Extreme Value Theory for Estimating Task Execution Time Bounds: A Careful Look

George Lima
Depart. of Computer Science
Federal University of Bahia
Salvador, Bahia, Brazil
gmlima@ufba.br

Dário Dias, Edna Barros
Center of Informatics
Federal University of Pernambuco
Recife, Pernambuco, Brazil
dsd@cin.ufpe.br, ensb@cin.ufpe.br

***Abstract*—Extreme Value Theory (EVT) is a powerful statistical framework for estimating maximum values of random variables and has recently been applied for deriving probabilistic bounds on task execution times (pWCET). Task execution time data are collected from measurements and the maximum measured values are fit to an extreme value model. In this paper we provide a careful study on the applicability and effectiveness of EVT in this application field. The study is based on extensive experiments for which we have designed an embedded platform equipped with random cache of configurable sizes. Based on evidences of the experiments, we provide the following contributions: we give a new definition of pWCET that conforms with the fact that pWCET estimates depend on input data distribution used during analysis; we show that using the Generalized Extreme Value (GEV) distribution is necessary since the more restrictive modeling, based on the Gumbel distribution, may yield unsafe or over-estimated values of pWCET; we confirm that hardware randomization favors the applicability of EVT, although it does not ensure it since the distribution of maxima for execution time data are not guaranteed to be analyzable via EVT.**

I. Introduction

Motivation. Extreme value theory (EVT) is a statistical inference framework aiming at describing rare (extreme) stochastic behavior, which usually lies beyond what has been observed in the sampled data. Its application ranges from hydrology and finance to sports, *e.g.,* [1], [2], [3].

In the field of real-time systems, estimating extreme value (EV) models to derive probabilistic task worst-case execution time (pWCET) is particularly appealing. Hardware and software architectures have become increasingly complex so that deriving deterministic bounds on task execution times is likely to produce overly pessimistic estimates or can be even infeasible in some circumstances. This has motivated intensive research in the area, where EVT is usually referred to as a *measurement-based* technique. In a nutshell, it consists of: (i) measuring how long a task takes to execute for a representative set of its inputs; (ii) selecting from these measurements a sample of maxima; (iii) deriving the EV model that best fit to the selected maxima; and (iv) once the derived model is validated, obtaining pWCET estimates.

Despite recent advances in applying EVT for deriving pWCET, some relevant issues have not been satisfactorily addressed. First, samples obtained from steps (i) and (ii) are highly dependent on how input data sets are generated and this may lead to completely different pWCET estimations for the same task. Second, it is known that EVT is only applicable if the sampled distribution obtained in step (ii) can be described by an EV distribution [4], [5], [6]. Such a requirement may not be necessarily fulfilled by task execution time distributions. Third, if it is, this distribution of maxima can be of three types [7]. However, most previous derivations of pWCET have been restricted to only one EV distribution type, namely the Gumbel distribution.

Contribution. In this paper we address the aforementioned issues by assessing the applicability and effectiveness of EVT for pWCET estimations. Based on evidences obtained from extensive experiments, we report important observations on using EVT in the field of timing analysis.

We first provide a new definition of pWCET to conform with the fact that the obtained execution time distributions depend on input data used during the measurements. Then we model the sample of task execution time maxima using the Generalized Extreme Value (GEV) distribution, which unifies the three possible EV distributions, *i.e.,* Weibull, Gumbel and Fréchet.[1] We then confirm that EVT is not applicable for deriving pWCET in general. In some cases, the distributions of maxima obtained from experiments do not converge to any EV distribution, making pWCET estimation via EVT not possible. We also show that any of the three EV distributions may arise when modeling pWCET and that restricting the model to only a single EV distribution may lead to unsafe or pessimistic estimations.

Due to the benefits that probabilistic timing analysis potentially brings about, several researchers have advocated the use of randomized hardware/software resources so as to improve the applicability of this kind of approach *e.g.,* [9], [10], [11], [12]. More specifically, recent debate has been established on the use of random caches [13], [14]. We contribute to this debate by applying EVT on two types of hardware platforms, which we name *time-predictable* and *time-unpredictable*. By time-predictability we mean that the execution time of a

[1]The GEV model is not the only characterization of extreme events. Other alternatives include the threshold excess and the point process models. From a theoretical perspective, all these characterizations should give equivalent estimates for rare events. Interested readers are referred to textbooks in the area for more details [8].

sequence of instructions is always the same whenever they execute over the same data. We report that EVT can be successfully applied in time-predictable platforms and so hardware randomization is not strictly necessary for ensuring analyzability. Further, our experiments show that in general time-unpredictability is not sufficient to ensure the applicability of EVT.

When EV models are found applicable, they indeed provide a flexible and powerful tool. We illustrate this aspect by determining pWCET estimates as a function of cache sizes. As cache partitioning has been pointed out as a way of controlling the inter-task interferences and improving timing analyzability, such information can be useful in determining ideal cache partition sizes to be assigned to tasks, as it is the case for some schemes *e.g.,* [15], [16], [17].

To carry out our experiments, we have implemented an embedded platform that consists of a 32-bit RISC V processor, 16MB of main memory and fully-associative instruction and data caches. Caches can be set off or configured with different sizes. Special-designed measurement instructions were also implemented so that time measurements are very accurate. Applications, coded in C/C++, are embedded into the platform, which is then compiled into System C and executed by a hardware simulator [18].

Paper structure. Section II discusses concepts related to pWCET, gives necessary background on EVT and briefly summarizes related work. The experiment platform is described in Section III. The experiment results are reported in Sections IV and V, which take time-predictable and time-unpredictable platforms into consideration, respectively. Our conclusions are drawn in Section VI.

II. Background and related work

A. Probabilistic worst-case execution time

For the purpose of this work, we informally define an *execution path* of a task as a finite sequence of (low level) instructions to be executed on a given platform in response to input stimuli. A task consists of at least one execution path and different stimuli may give rise to distinct execution paths for the same task. If the platform on which the task runs is *time-predictable*, the same execution path takes the same amount of time to complete its execution whenever it runs. Otherwise, we say that the platform is time-unpredictable. Possible sources of time-unpredictability are mechanisms to boost performance or to provide concurrence control. Both types of platforms are considered here and we are concerned with random cache effects as the only source of time-unpredictability.

A tight value for the worst-case execution time of a task can be defined as the time to execute its longest execution path. As determining the longest path is usually not feasible, it is acceptable that the worst-case execution time (WCET) of a task represents an upper bound on the time the task takes to execute any of its execution paths on a given platform. This means that if $t > 0$ is a WCET of a task τ on a given platform P and T is a random variable representing how long an arbitrary execution path of τ takes to run on P, then the probability that $T \leq t$ equals 1, *i.e.,* $\Pr(T \leq t) = 1$. A natural extension of this concept to incorporate uncertainty on the WCET would be:

Definition 1 (Strong pWCET): If tuple (t, p) is a probabilistic worst-case execution time of a task τ on a given platform P, with $t > 0$ and $0 < p \leq 1$, and T is a random variable representing the execution time of τ when run on P, then $\Pr(T \leq t) \geq 1 - p$.

The concept of pWCET, as stated in Definition 1 or in other alternative definitions previously given by other authors (see [19] for instance), assigns a degree of uncertainty (expressed by a probability value) to the knowledge on WCET. Unfortunately, this kind of definition is of little practical use when it comes to measurement-based estimations of pWCET.

Indeed, when inferences on pWCET are based on measured execution time values, such observations depend on which execution paths are triggered during the measurements, which in turn depend on the input stimuli applied to the analyzed task. Simply, there may be not a single distribution to describe a task execution time behavior. The following definition takes this into account:

Definition 2 (Weak pWCET): Consider a task τ to be executed on a platform P. Let $D(\tau)$ be a probability distribution describing how frequent input stimuli for τ takes place. If (t, p) is a probabilistic worst-case execution time of a task τ under the assumption that $D(\tau)$ holds and T is a random variable representing the execution time of τ when run on P, then $\Pr(T \leq t | D(\tau)) \geq 1 - p$.

