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

Franck Wartel[ψ], Leonidas Kosmidis[†,*], Code Lo[⋆], Benoit Triquet[ψ], Eduardo Quiñones[†], Jaume Abella[†], Adriana Gogonel[⋆], Andrea Baldovin[ϕ], Enrico Mezzetti[ϕ], Liliana Cucu[⋆], Tullio Vardanega[ϕ], Francisco J. Cazorla[†,‡]

[ψ]Airbus, France
[†]Barcelona Supercomputing Center, Spain
[*]Universitat Politècnica de Catalunya, Spain
[⋆]INRIA, France
[ϕ]University 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 the problems that impair current worst-case execution time (WCET) analysis techniques. MBPTA can compute tight WCET bounds expressed as probabilistic exceedance functions, without needing much information on the hardware and software internals of the system. Classic WCET analysis needs detail information instead, which may be costly and difficult to provide, and turns its absence into 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 therefore interesting to see how PTA responds to them. This paper discusses the application of MBPTA to a real avionics system and presents the 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. Unmanned Aerial Vehicles pose similar if not greater demands. A similar phenomenon has been observed in other application domains, over different time spans. For all of those application domains, the strict real-time nature of a large cross-section of their programs turns that trend into 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 likely is 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]. With it, 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 the predicted 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 execution time distribution of the program and can be incrementally fed to the timing analysis of the system. Hence, MBPTA has low information cost while achieving sound results. MBPTA requires the hardware to have a probabilistically characterisable timing behaviour [4][6]: this is not the case with conventional processors but can arguably be achieved without exceeding their 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 assessing 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. In this paper we: 1) show that the requirements posed by PTA can be met for real avionics applications; 2) illustrate how probabilistic WCET (pWCET) estimates are produced; 3) discuss issues occurring in the timing analysis process; and 4) evaluate the quality of the obtained pWCET

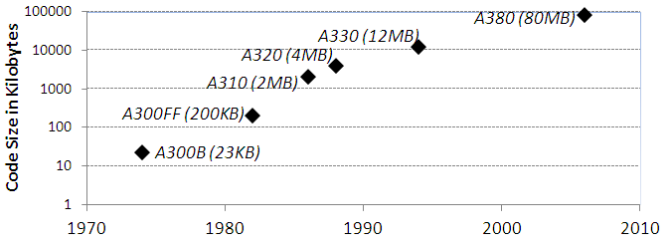


Fig. 1. Code size evolution for safety critical systems in Airbus aircraft.

estimate in relation to use in production. 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 ARINC 653 READ_SAMPLING_MESSAGE service, which – owing to the system architecture – resides in the application space and therefore contributes to end-to-end application timing. 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 5 minutes between preparation, measurement run and processing of the observations. 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 VI 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 may undergo local optimisations. However, since most of the federated computers perform essentially the same functions (input acquisition, processing and output generation), a natural global 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. This shift in system integration is illustrated in Figure 2: functions A, B and C, which run on fully isolated systems in conventional avionics, are executed together in the IMA approach, where they share platform resources.

Although this paradigm shift to Integrated Systems comes from the IMA concept inbred to the avionics community, it is certainly applicable to other application domains. In the automotive sector, for example, software components can be

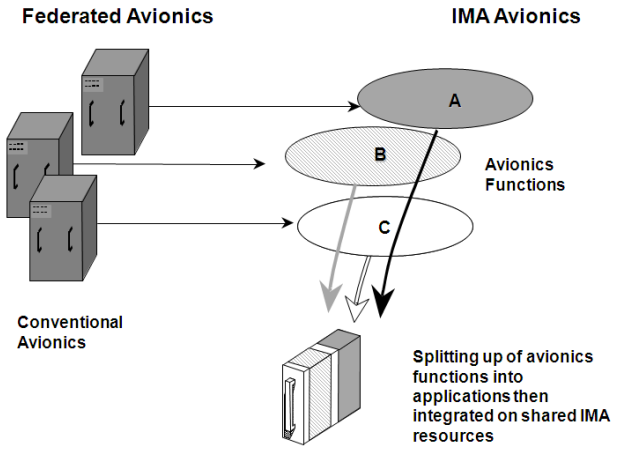


Fig. 2. Integrated Modular Avionics (IMA) concept.

