A. pWCET Analysis

The MBPTA technique follows the five steps depicted in Figure 5 and briefly recalled below:

1) Collecting observations: The first step consists in gathering a given number of execution time observations from end-to-end runs of each function under analysis. In case further observations were required to be able to compute the pWCET distribution (as we explain in the following steps), additional \( N_{\text{delta}} \) observations are made and included in the data collection submitted to the next steps. In our experiments we started collecting 100 execution time observations, with \( N_{\text{delta}} = 50 \). At every collection round, we checked whether the data are described by i.i.d. random variables: the two-sample Kolmogorov-Smirnov (KS) test [20] evaluates the fulfillment of the identical distribution property, and the runstest [21] the fulfillment of the independence property. These verification steps are mandatory for MBPTA with EVT.

Table I shows the results of the i.i.d. tests for all programs under analysis, for a number of observation runs sufficient to derive the pWCET distribution (see Table IV). In order to apply the KS test, we created two smaller samples of \( m = 50 \) and \( m = 100 \) elements, by randomly taking sequences of \( m \) consecutive elements from the original sample [6]. We then applied the two-sample KS test to the two smaller samples to prove that the distribution does not change over time. We note that in all cases the p-value obtained from the data for the KS test is higher than the required threshold, indicating that data are identically distributed. We note that all the data collected from our observation runs also pass the independence test.

2) Grouping: From the data collected in the previous step we pick the high-watermark values. We do so as we use EVT to upper-bound the probability of occurrence of the highest execution times and not the average behaviour. In this work we used the Block Maxima method [6] to convert the set of collected observations into a worst-case distribution fit for EVT. Block Maxima randomly picks execution times from the whole set of data collected into groups of a given size and considers the maximum value of each group. We use a block size of 50 as it provides a good balance between accuracy and number of executions required by the method.

Table I. p-value for the identical distribution (considering two samples of \( m = 50 \) and \( m = 100 \) elements) and independence test.

<table>
<thead>
<tr>
<th></th>
<th>Identical Distribution</th>
<th>Independence</th>
<th>Passed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNC1</td>
<td>( m=50 )</td>
<td>0.77</td>
<td>0.85</td>
</tr>
<tr>
<td>FUNC2</td>
<td>( m=50 )</td>
<td>0.69</td>
<td>0.80</td>
</tr>
<tr>
<td>FUNC3</td>
<td>( m=50 )</td>
<td>0.52</td>
<td>0.68</td>
</tr>
<tr>
<td>PS</td>
<td>( m=50 )</td>
<td>0.11</td>
<td>0.07</td>
</tr>
</tbody>
</table>
3) **Fitting:** Next we derive an EVT distribution from the set of execution times generated in the **Grouping** step. Such a distribution is denoted by $F$, which represents the common distribution function of the maximum of the $n$ random variables describing the observed execution times. $F$ is characterised by the shape ($\xi$), scale ($\sigma$) and location ($\mu$) parameters as follows:

$$F_{\xi}(x) = \begin{cases} 
  e^{-\frac{(x-\mu)}{\sigma\xi}} & \xi \neq 0 \\
  e^{-\frac{x-\mu}{\sigma \xi}} & \xi = 0 
\end{cases}$$

EVT requires that the data being analysed correspond to independent and identically distributed random variables (which we check in the **Collecting** step). On that condition, the maximum of those random variables converges to one of three possible EVT distributions: Gumbel, Frechet or Weibull. Of those, the Gumbel distribution, which has shape parameter $\xi = 0$, has been proven in [6][5] to fit well the problem of WCET estimation. In our application of EVT we use the exponential tail (ET) test [22] to validate that the distribution fits a Gumbel distribution. The decision of ET test is based on the calculation of a parameter $\hat{q}_{ET,n} = \frac{d_n - \sigma_n \ln(m_n)}{\sqrt{m_n}} (z_1 - \alpha/2)$, where $F_{\hat{d}_{n-1}}$ is the empirical distribution of the data to test. The test rejects the hypothesis that the data follow a Gumbel distribution if $\hat{q}_{ET,n} \notin [ICI,SCI]$, where $ICI$ and $SCI$ are calculated using parameters related to data to test with an error risk of $5\%$ as follows. We have $ICI = \hat{q}_{ET,n} + d_n - \sigma_n \ln(m_n) (z_1 - \alpha/2)$ and $SCI = \hat{q}_{ET,n} + d_n + \sigma_n \ln(m_n) (z_1 + \alpha/2)$. For the data to test we obtain $n$ as the number of execution traces contained in the data set, $d_n$ the first order through approximation of this data, $\sigma_n$ the empirical average of excesses, $m_n$ the number of excesses and $z_1 - \alpha/2$ the quantile of order $1 - \alpha/2$.