According to Definition 2, one must define $D(\tau)$ in order to estimate pWCET for a task τ. This is certainly a problem itself. A usual recommendation is that $D(\tau)$ be a uniform distribution on the data input space of τ [20] so as to give the same chance for every possible input choice. This will be the approach used in this paper. As will be discussed later, although convenient, this approach may not suit all application domains.

Next, we explain the derivation of EVT-based pWCET estimates, which should be interpreted as in Definition 2.

B. The generalized extreme value distribution

1) Definition: Let $X_1, X_2, \ldots, X_n$ be a sequence of independent random variables distributed according to a common (usually unknown) function F and define $M_n = \max_{i=1}^{n}(X_i)$. From the results by Fisher, Tippett [7] and Gnedenko [21], if there are sequences of normalizing constants $\{a_n > 0\}$, $\{b_n\}$ such that $(M_n - b_n)/a_n$ converge in distribution (as $n \to \infty$) to a non-degenerate distribution of M_n, namely $G(z) = \Pr\{M_n \leq z\}$, $G(z)$ can be only of three types; which can be expressed as [22], [23]:

$$G(z) = \exp\left\{-\exp\left[1 + \xi\left(\frac{z-\mu}{\sigma}\right)\right]^{-1/\xi}\right\} \qquad (1)$$

defined on $\{z : 1 + \xi(z - \mu)/\sigma > 0\}$; with parameters $\mu \in \mathbb{R}$, $\sigma > 0$ and $\xi \in \mathbb{R}$ being respectively the location, scale, and shape of G. Equation (1) is known as the Generalized Extreme Value (GEV) distribution. It unifies the three possible

EV distributions: (reversed) Weibull (if $\xi < 0$, short bounded tail); Gumbel (if $\xi = 0$, light unbounded tail)[2]; and Fréchet (if $\xi > 0$, heavy unbounded tail).

The GEV model is particularly convenient. It removes the need of assuming a specific EV model to represent the data, which would require special designed techniques capable of expressing the uncertainties associated with the assumption made. With the GEV distribution, estimated model uncertainties are expressed in confidence intervals for the shape parameter. More specifically, confidence intervals for ξ gives the lack of certainty as to which EV distribution better models the data in a straightforward manner.

2) Sampling the maxima: In order to estimate the model parameters for (1) μ, σ, and ξ, one must select a sample of maxima obtained from a set of task execution time data measurements. The *block maxima* approach is a natural way of doing so: observed data are partitioned into a set of disjunct m data blocks of equal lengths and the maximum of each block is selected [24]. This sample of m maximum values are then used for model fitting.

3) Inference for the GEV distribution: Since we do not assume any restriction on the shape parameter ξ of (1), we apply the L-moments method [25] to estimate the three parameters of the GEV distribution. This is particularly convenient since the usually flexible maximum likelihood estimation method cannot be applied when $\xi < -1$ or may exhibit asymptotic convergence problems when $\xi \leq -0.5$ [26]. L-moments are based on linear combinations of order statistics, which can be applied independently of the value of ξ. Standard deviation and confidence intervals must be inferred separately. One approach to doing so is via Parametric Bootstrap methods [27], according to which inferences can be carried out based on resampling the data sample with replacement. We have used R [28] and its package exTremes [29], which provides a large set of necessary routines to carry out both EV inference procedures and goodness-of-fit checking.

4) Goodness-of-fit and extreme value condition: Poor fit in GEV models can be found when: the i.i.d. assumption is violated; if difficulties in obtaining asymptotic convergence properties take place; or when the data simply cannot be analyzed via EVT.

As for the i.i.d assumption, previous work have addressed the issue – see for instance [30], [12], [31], [32], [19]. If all machine state variables (*e.g.,* cache, registers *etc.*) are cleared up before each time the analyzed task is set to run, which is the case in our experiments, the assumption that the observed execution times are i.i.d. is highly plausible. Further, the task input data must be chosen according to some i.i.d. distribution so as to avoid generating non-i.i.d. execution time data, which would make the model fitting harder.

With respect to asymptotic properties, the sample and block sizes may be required to be large in some scenarios in order to observe good model fitting. Since in general there is no strict constraint on the number of times the execution of a task is measured, sample and block sizes can be as large as necessary (subject to cost and time constraints to run the measurement experiments).

Several statistical tests to assess the goodness-of-fit of estimated GEV models are available – see [33] for some references and discussion on this issue. In this paper we have followed recommendations by Coles [8], which have been used in several EV application areas: goodness-of-fit is carried out by jointly analyzing several complementary plots comparing the estimated models with empirical data. Due to its expressiveness, Quantile-Quantile plots are shown here: estimated model quantile is plotted against the empirical quantile; a good fit is shown when the empirical and estimated distributions agree, which is indicated when sampled data appears on or close to the diagonal line, as can be seen in Figs. 7(c), 7(d), 8(a) and 8(b). Figs. 4, 7(a) and 7(b) exemplify models that should be rejected. As a rule of thumb, one must be conservative when evaluating estimated models for pWCET, rejecting the model if no strong evidence of its goodness-of-fit shows up.

In some cases, however, it is not possible to estimate a reliable GEV model. Such cases can be identified by carrying out extreme value condition tests. In particular, we have applied the test by Dietrich et al. [5], implemented and made publicly available by Hüsler and Li [6], [34], to check whether the execution time data can be analyzed via EVT. The test is based on comparing the values of two quantile functions: one constructed based on the first k largest sampled data at hand; and the other obtained by asymptotic approximation. The hypothesis that the distribution of maxima for the sampled data converges to the GEV distribution is rejected if the former function is consistently greater than the latter for some range of k. In our work we use this test for checking possible reasons for model fitting difficulties.

The above problems are also found in other EVT application areas. For example, assuming i.i.d. data for finance, insurance or weather-related events may not be reasonable; occurrences of these kinds of events are usually co-related and may not be described by a single underline distribution. Techniques to deal with non-i.i.d. side-effects can be applied, *e.g.,* de-clustering. Within the GEV model, the block maxima sampling approach provides a great degree of robustness. As mentioned by Coles [8], even in the case of non-i.i.d. data, samples obtained by this approach may be described by a GEV distribution provided that block sizes are large enough; values selected from different blocks are unlikely to be related to each other.

Based on the above observations, we assume, as in other research fields where EVT have been applied, that the random variables defined as the maxima of m blocks, namely $Z_1, Z_2, \ldots, Z_m$, are independent and identically distributed, although the underline distribution F from which Z_i was sampled may be not. If this assumption does not hold for some analyzed task, poor estimation takes place, which can be checked by suitable goodness-of-fit tests.

5) GEV quantiles and pWCET estimates: Denote $q(p)$ as the $(1-p)$-quantile of (1), $0 < p < 1$. That is, $q(p)$ is the value

[2] $\xi = 0$ should be interpreted as $\xi \to 0$

such that $G(q(p)) = 1 - p$. Inverting (1) yields:

$$q(p) = \begin{cases} \mu - \frac{\sigma}{\xi}\left\{1 - [-\log(1-p)]^{-\xi}\right\}, & \xi \neq 0 \\ \mu - \sigma \log\left[-\log(1-p)\right], & \xi = 0 \end{cases} \quad (2)$$

Values of $q(p)$ are related to the model (1), which is estimated based on an empirical distribution constructed by applying the block maxima approach on raw data. If each block contains b measured execution time values, $q(p)$, inferred from (2), is expected to be exceeded in one b-size block out of $(1/p)$ b-size blocks, on average.[3] In the case that b measured execution time data X_i in each block are i.i.d. following a probability distribution F, it follows that $F^b(q(p)) = \Pr(X_1 \leq q(p), \ldots, X_b \leq q(p)) = \Pr(\max(X_i) \leq q(p))$. This implies that the probability that a task's execution time does not exceed $q(p)$ can be estimated as

$$F(q(p)) = \Pr(\max(X_i) \leq q(p))^{1/b} = (1-p)^{1/b}$$

and so pWCET estimates can be derived straightforwardly as $(q(p), p')$ with $p' = 1 - (1-p)^{1/b} < p$. For $b = 50$ and $p = 10^{-2}$ or $p = 10^{-4}$, typical values used in our experiments, $p' = 2 \times 10^{-4}$ or $p' = 2 \times 10^{-6}$, respectively. In our experiments we did not find evidences to refute the independence of measurements X_i. Despite this, for the sake of generality, we assume that only the distribution of maxima is i.i.d. (not the underline distribution F). Thus, we conservatively use p as an upper bound for p' and denote $C(p) = q(p)$ for the corresponding conservative pWCET estimate $(q(p), p)$.