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 were made on 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 rests on the two principles of 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 an offline-computed static schedule, where a partition is the scheduling unit. The **MA**jor **F**rame represents the principal periodic sequence of partition executions, to be complied with by the Operating System. The MAF is broken down into an integral number of **MI**nor **F**rames of equal period and duration. A *Partition Time Window* represents a time slot during which the Operating System exclusively schedules processes belonging to a given partition. Each partition is given at least one Partition Time Window in every MIF. Processes belonging to a given partition can only be scheduled during the Partition Time Window of that partition.
- *Spatial partitioning* allows partitions to execute without affecting one another spatially. It is ensured by prohibiting memory accesses (Read, Write or Execute) outside of the memory areas statically allocated to a partition.

B. Case Study

The avionics application considered in the case study executes with a period of P ms over N consecutive MIF. As illustrated in Figure 3, the application activity consists of five functional blocks, each of which is a high-level procedure, root of a finite and acyclic call graph which changes at the boundaries of every MIF:

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

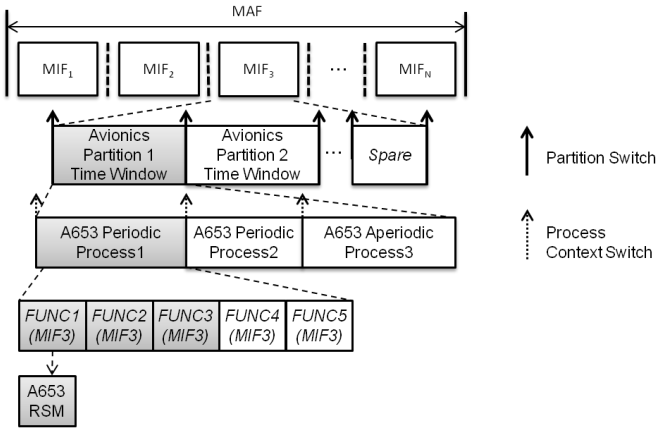


Fig. 3. Functional blocks and A653 services executed within a MIF and called by an A653 Periodic Process. Grey blocks denote the analysed programs.

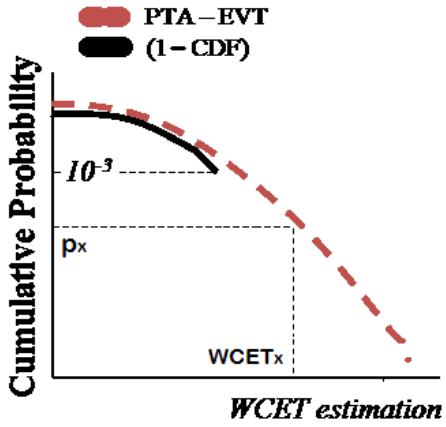


Fig. 4. Synthetic ICDF and 1-ICDF example

- Functional blocks *FUNC2* de-serialise 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 applied to it within acceptable bounds of engineering effort and containment of pessimism. The most 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 cannot be sanely analysed by application-level state-of-the-art STA tools.

The results presented in this paper were obtained by running the application on a PTA friendly simulator composed of PowerPC-family MPC755 instruction set and pipeline emulator attached to a time-accurate cache simulator. The application was run on a top of an ARINC 653 compliant operating system that provided zero-disturbance constant-time services [11] (thereby also for READ_SAMPLING_MESSAGE), 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 and Partition Switch time, when operating system activity deferred during the execution of the application is finally performed.

III. MBPTA

PTA has recently emerged as a viable alternative to current 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 $1-CDF$) 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 $\frac{1}{R}$. 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 4 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}. 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 4 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 the EVT with MBPTA requires that the observed execution times can be described by random variables that are proved 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 either be statically upper-bounded, or be probabilistically characterised.