According to table II, we have $\hat{q}_{ET,n} \in [ICI,SCI]$ and we accept that the data follow a Gumbel distribution with a risk of error of $5\%$.

4) **Convergence:** Finally we determine when we have collected enough samples (minimum number of runs, MNR) such that the EVT results are not subject to modifications if more samples were added. To do so, for all rounds subsequent to the first, the distribution obtained by EVT for the current round is compared to the result of the previous round. To compare any two such distributions we use the continuous rank probability score (CRPS) metric, defined as $\sum_{i=1}^{\infty} \left[ f_X(i) - f_Y(i) \right]^2 [21]$, where $f_X(i)$ and $f_Y(i)$ are the distribution functions from the current and the previous round respectively. If the CRPS metric reports values below a given threshold, we consider that the current EVT distribution converges to the real pWCET distribution so that no more observations need to be collected. It is important to remark that the smaller the CRPS threshold is set, the more precise the distribution gets, at the cost of more observations and EVT projections. In our experiments we considered a difference threshold of 0.001 as shown in figure 6. If the distribution had not yet converged instead, the process would return to the first step and collect $N_{delta}$ more observations. Notably, passing the ET test ensures that the method always converges [6].

5) **Tail Extension:** The resulting EVT distribution (a Gumbel distribution here) is used to compute the pWCET estimate associated to any exceedance probability threshold. Figure 7 shows the pWCET estimates we obtained for the FUNC1 (a), FUNC2 (b), FUNC3 (c) and Partition Switch (d) procedures. As explained in [6], for multi-path programs (as FUNC3 in our experiments) the pWCET obtained from the MBPTA-EVT method only holds for the set of paths traversed in the measurement runs$^1$. The path coverage obtained for FUNC3 was deemed sufficient by our industrial cognizant.

As the hypotheses needed for the application of MBPTA were showed to be fulfilled by programs running on a PTA-friendly architecture, the case-study application was run on a processor simulator that exhibits the required features. At that point, the only practical solution for a meaningful comparison between our setting, in processor architecture and analysis method, and a traditional alternative was to set the simulator cache to operate with modulo placement and least recently used (LRU) replacement, and to apply the common industrial practice of taking the highest measurement for the program of interest and adding an engineering margin to it as determined by in-house knowledge of the system.

The bounds we obtained making the same application runs on the simulator with caches configured in deterministic LRU mode are shown as dashed vertical lines in Figure 7. We can see that pWCET estimates are significantly below a traditional LRU+margin$\%$ WCET estimate, where the additional margin

---

$^1$Note that, as opposed to MBTA methods on top of deterministic architectures, MBPTA on top of PTA-friendly architectures has no dependence on the particular mapping of data and instructions in memory.
determined by engineering judgment is usually chosen around 20%. As shown in Figure 7, pWCET estimates are slightly above the execution times obtained with the traditional processor with LRU replacement (between 0.1% and 3.8%), so they are between 16% and 20% below the WCET estimates for LRU+20%.