C. Related work

Static probabilistic time analysis (*e.g.,* [35]) or non-parametric measurement-based techniques (*e.g.,* [36], [37], [38]) are not covered in the brief summary we present here. The reader may also find relevant information in other sources [39].

Our focus is on EVT-based pWCET estimation.

pWCET estimation via EVT has started with the work by Edgar and Burns [40], who advocated the use of the Gumbel distribution in the analysis. Instead of modeling samples of maxima, they modeled raw data. This fact was later observed [41], when the block maxima approach was first used in the context of pWCET. Latter on the applicability of EVT have been questioned [30]. The raised issues have been addressed by other researchers, who have brought about relevant contributions to the field: the concept of independence in the context of pWCET has been highlighted [19]; successful use of EVT for pWCET estimation *e.g.,* [20], [42], [32] and for determining probabilistic bonds task response time [43], [44] has been reported; solutions to issues related to sampling rare events [12] and to execution path coverage [45] have been given. More recently, hardware randomization has been suggested so as to improve system analyzability via probabilistic techiniques in general, and via EVT in particular, *e.g.,* [9], [10], [14], [11], [12], [46].

Recently, the restriction of using only the Gumbel distribution when applying EVT has been removed. In this context, pWCET estimation in the presence of dependent data have been addressed [47]; approaches for checking whether the measured data are in line with the standard EVT assumptions, such as independently and identically distributed data, have been described [48]; pWCET estimation for a GPU architecture has been presented [49]; and a study on selecting modeling parameters such as the size of block maxima for the GEV model has been reported [50]. In this paper we follow these trends, applying the GEV model for carrying out our study. Indeed, choosing a specific EV model beforehand is usually not recommended [8]: uncertainties to which EV distribution to use is already incorporated in the estimated confidence interval for the shape parameter. We aim at bringing complementary information on aspects not reported in previous work and are concerned primarily with identifying corner EVT applicability scenarios. To do this, an experimentation environment was set up. In this environment, we circumvent possible difficulties due to non-i.i.d data and avoid possible interferences due to measurements. The only sources of uncertainties in the measurements are due to the input data or random-based cache. As will be seen, we found in our experiments models belonging to domain of the three EV distributions (*i.e.,* Weilbull, Gumbel, and Fréchet). Moreover, we have identified scenarios that cannot be modeled by *any* EV distribution even when employing random caches, a fact up to now not reported.

III. Experimentation environment

To run the experiments, measuring task execution times in an accurate and controlled manner, we have designed an embedded hardware platform equipped with a 32-bit RISC-V processor with its integer instruction set [51], 16MB of main memory, and instruction and data caches. The platform can be instantiated in two different configurations, the time-predictable configuration, which does not include cache, and the time-unpredictable configuration, which adds both instruction and data caches to the platform. Cache sizes are configurable and two cache arrangements are possible: either instructions and data occupy the same cache space or they are mapped onto distinct spaces. For any cache arrangement, caches are fully-associative and implement the random-replacement approach: upon a cache miss event, a cache block is randomly chosen to store the new block.

The hardware platform was specified in ArchC [18], an open-source SystemC-based architecture description language. ArchC allows for the ISA processor specification and generates a SystemC model of the processor. This model has been integrated in a SystemC virtual platform and the execution of an embedded application is simulated instruction by instruction. This simulation capability was used during our experiments. Each task under analysis is programmed in C (or C++) and cross-compiled into the target-application code, which is then

[3] In EVT terminology, $q(p)$ is called *return level* and $1/p$ is the corresponding *return period*. EVT is usually applied in time series with block size b containing measurements collected during periods of time of one year. In this case, return level $q(p)$ is expected to be exceeded on average in one year out of $(1/p)$ years.

embedded into the platform hardware. The simulation code is generated by a native ArchC compile and is executed by the simulator.

The sizes and arrangements of caches were configured at design time and the corresponding embedded systems were then instantiated and run independently by the RISC V virtual platform. For notation convenience, the platform cache size is specified in terms of the number of cache entries considered, namely κ. As in our 32-bit architecture instruction or data item has 4 bytes each, the cache size of our embedded platform occupies $\kappa \times 8$ bytes. Instruction and data cache spaces were configured to have the same size. As will be seen, typical values for κ are small due to both the simplicity of our embedded architecture (no OS, no interruption handlers *etc.*) and the fact that the analyzed algorithms exhibit high memory access locality.

Time measurements are carried out by the embedded platform itself for which we implemented a processor cycle counter at the processor instruction level. Each time an instruction is executed, the instruction itself adds the corresponding amount of execution time to the processor cycle counter.

In RISC V architectures, instructions operate over processor registers only except for load and store instructions, which access memory. Instruction execution time can be taken as a single time constant independently of the instruction considered since RISC V instructions are simple and have the same size. This allows us to consider in our experiment set-up that execution time of an execution path is determined by the time spent to load instruction or data from cache (upon a cache hit event) or from main memory (upon a cache miss event or in the time-predictable configuration). We have specified that the former takes 1 processor cycle whereas the latter takes 10 times as much. Hence, the internal overhead for managing the counter is not accounted for (recall that the embedded code is being simulated).

It should be stressed that these execution time values (1 and 10 processor cycles) are not the actual hardware time measures since the co-designed system is simulated. The chosen values are not a restriction of our experiment platform either. These time parameters were found representative for the embedded application domain and were chosen mainly to verify possible corner conditions for applicability of EVT since the resulting path execution time measurements tend to produce discrete distributions.

Fig. 1 illustrates how the source code (representing a task) is instrumented for measurements. The code for which execution time is to be measured is in between two macros, denoted as `wrt(0,`i`)` in the figure, $i = 1, 2$. Each of these macros is translated by the ArcC cross-compiler to two assembly instructions, one to load the literal value i from memory and the other to store it at memory address 0. In our implementation, an extra piece of code introduced in the store instruction checks whether value 1 or 2 are being stored at address 0. If this is the case, the internal processor cycle counter is activated or deactivated, accordingly. This measurement approach in very accurate: (i) time measurement is started/stopped only after the complete execution of the corresponding store instruction; (ii) this causes a measurement error of only two assembly instructions, which corresponds to the execution of the load/store instructions related to the stop measurement macro; (iii) only two instructions, related to the start macro, may affect the cache state, which can interfere in the cache miss/hit ratio and this is considered negligible comparing with the typical execution paths containing hundreds or thousands of instructions; (iv) the embedded code in charge of the measurements do not interfere in the value of the processor cycle counter.

```
1: int main() {
2:  for (int i = 0; i < MAXRUN; i++) {
3:     // input data generation goes here
4:     wrt(0,1) // starts measurement
5:        // task code goes here
6:     wrt(0,2) // stops measurement
7:   }
8: }
```

Fig. 1. Typical instrumentation for measuring task execution time.

All analyzed pieces of code were instrumented as depicted in Fig. 1. The reported results are for: `MAXRUN` $= 10^4$, *i.e.,* execution time of a task was measured $10,000$ times; and block maxima size equal to 50. These values were chosen so as to ensure good convergence conditions, which make it possible to better evaluate those scenarios for which negative results (*i.e.,* bad fit or non-convergence to EV distributions) were obtained. In general, smaller sizes for sample and block suffice. The values for exceedance probability p were 10^{-2} or 10^{-4}. Typically, industry may require much lower values. However, as will be seen, these values of p suffice for good pWCET estimates w.r.t. the algorithms we analyze.