We make this assumption hold 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, to which we will return in Section IV –

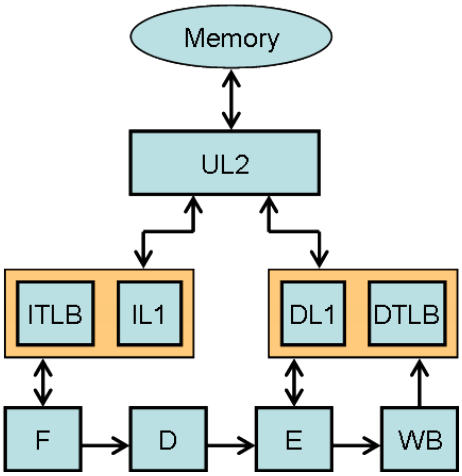


Fig. 5. Reference processor architecture

the only events that can cause (local and global) variations in execution times are truly random, as so by construction. In our case the instruction and data cache and TLB (translation lookaside buffer) respectively (see Section IV for further details). How to design "PTA-friendly" processors avoiding extensive modifications on 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 fulfills 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 5). 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 separated 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 a unified second level of 8 ways set-associative cache of 64 KB in size and 32-byte cache line each, 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, also 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-cycles latency. After the decode stage, memory operations access the data cache and TLB analogously to the fetch stage. Overall, the possible latencies of a non-memory instruction depend 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 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.

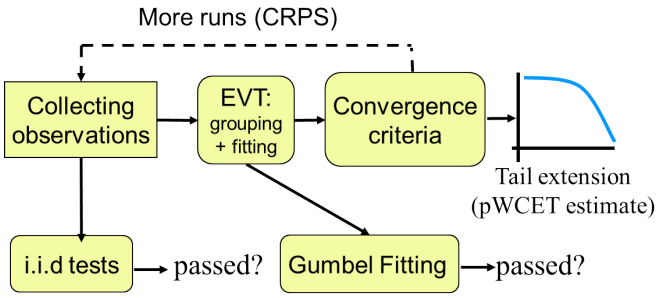


Fig. 6. Steps in the application of the MBPTA-EVT technique [6]

It is important to remark that the MBPTA approach cannot be applied to a standard deterministic architecture since the frequencies of the execution times observed during testing do not represent actual probabilities [6]. This is so because the events that affect execution time are not randomised. Hence, no probabilistic guarantees can be determined from the observed behaviour for the probability of occurrence of any given execution time in the future.

V. Results

We can now present the probabilistic timing analysis and average performance analysis for the three functions (FUNC1, FUNC2 and FUNC3) and the Partition Switch procedure (PS) selected as the case study. The measurements we report have been obtained on a cycle-accurate execution-driven 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 6 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 more observations were required to be able to compute the pWCET distribution (as we explain in the following steps), an additional N_{delta} observations are made and included in the data fed to the next steps. In our experiments we started collecting 100 execution time observations, with $N_{delta} = 50$. At every collection round, we checked whether the data are described by i.i.d random variables: the two-sample Kolmogorov-Smirnov (KS) [20] test evaluates the fulfilment of the identical distribution property, and the runs-test [21] the fulfilment 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 the set of observation runs sufficient to

TABLE I. P-VALUE FOR THE IDENTICAL DISTRIBUTION (CONSIDERING TWO SAMPLES OF $m = 50$ AND $m = 100$ ELEMENTS) AND INDEPENDENCE TEST.

	Identical Distr.		Indep	Passed?
	m=50	m=100	p	
FUNC1	0.77	0.89	0.85	Ok
FUNC2	0.69	0.90	0.08	Ok
FUNC3	0.52	0.68	0.68	Ok
PS	0.11	0.89	0.07	Ok

TABLE II. ET TEST RESULTS. *Inf Conf. Interval* AND *Sup Conf. Interval* STAND FOR INFERIOR AND SUPERIOR CONFIDENCE INTERVAL BOUNDS RESPECTIVELY.

	Estimated q	Inf Conf. Interval	Sup Conf. Interval	passed?
FUNC1	2732099.95	2731953.71	2732199.27	Ok
FUNC2	1290199.02	1290198.36	1290441.20	Ok
FUNC3	1414051.63	1410751.90	1414588.60	Ok
PS	1110203.18	1109958.36	1110642.24	Ok

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.