Currently, for commercial airborne systems at the highest integrity level (DAL-A), the maximum allowed failure rate in a system component is $10^{-9}$ per hour of operation [23]. To translate that failure rate requirement into the equivalent exceedance probability threshold, we need to know the frequency at which jobs are released. Thus, if we consider that tasks under analysis are released with a frequency of $10^{-2}$ seconds (i.e. $10^2$ activations per second), the pWCET of that task should have an exceedance probability in the range $[10^{-14}, 10^{-15}]$: $10^{-9} \frac{\text{failures}}{\text{hour}} / (3600 \times 10^2 \text{ task activations})$. Therefore, an exceedance probability threshold of $10^{-15}$ suffices to achieve the highest integrity level.

For the sake of illustration, we set our range of probabilities of interest to lie in the interval $[10^{-13}, 10^{-16}]$. Table III shows the pWCET estimates increment when increasing the exceedance probability from $10^{-10}$ to $10^{-13}$ and $10^{-16}$ (corresponding to a failure rate per hour of $10^{-5}$, $10^{-8}$ and $10^{-11}$ respectively), highlighting that, as far as the functional blocks of the application under test are concerned, reasonably low extra time must be taken into account to decrease the exceedance probability threshold.

**B. Cost of Application**

It is crucial for industrial users to gauge the cost of applying the method. Determining the MNR for an application with respect to a target platform is the most consuming step of MBPTA. On the one hand, determining the MNR is an iterative process that requires to compute multiple EVT distributions to calculate the CRPS. The cost of computing an EVT distribution from data fulfilling the i.i.d. property includes the cost of applying block maxima and of characterising the EVT distribution parameters ($\xi, \sigma$ and $\mu$). For the sake of illustration, the R-script that we have developed for analysis takes 2 seconds to derive an EVT distribution from a collection of 1,000 observations, running on an Intel Core i7 @ 2.80 GHz with 8 GB of RAM.

The number of runs of the program under analysis on the target platform and the number of times that an EVT distribution needs to be computed depend on the MNR. In our experiments we started collecting 100 execution time observations, with $N_{\text{delta}} = 50$. The monitored functions required in the order of a few hundreds of runs, as illustrated in Table IV.

The computation of the pWCET distribution for the programs considered in the experiments required less than 3 hours of run time of our simulator, hosted on the computer where the R-script was subsequently run. If the needed observation runs were made on a real PTA-compliant processor operating at 500 MHz instead, then the pWCET distribution of all the programs could be computed in less than 1 minute. We may therefore conclude that the MBPTA technique reduces considerably the information costs incurred by other timing analysis methods.

**C. Analysis of Average Performance**

Figure 8 compares the density function of the observed execution times when running the programs under study on a processor equipped with the time-randomised cache (continuous curve) implementing random placement and replacement policies against a deterministic cache implementing modulo placement and LRU replacement policies (dashed vertical line), under exactly the same execution conditions. As shown in Table V and not surprisingly, the deterministic cache achieves better average performance on account of better average preservation of locality, however the overhead on average performance introduced by time-randomised cache is very low, actually below 1%. We therefore maintain that the minor average performance loss caused by the time-randomised cache for the selected application is compensated by the low cost of applying probabilistic timing analysis.

Figure 7 also relates the average execution time obtained with the deterministic cache configuration against the pWCET

---

1. **TABLE III.** pWCET INCREASE WHEN RAISING THE EXCEEDANCE PROBABILITY FROM $10^{-10}$ TO $10^{-13}$ AND $10^{-16}$, WHICH CORRESPONDS A FAILURE RATE PER HOUR OF $10^{-5}$, $10^{-8}$ AND $10^{-11}$ RESPECTIVELY.

<table>
<thead>
<tr>
<th>Program</th>
<th>FUNC1</th>
<th>FUNC2</th>
<th>FUNC3</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Rate</td>
<td>$10^{-13}$</td>
<td>$10^{-10}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNC1</td>
<td>0.03%</td>
<td>0.03%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNC2</td>
<td>0.06%</td>
<td>0.07%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNC3</td>
<td>0.81%</td>
<td>0.87%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>0.26%</td>
<td>0.28%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. **TABLE IV.** MINIMUM NUMBER OF RUNS (MNR) PER PROGRAM.