IV. pWCET on a time-predictable platform

By analyzing some simple algorithms we show in this section that: applying EVT may not be possible; despite the deterministic behavior of the execution platform, accurate estimates of pWCET can be obtained; restricting the modeling to only one member of the GEV distribution family may lead to poor-quality estimations; and using uniformly distributed input data during analysis may not suit all application domains. Two of the algorithms were taken from the Malärdalen Benchmark [52] while the others were specially constructed to demonstrate the validity of our arguments. The algorithms from the benchmark, which were originally configured to exhibit a single-path behavior (*i.e.,* a single entry is given as input data), were slightly modified to run accepting randomly generated input data in line with Fig. 1.

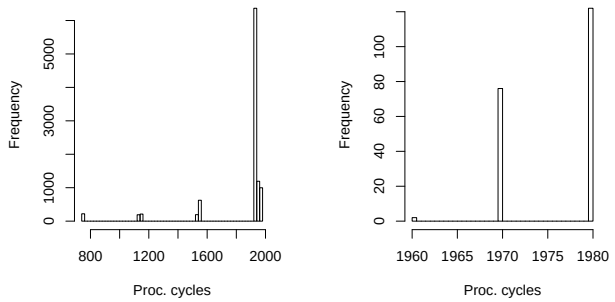
A. Cases for which EVT cannot be applied

Examples 1 and 2 are from the Malärdalen Benchmark:

Example 1 (Binary search): A 15-size vector is statically loaded with tuples in the form (k, v), where the keys $k = 1, 5, 6, 7, \ldots, 18$. Its execution time was measured for randomly

generated integer keys uniformly distributed in $[0, 50]$. As the algorithm has logarithm complexity and the considered key space tends to generate keys not in the vector, the observed execution times tend to be either the actual WCET or very close to it. The WCET was determined to be 1980 processor cycles, taking place for $k = 0$, for instance.

Fig. 2 shows the distributions of both raw and maximum data for Ex. 1. Both distributions present a high level of discreteness. As the algorithm takes a few steps during its execution, not many execution time values are generated in our experiment platform: the empirical distribution of maxima contains only three values, 1960, 1970, 1980 processor cycles, with frequency 2, 76, and 122, respectively. As expected, data are concentrated near the WCET.



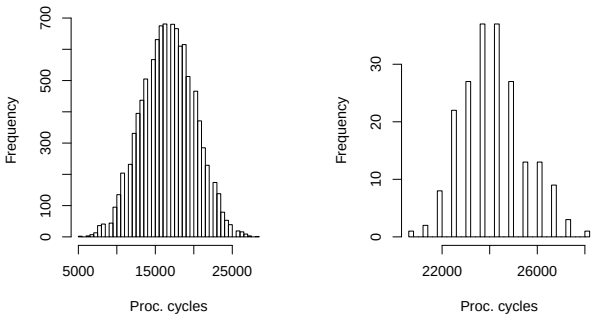
(a) Distribution of raw data (b) Distribution of maxima

Fig. 2. Distributions of data collected for Ex. 1. Minimum, maximum and mean values are respectively 740, 1980 and 1854.65 processor cycles for the raw data. Minimum and mean values for the distribution of maxima are 1960 and 1964.35 processor cycles, respectively.

Example 2 (Insertion sort): A 10-size integer vector is subject to the classical insertion sort algorithm, which consists of two nested loops. For each run of the algorithm, a vector whose elements were randomly chosen in interval $[0; 1000]$ was taken as input. The WCET was determined to be 30450 processor cycles, which represents the time the algorithm takes to sort a reversed-ordered vector. This worst-case scenario was not observed during the measurements.

The timing behavior of the insertion sort algorithm is expressed in Fig. 3. The maximum measured execution time was 28010 processor cycles. Interestingly, the histogram of maximum values displays a discrete distribution although the raw data present a reasonably smooth distribution. This is due to the nature of the algorithm and the execution costs of our experiment platform (10 cycles per operation). For example, a vector of size 3 would give rise to $3! = 6$ possible input configurations; if we assume that vector element swapping costs c cycles each, the execution time distribution w.r.t. the element swapping for this reduced vector would be $0, c, 2c, 3c$, with probabilities $1/6$, $2/6$, $2/6$, and $1/6$, respectively. As there are 10! input configurations related to Ex. 2, the distribution for raw data appears to be smooth as shown in the figure. The distribution of maxima, however, does not exhibit such as smoothness since it contains much fewer values.

When trying to estimate GEV models for these two examples, one is expected to find poor fittings, as shown in the Quantile-Quantile plots of Fig. 4. It is evident that the estimated models are not acceptable as valid. Carrying out the extreme value conditions tests [6] indicates that both data sets do not converge to EV distributions and so forcing an EV model for these cases is not recommended.



(a) Distribution of raw data (b) Distribution of maxima

Fig. 3. Distributions of data collected for Ex. 2. Minimum, maximum and mean values for raw data are respectively 4830, 28010 and 16684.98 processor cycles. Minimum and mean values for the distribution of maxima are 20690 and 24163.95 processor cycles, respectively.

One may be tempted to accept these models based on the fact that in both cases reasonable estimates for pWCET would be given. For example, according to the GEV models (neglecting their bad model fittings and the convergence test failure), values for $C(10^{-2})$ would be estimated to be 1981.73 (Ex. 1) and 27538.76 (Ex. 2) processor cycles, which are close to their WCET. These reasonably good estimates are due to the fact that there were measured execution times close (or equal to) the known WCET. However, in practical situations where probabilistic methods are called for, no information on WCET is expected to be available nor WCET values should be believed to appear in the measurement data sets. We stress that accepting such estimations in similar situations must not be done in practice.

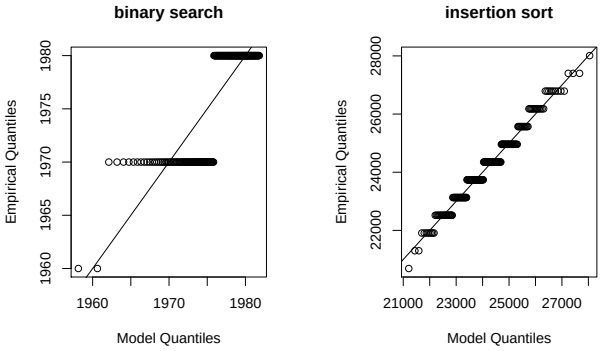


Fig. 4. QQ-plots for estimated models of Ex. 1 and 2.

B. The GEV model is needed for pWCET estimations

The negative results from the previous section must not be believed to be only due to time-predictability of the platform. This section shows that good GEV fittings are possible even when this kind of platform is considered. We also show here the risks associated with not considering the more general GEV model when estimating pWCET. We chose to show side-effects of restricting the modeling to the Gumbel distribution, as it has been done in most previous work. Similar effects can be found if one assumes Weilbull or Fréchet distributions instead. A very simple algorithm applied in two different contexts suffice to our purposes:

Example 3 (Counters): Consider a counter function, specified in language C as shown in Fig. 5.

```
1: void counter(unsigned int v) {
2:   unsigned int i;
3:   for (i = 0; i < v; i++);
4: }
```

Fig. 5. A simple piece of code.

Now let $\texttt{call\_counter}(u, i)$ be a function that receives an external parameter u, uniformly distributed in interval $[10^{-5}; 1]$, computes the corresponding value of v in two possible ways, either as v_1 (when $i = 1$) or v_2 (when $i = 2$), and calls $\texttt{counter}(v_i)$, where

$$v_1 = 10^3 + \left\lfloor \frac{9000u}{1 - 10^{-5}} \right\rfloor \quad \text{and} \quad v_2 = \left\lceil \frac{10^3}{u^{0.2}} \right\rceil$$

We are interested in determining the pWCET of $\texttt{counter}(v_i)$ when it is called either by $\texttt{call\_counter}(u, 1)$ or by $\texttt{call\_counter}(u, 2)$. Note that in either case the loop in the `counter` function takes from 10^3 up to 10^4 iterations. In our platform, the worst-case behavior represents 1000270 processor cycles.