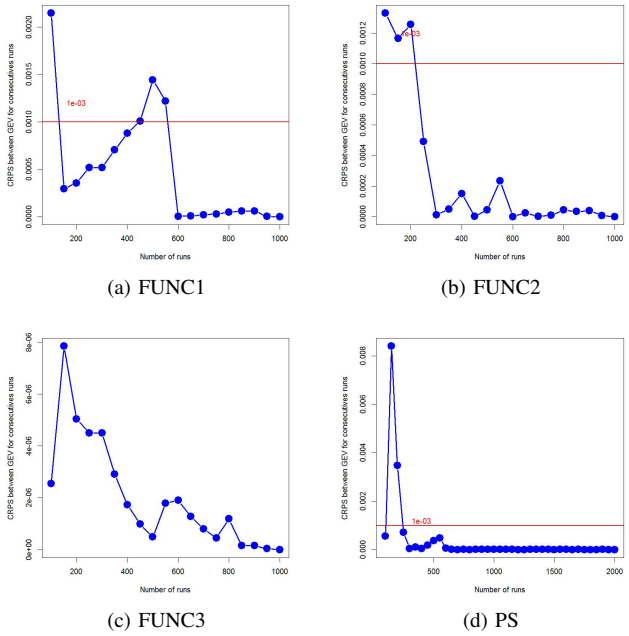
2) Grouping: From the data collected in the previous step we pick the high-watermark values. We do so as we use EVT to upper-bound the probability of occurrence of the highest execution times and not the average observed execution times. In this work we used the Block Maxima method to convert the set 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.

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 (ξ), scale (σ) and location (μ) parameters as follows:

$$F_\xi(x) = \begin{cases} e^{-(1+\xi\frac{x-\mu}{\sigma})^{\frac{1}{\xi}}} & \xi \neq 0 \\ e^{-e^{-\frac{x-\mu}{\sigma}}} & \xi = 0 \end{cases}$$

EVT requires that two hypotheses are verified: (1) the n random variables are independent and identically distributed (which we check in the *Collecting* step) and (2) the maximum of the n random variables converges to one of the three possible EVT distributions: Gumbel, Frechet or Weibull. The Gumbel distribution, with shape parameter $\xi = 0$, has been proven to fit well the problem of WCET estimation [6][5]. In our application of EVT we use the exponential tail (ET) test [22] to validate that the distribution fits a Gumbel distribution, as shown in table II, and to determine the σ and the μ that characterise the specific Gumbel distribution.

4) Convergence: Finally we use a method that determines when we have collected enough samples (minimum number of runs, MNR) such that the EVT results are probabilistically sound. 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. We use the continuous rank probability score (CRPS) metric, defined as



(a) FUNC1 (b) FUNC2

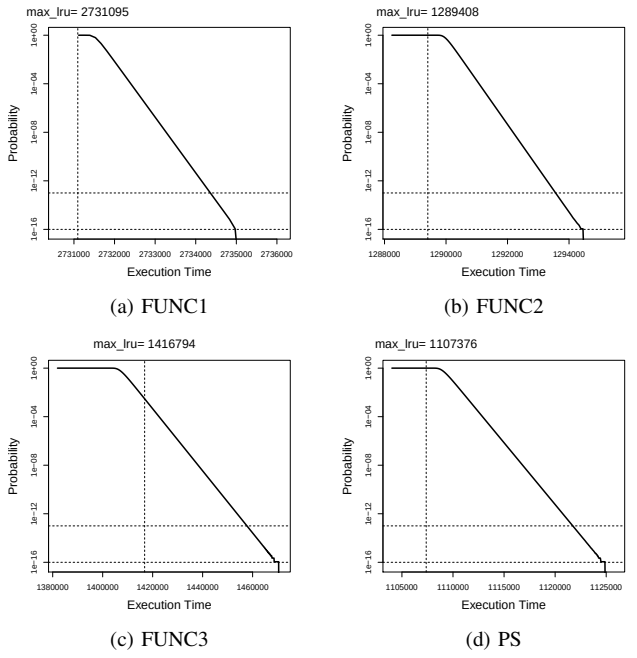
(c) FUNC3 (d) PS

Fig. 7. CRPS variation as we increase number of collected execution times. $N_{delta} = 50$ and threshold $= 0.001$

$CRPS = \sum_{i=0}^{+\infty}[f_{\mathcal{X}}(i) - f_{\mathcal{Y}}(i)]^2$ [21], to compare those two distributions, being $f_{\mathcal{X}}(i)$ and $f_{\mathcal{Y}}(i)$ the functions from current and 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 the more precise the distribution, albeit at the cost of needing more observations and EVT projections. In our case we considered a difference threshold of 0.001 as shown in Figure 7. 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 for us) can be used to compute the pWCET estimate associated to any exceedance probability threshold. Figure 8 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 actually traversed in the measurement runs[1]. The path coverage we obtained for FUNC3 was deemed sufficient by the scrutiny of

[1]Note that, as opposed to MBTA methods on top of deterministic architectures, MBTPA on top of PTA-friendly architectures has *no dependence* on the particular mapping of data and instructions in memory.