<table>
<thead>
<tr>
<th>Program</th>
<th>FUNC1</th>
<th>FUNC2</th>
<th>FUNC3</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNR</td>
<td>600</td>
<td>250</td>
<td>300</td>
<td>250</td>
</tr>
</tbody>
</table>

---

2. [http://www.r-project.org/](http://www.r-project.org/)
estimates computed at an exceedance probability threshold of $10^{-16}$ using the random cache design. From those plots we see that MBPTA produces tight bounds. As shown in Table VI the pWCET are slightly higher but close enough to the maximum observed execution time with deterministic caches.

VI. RELATED WORK

This paper belongs to the realm of probabilistic real-time analyses. A probabilistic real-time analysis concerns systems that have at least one parameter described by a probability distribution. The work related to probabilistic real-time analyses may be classified into two main strands. The first strand concerns the schedulability analysis of probabilistic real-time systems. The second one the determination of probability distributions for the parameters.

For the first strand, since the seminal paper of Lehoczky [24], different results have been proposed for the schedulability of systems with probabilistic execution times [25] or probabilistic arrivals [26], taking into account the preemptions [27], architectures based on CAN [28], for hierarchical scheduling [29] or for resource reservation [30]. Some other work consider only calculation based on average values and bounds [31] as well as on real-time calculus [32]. Optimal algorithms for these systems are recently studied [33]. Operations between probability distributions may have high complexity and re-sampling techniques are usually used to low this complexity [34], [35].

For the second strand, since the seminal paper of Edgar [12], different techniques provide probability distributions mainly for the probabilistic (worst-case) execution time [5], [13], [6] in general or for component-based systems [36]. Probabilistic arrivals are studied recently [37], [38].

Our work belongs to the second group and it concerns the proposition of probability distributions for probabilistic worst-case execution time. This work presents the utilisation of the method originally presented in [6] on an industrial case study and it proves its scalability in this precise case.

VII. CONCLUSIONS

We have shown that applying MBPTA to the selected IMA-based application produces tight pWCET estimates, allowing increased system utilisation as required for new-generation avionics systems. We have also shown that the cost of computing a pWCET bound with MBPTA on a PTA-friendly architecture would be in the region of minutes per program (currently a few hours in our simulation environment), which is a very competitive cost for use in production.

While this paper presented the use of MBPTA against a specific avionics application, the results we obtained with it enable us to draw general conclusions that should apply for other systems in the same or other domains:

MBPTA shows no scalability issues since it imposes no particular constraints on the size of the program under analysis. MBPTA requires some hundreds of measurement runs, which cause the analysis cost to grow linearly with the average execution time of the program.

Once the execution time observations have been collected, applying the MBPTA method itself takes just a few seconds and that cost is independent on the input observations provided. As the observations made in the process describe some behaviour of the system, the user of MBPTA must ensure that the execution conditions under which those observations are made at analysis time accurately represent the execution conditions that may occur during operation. Although this sounds overly difficult, it can be done in practice using the power of randomisation. Discussing how falls outside of the scope of this paper, and it is the subject of dedicated work that will soon be published.

The average performance of PTA-friendly processor architectures is marginally worse than that of conventional architectures, and pWCET estimates are typically marginally higher than average execution time (as a consequence of running on a processor that intrinsically reduces the width of response time jitter): pWCET estimates are therefore comparable to actual performance on conventional architectures and attain the strong performance guarantees required for safety critical systems.
MBPTA can be applied on legacy code and does not impose any further constraint on the application software other than those in place for current practice based on measurement-based and static timing analysis. In fact, MBPTA drastically reduces the amount of information needed for conventional timing analysis by, for instance, removing any dependence on the particular address where code and data are mapped in memory.

All in all, we have shown that MBPTA has great potential for enabling industrial users to determine tight and trustworthy upper-bounds to the worst-case execution time of safety-critical applications with attractively low cost and complexity in terms of detail system knowledge, infrastructure, analysis procedure.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7/2007-2013] under the PROARTIS Project (www.proartis-project.eu), grant agreement no 249100. This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant TIN2012-34557 and theHiPEAC Network of Excellence. Leonidas Kosmidis is funded in memory.

REFERENCES