Fig. 6 illustrates the distribution of maxima for both versions of the counter function, which, as can be seen, have completely different shapes, although the analyzed function is exactly the same. The estimated models are given in Table I. As can be seen, the derived models are clearly Weibull (for $\texttt{counter}(v_1)$) and Fréchet (for $\texttt{counter}(v_2)$); the estimated 95%-confidence intervals for $\hat{\xi}$ are respectively $[-1.170; -0.852]$ and $[0.085; 0.334]$. These models are in line with the histograms shown in Fig. 6. The former exhibits a (very) short bounded tail whereas the latter is characterized by an unbounded heavy tail. Forcing the Gumbel model in these cases may lead to highly unreliable results (recall that the WCET for this example is 1000270 processor cycles). The estimates for pWCET are given in Table II. As can be seen, the Gumbel model in these cases over-estimate or underestimate the values of pWCET. Fig. 7 makes it clear the poor quality of the model fitting when the Gumbel model is assumed as compared with the GEV fit in Figs. 7(c) and 7(d): We observe in the plots for the GEV models some small fluctuations around the diagonal line but no apparent tendency in the data shows up. The estimated GEV models are thus acceptable.

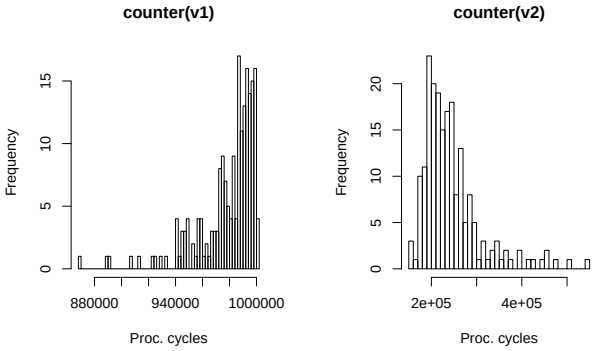


Fig. 6. Distributions of maxima for Ex. 3: $v = v_1$ (left) and $v = v_2$ (right). The maximum (minimum) observed values were 540470 (100270) and 1000170 (470) processor cycles for $\texttt{counter}(v_1)$ and $\texttt{counter}(v_2)$, respectively.

TABLE I
ESTIMATED MODELS FOR EX. 3.

	$\texttt{counter}(v_1)$		$\texttt{counter}(v_2)$	
	GEV	Gumbel	GEV	Gumbel
$\hat{\mu}$	979197.00	976490.75	213115.5	213115.53
$\hat{\sigma}$	21857.92	31736.45	37153.73	37153.74
$\hat{\xi}$	-1.037	$--$	0.207	$--$

C. A relevant observation on input data

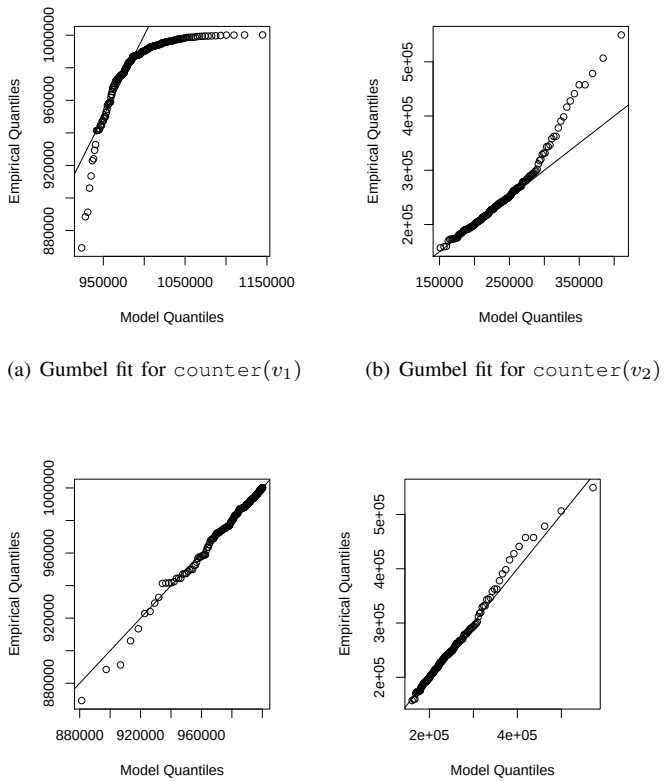
Definition 2 makes the concept of pWCET conditional to the input data. Experimental evidence provided in the previous section backs up the need for this definition. Indeed, when analyzing the v_1-type counter in Ex. 3, the distribution of its execution time maxima converges to a Weibull distribution. The raw data is uniformly distributed within interval $[10^3; 10^4]$ since u is uniformly distributed within interval $[10^{-5}; 1]$. In practice, the application environment may work differently, with data distributed as v_2 for instance. In this case, pWCET estimates derived during analysis would not conform with what would be observed in practice, which may lead to unsafe estimations. Determining which data input distributions suit each application is thus a relevant problem to be dealt with by the analyst. Simply applying uniformly distributed data during the time analysis phase may bring poor or even unsafe estimations. This is a problem to be addressed in future work.

V. PWCET ON A TIME-UNPREDICTABLE PLATFORM

In this section we consider data and instruction caches to analyze their effects on pWCET estimation and EVT-based analyzability. For each cache size (determined by the value of κ), a GEV model is estimated, and goodness-of-fit and extreme

TABLE II
PWCET FOR EX. 3 GIVEN BY GEV AND GUMBEL MODELS.

	$\texttt{counter}(v_1)$		$\texttt{counter}(v_2)$	
	$C(10^{-2})$	$C(10^{-4})$	$C(10^{-2})$	$C(10^{-4})$
GEV	1000086	1000263	491313	1182207
Gumbel	1122484	1268793	384029	555313



(a) Gumbel fit for counter(v_1)

(b) Gumbel fit for counter(v_2)

(c) GEV fit for counter(v_1)

(d) GEV fit for counter(v_2)

Fig. 7. QQ-Plots for the estimated Gumbel and GEV models given in Table I derived for Ex. 3. Both Gumbel models must be rejected whereas the GEV models are acceptable.

value condition tests are carried out similarly to what has been shown in the previous section. We show here only some of the plots. To determine the effect of time-unpredictability has on the analyzability, we have selected examples for which no estimation was possible on a time-predictable (*i.e.,* $\kappa = 0$) platform either because no good model fitting was possible or because the data did not pass convergence tests. We reconsider the examples analyzed in Section IV-A and introduce other two algorithms below, both from the Malärdalen Benchmark.

Example 4 (Square root): It consists of a function that receives a float point number and returns its square root by approximation, obtained by carrying out up to 19 iterations. If the results of two consecutive iterations is within 10^{-5} of each other, the computation stops earlier. Randomly generated integer number, uniformly distributed within interval $[2; 1000]$, were considered as input data. The WCET was determined to be 703730 processor cycles (for $\kappa = 0$), which corresponds to the case for $\sqrt{288}$.

Example 5 (Primality test): The primality test is carried out by a function that receives an unsigned integer v and returns 1 when v is prime or 0 when it is not. It first checks whether v is even. If it is not, it performs a loop verifying whether this number is divisible by any odd integer $i \in [3; \sqrt{v}]$. If v is not even and is not divisible by all these numbers, it is considered prime. Instead of computing the square root of v, the loop bound is verified by checking whether $i \times i \leq v$. Divisibility verification is carried out by another function which uses the `mod` operator. This construction avoids the use of float pointer instructions, which, for our architecture, implies a smaller code size when compared with Ex. 4, for instance. The WCET of this example was determined to be 63670 processor cycles (for $\kappa = 0$), corresponding to the execution taking $v = 99991$, namely the largest prime less than the considered upper limit in our experiment, whose interval was set to $[2; 10^5]$.

Next, we report the found results for Examples 1, 2, 4, and 5 taking into consideration different cache sizes and two cache arrangements: instruction and data caches occupy distinct spaces; or they share the same space.