(a) FUNC1 (b) FUNC2

(c) FUNC3 (d) PS

Fig. 8. pWCET estimates (in processor cycles) for the programs under study. Horizontal lines stand for particular exceedance probability thresholds. Vertical lines stand for execution maximum observed time on deterministic LRU cache setup.

our industrial experts.

We saw earlier in section II that the use of state-of-the-art STA methods were frustrated by technical difficulties and poor results. The only other static option that could be tried on a processor with random placement and replacement caches was probabilistic STA, but MBPTA was preferred for its much lower cost of application. We thus had no STA data to compare against.

As MBPTA can only be meaningfully applied to programs running on a PTA-friendly architecture as described earlier in this paper, the case-study application has been run on a processor simulator that supports the required features. At that point, the only practical solution for a cycle-true accurate comparison was to set the simulator cache to operate with modulo placement and least recently used (LRU) replacement, and to use the common industrial practice of taking the highest measurement for each program of interest and adding an engineering margin to it as determined by in-house knowledge on the system. The bounds we obtained by running the same experiments on the simulator with caches configured in deterministic LRU mode are shown as dashed vertical lines in Figure 8.

pWCET estimates are significantly below a traditional *LRU+margin%* WCET estimate, where the additional margin determined by engineering judgment is usually chosen around 20%. As shown in Figure 8, pWCET estimates are slightly above LRU execution times (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 $[10^{-14}, 10^{-15}]$: $10^{-9}\frac{failures}{hour} / \left(3600 \times 10^2 \frac{task\ activations}{hour}\right)$. Therefore, an exceedance probability threshold of 10^{-15} suffices to achieve the highest integrity level.

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.

	10^{-13}	10^{-16}
FUNC1	0.03%	0.03%
FUNC2	0.06%	0.07%
FUNC3	0.81%	0.87%
PS	0.26%	0.28%

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

Program	FUNC1	FUNC2	FUNC3	PS
MNR	600	250	300	250

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.

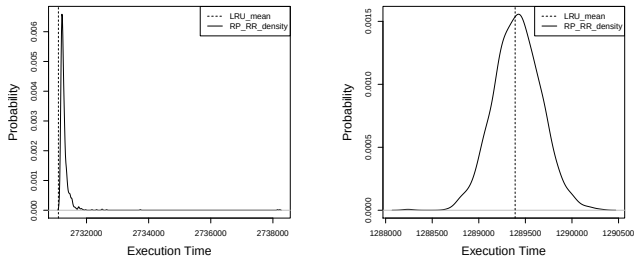
B. Cost of application

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 verifying the fulfilment of the i.i.d. property, to apply block maxima and to characterise the EVT distribution parameters (ξ, σ and μ). To do so, we have developed an R-script[2] which takes 2 seconds to derive an EVT distribution from a collection of $1,000$ execution times, 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 $N_{delta} = 50$. The monitored functions required in the order of a few hundreds of runs, as illustrated in Table IV.

Overall, the amount of time required to collect data and compute the pWCET distribution of each of the tasks considered in this paper is less than 5 minutes. Therefore, we conclude that the MBPTA technique reduces considerably the information costs incurred by other timing analysis methods.

[2]http://www.r-project.org/



(a) A653 RSM (b) FUNC1

(c) FUNC2 (d) FUNC3

Fig. 9. Comparison of the density functions of observed execution times (in processor cycles) when running with the time-randomised cache (continuous curve) and a deterministic cache (dashed vertical line).

TABLE V. COMPARISON OF RELATIVE DISTANCE BETWEEN MAX OBSERVED VALUE WIT LRU CACHE CONFIGURATION AND PWCET ESTIMATE WITH MBPTA

	LRU_MAX	pWCET	pWCET relative distance
FUNC1	2731095	2735109	0.14%
FUNC2	1289408	1294519	0.39%
FUNC3	1416794	1470490	3.78%
PS	1107376	1124934	1.58%

C. Analysis of Average Performance

Figure 9 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 VI 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-randmised cache is very low, actually below 1%. This suggests that the minor average performance loss caused by the time-randomised cache for the selected application is more than compensated by the low cost of applying probabilistic timing analysis.