A. Separated cache space for instruction and data

Table III indicates whether reliable GEV models were obtained for each example. As can be seen in the first and last columns, no GEV models could be accepted for scenarios with no cache or when large caches were employed. When the cache size becomes larger than the working set of the analyzed task, its observed execution time tends to present less variability, which in turn makes the model fitting harder. Data collected for Ex. 1 did not pass convergence tests for any cache set-up and thus no reliable model estimation was possible. The reasonably predictable behavior of the binary search algorithm (not so many time execution values – recall Fig. 2) and its small working set may explain the reasons for this impossibility. The absence of convergence to EV distributions also explains estimated model the rejection for Ex. 5 with $\kappa \geq 128$.

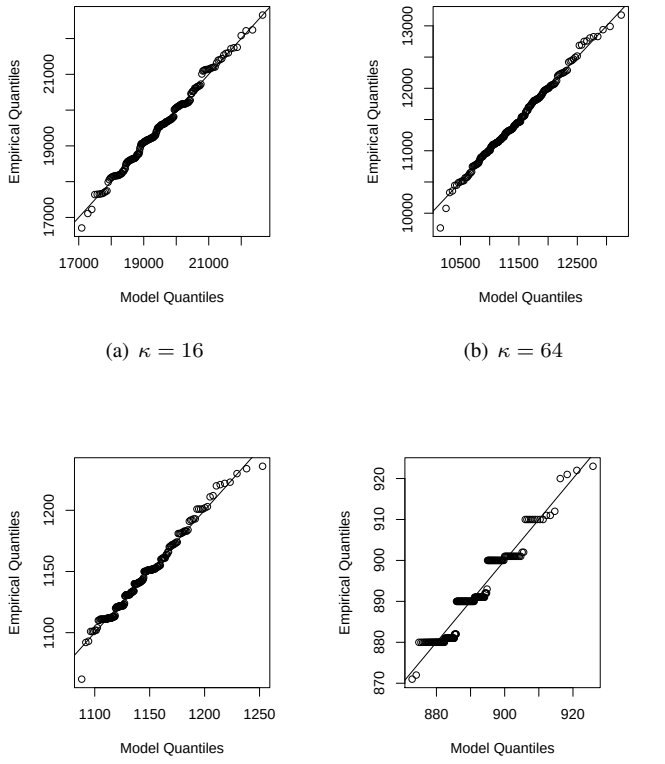
TABLE III
EFFECT OF CACHE SIZE ON MODEL FITTING (SEPARATED CACHES).

	κ						
Ex.	0	16	32	64	128	256	2048
1	–	–	–	–	–	–	–
2	–	✓	✓	✓	✓	–	–
4	–	✓	✓	–	✓	✓	–
5	–	✓	✓	✓	–	–	–

For illustration purposes, Fig. 8 depicts the model fitting as a function of cache size for Ex. 2. It is clear from Fig. 8(d) that the estimated model must be rejected. It presents similar behavior to scenarios for $\kappa = 0$, for instance. Figs. 8(a) and 8(b) show reasonable fits and so the models are to be accepted: The points (of the empirical distribution quantiles) are in reasonable agreement with the estimated models. The model fitting shown in Fig. 8(c), on the other hand, has questionable quality. In high quantiles the points seem to present a consistent deviation from the diagonal line and some discretization starts to appear although this latter effect is much less than what is observed in Fig. 8(d). In this work we conservatively rejected any estimated models if their quality could not be undoubtedly verified. The estimated model for Ex. 4 when $\kappa = 64$ was also not accepted for this reason.

Accepting models with small quantile deviations can be justified if complementary supporting arguments are presented. In the case of Fig. 8(c), if the shown deviation of high quantiles was the only problem, the estimated model would be overestimating the empirical distribution, which could be acceptable depending on the application.

The scenarios for $\kappa = 16$ and $\kappa = 256$ were verified to be possible corner cache configurations as for model fitting. Smaller or larger values of κ yield fitting difficulties, respectively.



(a) $\kappa = 16$ (b) $\kappa = 64$

(c) $\kappa = 256$ (d) $\kappa = 2048$

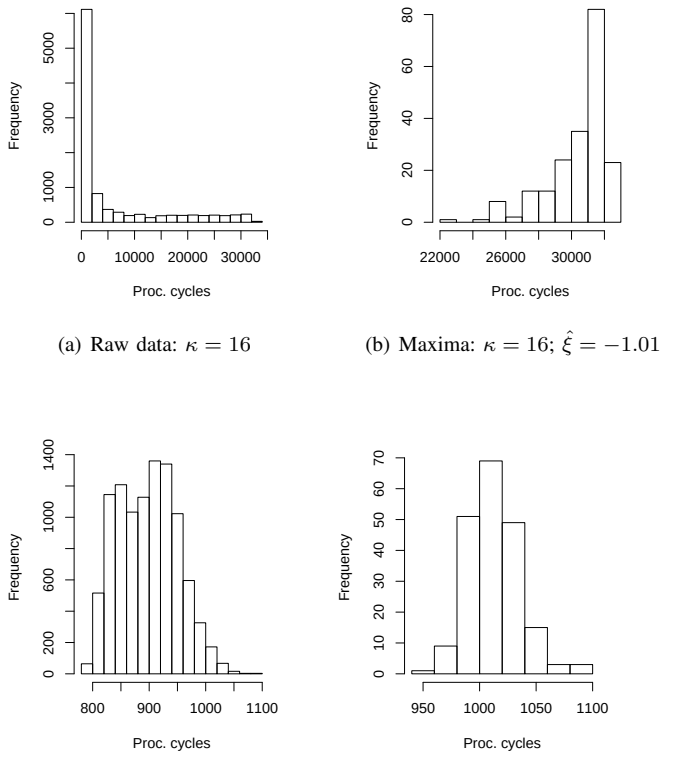
Fig. 8. QQ-plots for Ex. 2 for different cache sizes.

TABLE IV
ESTIMATED DISTRIBUTION SHAPE FOR DIFFERENT CACHE SIZES. INSTRUCTION AND DATA ARE CACHED SEPARATELY.

	Estimated 95%-confidence intervals for ξ		
κ	Ex. 2	Ex. 4	Ex. 5
16	$(-0.27; -0.07)$	$(-0.18; 0.03)$	$(-1.15; -0.83)$
32	$(-0.27; -0.07)$	$(-0.20; 0.02)$	$(-1.15; -0.82)$
64	$(-0.24; -0.05)$	–	$(-0.22; 0.00)$
128	$(-0.14; 0.06)$	$(-0.52; -0.20)$	–
256	–	$(-0.54; -0.32)$	–

As can be seen from Table IV, the GEV models for the considered examples and cache configurations are better described by the Weibull distributions. Gumbel fittings can be applied for Ex. 2 ($\kappa = 128$), Ex. 4 ($\kappa = 16, 32$), and Ex. 5 ($\kappa = 64$). The Fréchet type is less evident for these examples, although some intervals admit $\xi > 0$.

The low negative values for Ex. 5 ($\kappa = 16, 32$) indicate a very short bounded tail. Fig. 9(b) illustrates this for $\kappa = 16$. Note that most observed execution times for this example are low, as shown in Fig. 9(a). High execution times are observed for odd large numbers, some of which are prime. The execution time increases proportionally to the value of these odd numbers. As odd numbers are uniformly distributed within the input interval, we observe that a uniform distribution above a certain value. The distribution of maxima thus tends to be formed with large execution times uniformly distributed in different data blocks. This explains the observed tail since the maximum of uniform distribution converges to Weibull [8]. However, when the cache size is large enough to produce data disturbance, such an effect is decreased and the tail tends to be longer (less negative shape). These observations are illustrated in Fig. 9 for $\kappa = 64$. For higher values of κ, however, no reliable model was obtained.



(a) Raw data: $\kappa = 16$ (b) Maxima: $\kappa = 16$; $\hat{\xi} = -1.01$

(c) Raw data: $\kappa = 64$ (d) Maxima: $\kappa = 64$; $\hat{\xi} = -0.10$

Fig. 9. Distributions of sampled data for Ex. 5. The higher the cache size, the higher the observed data variability leading to longer tail.

Once confidence in the estimated models is ensured, derived values of pWCET can be obtained. For the configurations under analysis, this information can be useful for assigning cache partitions to tasks sizes, similarly to what has been done by other researchers, *e.g.,* [15], [16], [17]. Table V presents the estimated pWCET ($C(10^{-4})$) for the considered algorithms as a function of cache sizes. The higher values for Ex. 4 are due to fact that in our experiment platform float point operations

are translated into integer instructions by the compiler, making the executable code considerably larger.

TABLE V
$C(10^{-4})$ IN PROC. CYCLES FOR DIFFERENT CACHE SIZES. INSTRUCTIONS AND DATA ARE CACHED SEPARATELY.

	κ				
Ex.	16	32	64	128	256
1	–	–	–	–	–
2	23856	21509	14032	2419	–
4	624407	489225	–	337514	293110
5	42560	32324	1889	–	–

B. Single cache space for instruction and data

Results from previous section confirm that time-unpredictability introduced by randomized caches improve analyzability, although it is not enough to ensure that reliable estimation of pWCET is possible. Too large or too small cache sizes were observed to make the estimation harder. In this section we continue this study by considering that the same cache space is used for both instruction and data. This arrangement is expected to cause more execution time variability and we check whether it benefits analyzability.

Table VI shows that 13 GEV models out of 35 could be obtained as opposed to 11 in Table III. The model for Ex. 5 and $\kappa = 32$ could not be accepted contrasting to its counterpart in Table III; but a model for Ex. 4 with κ as high as 2048 could be obtained. That is, although there is an overall tendency that higher execution time variability improves analyzability, this alone may not suffice.

Indeed, randomization seems to be a key aspect in EVT-based time analysis since it smooths out the discreteness of the measurements (recall that EV distributions are continuous functions). Thus, the more discrete the empirical distribution appears to be, the harder it is to derive an EV model that reliably fits to the measured data. However, we emphasize that relying only on hardware randomization for achieving good model fitting may not suffice; no model fitting was possible for Ex. 1, for instance. The problem of whether it is possible to ensure that execution time measurements is EVT-compliant is thus an issue deserving further investigation.

TABLE VI
EFFECT OF CACHE SIZE ON MODEL FITTING (SINGLE CACHE).

	κ						
Ex.	0	16	32	64	128	256	2048
1	–	–	–	–	–	–	–
2	–	✓	✓	✓	✓	✓	–
4	–	✓	✓	✓	✓	✓	✓
5	–	✓	–	✓	–	–	–

Tables VII and VIII can be used for comparison with the estimates obtained in the previous section. As for the shape parameter, although the values in Table IV and VII share some similarities, we could not notice any pattern. The values in Tables V and VIII were found consistent with the observed cache hit ratios. For example, the mean cache hit ratio for Ex. 2 with $\kappa = 32$ is equal to 0.52 when instructions and data are cached separately whereas it is 0.81 when they share the same cache space. When there is a sudden drop in pWCET estimates in function of κ, there is a corresponding increase in cache hit ratios, indicating the consistence of the estimated models.

TABLE VII
ESTIMATED DISTRIBUTION SHAPE FOR DIFFERENT CACHE SIZES. INSTRUCTION AND DATA ARE CACHED IN A SHARED SPACE.

	Estimated 95%-confidence intervals for ξ		
κ	Ex. 2	Ex. 4	Ex. 5
16	$(-0.25; -0.06)$	$(-0.16; 0.05)$	$(-1.12; -0.81)$
32	$(-0.29; -0.09)$	$(-0.60; -0.37)$	–
64	$(-0.15; 0.04)$	$(-0.48; -0.27)$	$(-0.12; 0.09)$
128	$(-0.19; 0.02)$	$(-0.35; -0.16)$	–
256	$(-0.15; 0.06)$	$(-0.51; -0.30)$	–
2048	–	$(-0.17; 0.03)$	–

TABLE VIII
$C(10^{-4})$ IN PROC. CYCLES FOR DIFFERENT CACHE SIZES. INSTRUCTIONS AND DATA ARE CACHED IN A SHARED SPACE.

	κ					
Ex.	16	32	64	128	256	2048
1	–	–	–	–	–	–
2	25862	18721	5563	1690	1243	–
4	537564	444025	406757	346863	264092	12279
5	46126	–	1334	–	–	–

VI. CONCLUSIONS

A careful study on EVT-based pWCET estimation has been presented for which the unifying GEV distribution has been used. Extensive experiments have identified scenarios for which no pWCET estimation was possible and that, when EVT can be safely applied, any of the three extreme value distributions (Weibull, Gumbel or Frechét) may arise. Experimental evidences indicating that being restricted to a single distribution family, as it has been the case in most previous work, may lead to poor or even unsafe pWCET estimates. The found results also indicate that the use of hardware randomization via random caches improves EVT analyzability, although they also have shown that this randomization may not suffice. Too large or too small caches were found not sufficient to cause the necessary effects for good model fitting even under the random replacement policy.

Applying EVT frameworks for real-time systems analysis is a relatively new research branch and some issues still deserve attention. For example, the data collected for analysis are produced by the analyst. We have illustrated that this can be a problem in cases where the assumption made during analysis differs from how applications actually run in practice. Further studies related to the extent to which one needs randomization to ensure analyzability via EVT must also be further investigated. Relying on random effects caused by hardware or by the system may not be enough for good pWCET estimates. These and other open problems require careful studies to identify boundaries in applying EVT for real-time systems. Information provided in this paper certainly contributes to such developments.

ACKNOWLEDGMENT

This work has been jointly funded by FAPESB (grant numbers 4932/2015 and 9005/2015) and CNPq (grant number 456193/2014-6). The authors would like to thank the anonymous reviewers and Dr. Verônica Lima for their comments on previous versions of this paper.

REFERENCES

[1] F. Ashkar, N. El-Jabi, and S. Sarraf, "Study of hydrological phenomena by extreme value theory," *Natural Hazards*, vol. 4, no. 4, pp. 373–388, 1991.

[2] F. Longin, *Extreme Events in Finance: A Handbook of Extreme Value Theory and Its Applications*, ser. Wiley Handbooks in Financial Engineering and Econometrics. Wiley, 2016.

[3] J. H. J. Einmahl and J. R. Magnus, "Records in athletics through extreme-value theory," *Journal of the American Statistical Association*, vol. 103, no. 484, pp. 1382–1391, 2008.

[4] H. Drees, L. Haan, and D. Li, "Approximations to the tail empirical distribution function with application to testing extreme value conditions," *Extremes*, vol. 136, pp. 3498–3538, 2006.

[5] D. Dietrich, L. de Haan, and J. Hüsler, "Testing extreme value conditions," *Extremes*, vol. 9, pp. 75–85, 2002.

[6] J. Hüsler and D. Li, "On testing extreme value conditions," *Extremes*, vol. 9, pp. 69–86, 2006.

[7] R. A. Fisher and L. H. C. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," *Proc. of the Cambridge Philosophical Society*, vol. 24, pp. 180–190, 1928.

[8] S. Coles, *An Introduction to Statistical Modeling of Extreme Values*, 3rd ed. Springer-Verlag, 2001.

[9] J. Jalle, L. Kosmidis, J. Abella, E. Quiñones, and F. Cazorla, "Bus designs for time-probabilistic multicore processors," in *Proc. of the Conference on Design, Automation and Test*, 2014.

[10] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla, "A cache design for probabilistically analysable real-time systems," in *Proc. of the Conference on Design, Automation and Test in Europe*, 2013, pp. 513–518.

[11] L. Kosmidis, E. Quiñones, J. Abella, T. Vardanega, I. Broster, and F. Cazorla, "Measurement-based probabilistic timing analysis and its impact on processor architecture," in *Proc. of the 17th Euromicro Conference on Digital System Design*, 2014, pp. 401–410.