Figure 8 also relates the average execution time obtained with the deterministic cache configuration against the pWCET 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 V the pWCET are slighly higher but very close to the maximum observed execution time with caches configured in a deterministic way.

TABLE VI. COMPARISON OF RELATIVE DISTANCE BETWEEN MEAN EXECUTION TIME VALUE WITH DETERMINISTIC CACHE CONFIGURATION AND TIME-RANDOMIZED ONE.

	LRU_Mean	RP_RR_Mean	Relative distance
FUNC1	2731095	2731293	0.007%
FUNC2	1289390	1289415	0.001%
FUNC3	1381245	1391076	0.712%
PS	1105708	1106801	0.099%

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 types of contributions. The first class concerns the schedulability analysis of probabilistic real-time systems and the second the proposition of probability distributions for the parameters.

With respect to the first class and 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].

With respect to the second class and 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 class 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 IMA-based applications produces tight pWCET estimates, thus increasing system utilisation as required for new-generation avionics systems. We have also shown that the cost of computing a pWCET bound with the MBPTA technique nears 5 minutes per program, 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 apply for other systems in the same or other domains:

1) 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.

2) 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.
3) 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.
4) 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.

REFERENCES

[1] G. Edelin, "Embedded systems at thales: the artemis challenges for an industrial group," in *Presentation at the ARTIST Summer School in Europe 2009*, 2009.

[2] E. Mezzetti and T. Vardanega, "On the Industrial Fitness of WCET Analysis," in *Proceedings of the 11th International Workshop on Worst-Case Execution Time Analysis (WCET)*, 2011.

[3] R. Kirner and P. Puschner, "Obstacles in Worst-Case execution time analysis," in *ISORC 2008: Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, 2008, pp. 333 – 339.

[4] F.J. Cazorla et al., "Proartis: Probabilistically analysable real-time systems," INRIA, Tech. Rep. 7869 (http://hal.inria.fr/hal-00663329), 2012.

[5] J. Hansen, S. Hissam, and G. A. Moreno, "Statistical-based wcet estimation and validation," in *the 9th International Workshop on Worst-Case Execution Time (WCET) Analysis*, 2009.

[6] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *ECRTS*, 2012.

[7] J. Poovey, *Characterization of the EEMBC Benchmark Suite*, North Carolina State University, 2007.

[8] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The Mälardalen WCET benchmarks – past, present and future," in *the International Workshop on Worst-case Execution-time Analysis*, 2010.

[9] ARINC, "Avionics Application Software Standard Interface: ARINC Specification 653P1-3. Aeronautical Radio," www.arinc.com.

[10] Esterel, "SCADE," www.esterel-technologies.com/products/scade-suite/.

[11] A. Baldovin, E. Mezzetti, and T. Vardanega, "A time-composable operating system," in *12th International Workshop on Worst-Case Execution Time Analysis (WCET)*, 2012.

[12] Edgar S and Burns A, "Statistical analysis of WCET for scheduling," in *the 22nd IEEE Real-Time Systems Symposium (RTSS01)*, 2001, pp. 215–225.

[13] L. Yue, I. Bate, T. Nolte, and L. Cucu-Grosjean, "A new way about using statistical analysis of worst-case execution times," *ACM SIGBED Review*, September 2011.

[14] S. Kotz and S. Nadarajah, *Extreme value distributions: theory and applications*. World Scientific, 2000.

[15] L. Kosmidis, J. Abella, E. Quinones, and F. Cazorla, "A cache design for probabilistic real-time systems," To appear at DATE 2013, Tech. Rep. UPC-DAC-RR-CAP-2012-18 (https://www.ac.upc.edu/app/research-reports/html/2012/24/CacheDesignForPRTS.pdf), 2012.

[16] L. Kosmidis, C. Curtsinger, E. Quinones, J. Abella, E. Berger, and F. Cazorla, "Probabilistic timing analysis on conventional cache designs," To appear at DATE 2013, Tech. Rep. UPC-DAC-RR-CAP-2012-20 (https://www.ac.upc.edu/app/research-reports/html/2012/27/main.pdf), 2012.