[12] J. Abella, E. Quiñones, F. Wartel, T. Vardanega, and F. Cazorla, "Heart of gold: Making the improbable happen to increase confidence in MBPTA," in *Proc. of the 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 255–265.

[13] J. Reineke, "Randomized caches considered harmful in hard real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 1, no. 1, pp. 03–1–03:13, 2014.

[14] E. Mezzetti, M. Ziccardi, T. Vardanega, J. Abella, E. Quiñones, and F. Cazorla, "Randomized caches can be pretty useful to hard real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 2, no. 1, pp. 01–1–01:10, 2015.

[15] B. Bui, M. Caccamo, L. Sha, and J. Martinez, "Impact of cache partitioning on multi-tasking real time embedded systems," in *Proc. of the 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2008, pp. 101–110.

[16] S. Altmeyer, R. Douma, W. Lunniss, and R. Davis, "Evaluation of cache partitioning for hard real-time systems," in *Proc. of the 26th Euromicro Conference on Real-Time Systems*, July 2014, pp. 15–26.

[17] S. Altmeyer, R. Douma, W. Lunniss, and R. I. Davis, "On the effectiveness of cache partitioning in hard real-time systems," *Real-Time Systems*, pp. 1–46, January 2016.

[18] R. Azevedo, S. Rigo, M. Bartholomeu, G. Araujo, C. Araujo, and E. Barros, "The ArchC architecture description language and tools," *International Journal of Parallel Programming*, vol. 33, no. 5, pp. 453–484, 2005.

[19] L. Cucu-Grosjean, "Independence - a misunderstood property of and for probabilistic real-time systems," in *Real-Time Systems: the past, the present and the future*, S. Baruah and N. Audsley, Eds. CreateSpace Independent Publishing Platform, 2013, pp. 29–37.

[20] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A new way about using statistical analysis of worst-case execution times," *SIGBED Rev.*, vol. 8, no. 3, pp. 11–14, 2011.

[21] B. V. Gnedenko, "Sur la distribution limite du terme maximum d'une série aléatoire," *Annals of Mathematics*, vol. 44, pp. 423–453, 1928.

[22] A. F. Jenkinson, "The frequency distribution of the annual maximum (or minimum) values of meteorological events," *Quarterly Journal of the Royal Meteorological Society*, vol. 81, pp. 158–172, 1955.

[23] R. von Mises, "La distribution de la plus grande de n valeurs," in *Selected papers*, 1954, vol. vol. II, pp. 271–294.

[24] E. J. Gumbel, *Statistics of Extremes*. Columbia University Press, 1958.

[25] J. R. M. Hosking, "L-moments: analysis and estimation of distributions using linear combinations of order statistics," *Journal of the Royal Statistical Society*, vol. 52, pp. 105–124, 1990.

[26] R. L. Smith, "Maximum likelihood estimation in a class of non-regular cases," *Biometrika*, vol. 72, pp. 67–90, 1985.

[27] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1994.

[28] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2015. [Online]. Available: http://www.R-project.org/

[29] E. Gilleland and R. W. Katz, "New software to analyze how extremes change over time," *Eos*, vol. 92, no. 2, pp. 13–14, 2011.

[30] D. Griffin and A. Burns, "Realism in statistical analysis of worst case execution times," in *Proc. of 10th Workshop on Worst-Case Execution Time Analysis*, 2010, pp. 44–53.

[31] F. J. Cazorla, T. Vardanega, E. Quiñones, and J. Abella, "Upper-bounding program execution time with extreme value theory," in *Proc. of the 13th International Workshop on Worst-Case Execution Time Analysis*, C. Maiza, Ed., vol. 30, 2013, pp. 64–76.

[32] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quiñones, and F. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *Proc. the 24th Euromicro Conference on Real-time Systems*, 2012, pp. 91–101.

[33] A. Zempléni, "Goodness-of-fit test in extreme value applications," Sonderforschungsbereich 386 der Ludwig-Maximilians-Universität München, München, Discussion paper 383, 2004.

[34] J. Hüsler and D. Li, "Testevc1d – r package," 2006. [Online]. Available: http://my.gl.fudan.edu.cn/teacherhome/lideyuan/pdf/TestEVC1d.txt

[35] B. Lesage, D. Griffin, S. Altmeyer, and R. Davis, "Static probabilistic timing analysis for multi-path programs," in *Proc. of the IEEE Real-Time Systems Symposium*, 2015, pp. 361–372.

[36] G. Bernat, A. Colin, and S. M. Petters, "WCET Analysis of Probabilistic Hard Real-time Systems," in *Proc. of the 23rd IEEE Real-Time Systems Symposium*, 2002, pp. 279–288.

[37] A. C. G. Bernat and S. Petters, "pWCET: A tool for probabilistic worst-case execution time analysis of real- time systems," University of York., Tech. Rep. YCS-2003-353, 2003.

[38] "Rapita Systems LTD," http://www.rapitasystems.com.

[39] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem – overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 36:1–36:53, May 2008.

[40] S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *Proc. IEEE Real-Time Systems Symposium*, 2001.

[41] J. Hansen, S. Hissam, and G. Moreno, "Statistical-based wcet estimation and validation," in *Proc. 9th International Workshop on Worst-Case Execution Time Analysis*, 2009, pp. 123–133.

[42] L. Kosmidis, T. Vardanega, J. Abella, E. Quiñones, and F. J. Cazorla, "Applying measurement-based probabilistic timing analysis to buffer resources," in *13th International Workshop on Worst-Case Execution Time Analysis*, 2013, pp. 97–108.

[43] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A statistical response-time analysis of real-time embedded systems," in *Proc. of the 33rd Real-Time Systems Symposium*, 2012, pp. 351–362.

[44] D. Maxim, F. Soboczenski, I. Bate, and E. Tovar, "Study of the reliability of statistical timing analysis for real-time systems," in *Proc. of the 23rd International Conference on Real Time and Networks Systems*. ACM, 2015, pp. 55–64.

[45] L. Kosmidis, J. Abella, F. Wartel, E. Quiñones, A. Colin, and F. Cazorla, "PUB: Path upper-bounding for measurement-based probabilistic timing

analysis," in *Proc. of the 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 276–287.

[46] L. Kosmidis, J. Abella, E. Quiñnones, and F. Cazorla, "Multi-level unified caches for probabilistically time analysable real-time systems," in *Proc. of the 34th IEEE Real-Time Systems Symposium*, Dec 2013, pp. 360–371.

[47] A. Melani, E. Noulard, and L. Santinelli, "Learning from probabilities: Dependences within real-time systems," in *Proc. of the 18th IEEE Conference on Emerging Technologies Factory Automation*, Sept 2013, pp. 1–8.

[48] F. Guet, L. Santinelli, and J. Morio, "On the reliability of the probabilistic worst-case execution time estimates," in *8th European Congress on Embedded Real-Time Software and Systems*, 2016, p. 10 pages.

[49] K. Berezovskyi, F. Guet, L. Santinelli, K. Bletsas, and E. Tovar, "Measurement-based probabilistic timing analysis for graphics processor units," in *Proc. of the 29th International Conference on Architecture of Computing Systems*, F. Hannig, M. J. Cardoso, T. Pionteck, D. Fey, W. Schröder-Preikschat, and J. Teich, Eds. Springer International Publishing, 2016, pp. 223–236.

[50] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart, "On the sustainability of the extreme value theory for wcet estimation," in *14th International Workshop on Worst-Case Execution Time Analysis*, H. Falk, Ed., vol. 39, 2014, pp. 21–30.

[51] A. Waterman, Y. Lee, D. Patterson, and K. Asanovic, "The RISC-V instruction set manual, volume I: Base user-level ISA," EECS Department, University of California, Berkeley, CA, Tech. Rep. UCB/EECS-2011-62, 2011.

[52] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The Mälardalen WCET benchmarks – past, present and future," in *Proc. of Workshop on Worst-Case Execution Time Analysis (WCET)*, B. Lisper, Ed. Brussels, Belgium: OCG, Jul. 2010, pp. 137–147.