[17] I. Wenzel, R. Kirner, P. Puschner, and B. Rieder, "Principles of timing anomalies in superscalar processors," *Proceedings of the Fifth International Conference on Quality Software*, pp. 295–306, 2005.

[18] LiP6, "SoCLib," www.soclib.fr/trac/dev.

[19] J. Wetzel, E. Silha, C. May, B. Frey, J. Furukawa, and G. Frazier, *PowerPC User Instruction Set Architecture*, IBM Corporation, 2005.

[20] W. Feller, *An introduction to Probability Theory and Its Applications*. John Willer and Sons, 1996.

[21] J. Bradley, *Distribution-Free Statistical Tests*. Prentice-Hall, 1968.

[22] M. Garrido and J. Diebolt, "The ET test, a goodness-of-fit test for the distribution tail," in *Methodology, Practice and Inference, second international conference on mathematical methods in reliability*, 2000, pp. 427–430.

[23] , "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," *ARP4761*, 2001.

[24] J. Lehoczky, "Real-time queueing theory," in *the 10th IEEE Real-Time Systems Symposium (RTSS96)*, 1996, pp. 186–195.

[25] J. Díaz, D. Garcia, K. Kim, C. Lee, L. Bello, L. J.M., and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *the 23rd IEEE Real-Time Systems Symposium (RTSS02)*, 2002, pp. 289–300.

[26] L. Cucu and E. Tovar, "A framework for the response time analysis of fixed-priority tasks with stochastic inter-arrival times," *SIGBED Review*, vol. 3, no. 1, pp. 7–12, 2006.

[27] A. Thekkilakattil, R. Dobrin, and S. Punnekkat, "Probabilistic preemption control using frequency scaling for sporadic real-time tasks," in *SIES*, 2012, pp. 158–165.

[28] H. Zeng, M. D. Natale, P. Giusto, and A. L. Sangiovanni-Vincentelli, "Using statistical methods to compute the probability distribution of message response time in controller area network," *IEEE Trans. Industrial Informatics*, vol. 6, no. 4, pp. 678–691, 2010.

[29] G. A. Kaczynski, L. L. Bello, and T. Nolte, "Towards stochastic response-time of hierarchically scheduled real-time tasks," in *Proceedings of 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2006)*, 2006, pp. 453–456.

[30] L. Abeni, N. Manica, and L. Palopoli, "Efficient and robust probabilistic guarantees for real-time tasks," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1147–1156, 2012.

[31] M. Lombardi, M. Milano, and L. Benini, "Robust scheduling of task graphs under execution time uncertainty," *IEEE Trans. Computers*, vol. 62, no. 1, pp. 98–111, 2013.

[32] L. Santinelli, P. M. Yomsi, D. Maxim, and L. Cucu-Grosjean, "A component-based framework for modeling and analyzing probabilistic real-time systems," *16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'11)*, 2007.

[33] D. Maxim, O. Buffet, L. Santinelli, L. Cucu-Grosjean, and R. I. Davis, "Optimal priority assignment algorithms for probabilistic real-time systems," in *RTNS*, 2011, pp. 129–138.

[34] D. Maxim, M. Houston, L. Santinelli, G. Bernat, R. I. Davis, and L. Cucu-Grosjean, "Re-sampling for statistical timing analysis of real-time systems," in *RTNS*, 2012, pp. 111–120.

[35] K. S. Refaat and P.-E. Hladik, "Efficient stochastic analysis of real-time systems via random sampling," pp. 175–183, 2010.

[36] R. Perrone, R. Macedo, G. Lima, and V. Lima, "An approach for estimating execution time probability distributions of component-based real-time systems," *J. UCS*, vol. 15, no. 11, pp. 2142–2165, 2009.

[37] F. Dewan and N. Fisher, "Efficient admission control for enforcing arbitrary real-time demand-curve interfaces," in *RTSS*, 2012, pp. 127–136.

[38] M. Neukirchner, T. Michaels, P. Axer, S. Quinton, and R. Ernst, "Monitoring arbitrary activation patterns in real-time systems," in *RTSS*, 2012, pp. 293–302.