Modelling and Predicting Extreme Behavior in Critical Real-Time Systems with Advanced Statistics

Sergi Vilardell Moreno

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy in Computer Architecture

Supervisors: Dra. Isabel Serra & Dr. Jaume Abella

Tutor: Dr. Francisco J. Cazorla

January 25, 2023

Acknowledgements

I wish to thank in general all people involved, directly or not, in my journey to complete my PhD. studies. While I am the main driver of this thesis, it has been a collaborative effort with a strong network of technical and emotional support.

First and foremost, thanks to my first co-advisor Dr. Jaume Abella, and my tutor Dr. Francisco J. Cazorla, for the opportunity to develop my thesis under their excellent guidance. I would also like to thank Dr. Enrico Mezzetti for his contributions throughout these years. Finally, my most sincere gratitude towards my second co-advisor Dr. Isabel Serra for being the responsible of my development as a scientist and putting her trust in me from the very beginning by recommending me to pursue my thesis in Barcelona Supercomputing Center. I want to also thank Laurent Rioux for the opportunity to make an internship at Thales.

This thesis would not be possible without the great work of the students and engineers of the Computer Architecture / Operating System group by paving the way towards incorporating statistical analysis in the Computer Architecture field.

On the personal side, I wish to thank my family, Joan, Puri, and Enric, for raising me up to what I am today and always pushing me towards my real capabilities with tenderness and support. To my dear friends, the Brotherhood, for all those wonderful crucial years for our growth as people and scientists. In particular, my deep gratitude to Alejandro for being always there despite time and distance. Finally, to my dear love and main support during these years, Olaya, who demonstrated that not only technical ability is necessary to carry on a thesis, but that her love and emotional support has been the key to this journey.

This work has been partially supported by Barcelona SuperComputing Center and the Spanish Ministry of Economy and Competitiveness (MINECO) under grant PID2019-110854RB-I00 / AEI / 10.13039/501100011033 and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 772773).

Abstract

Critical Real-Time Embedded Systems (CRTES) are used in domains like transportation (e.g. avionics, automotive, space, and railway), healthcare, and industrial machinery. This subset of embedded systems requires undergoing a stringent Validation & Verification (V&V) process before they are allowed to enter in operation since any misbehavior can result in harm of humans or even fatalities. Software timing behavior is a key element to cover in the V&V process, providing with evidence that software runs timely. Software timing analysis, in turn, requires deriving bounds to each application task's execution time. These bounds are referred to as Worst-Case Execution Time (WCET) estimates.

As CRTES implement more complex safety-related functionalities in every new product, more complex software and consequently more performant computing hardware is used to satisfy the high-performance requirements. The side effect, however, of using more complex hardware and software is challenging state-of-the-art software timing analysis techniques.

Measurement-Based Probabilistic Timing Analysis (MBPTA) techniques have been proposed to handle such hardware and software complexity providing tight and trustworthy WCET bounds (estimates). Specifically, Extreme Value Theory (EVT) has been used to provide with models for the most extreme occurrences in form of a probabilistic distribution. The output of the timing model is referred as a probabilistic WCET (pWCET). However trustworthy, EVT models can be cumbersome to apply and they sometimes can be exceedingly pessimistic which adds extra cost into timing budgets.

This thesis investigates MBPTA techniques and develops novel methodologies within this framework in three distinct fronts. Firstly, by improving the tightness of pWCET models on sky-high quantiles with two models. A first one that combines risk analysis with EVT for a safe and accurate pWCET. And a second one that introduces Markov's Inequality to the pWCET estimation problem, which provides with trustworthy guarantees with less requirements for its correct application. Secondly, in order to boost the use of data coming from performance monitoring counters - increasingly used by MBPTA techniques to tighten estimates-, this thesis shows two mathematically-based ways of merging multiple disjointed readings based on order statistics and copula models. Finally, this thesis proposes a model for the contention of competing tasks, when the timing profile obtained is limited, that allows to provide with more extreme WCET scenarios based on the dependencies between tasks.

Summarizing, this thesis pushes the state-of-the-art forward in the V&V methodologies for CRTES in the framework of MBPTA in terms of WCET estimation and data gathering.

Contents

List of Figures

List of Tables

List of Acronyms

AD Autonomous Driving. 7

BM Block Maxima. 5

CCDF Complementary Cumulative Distribution Function. 11
CDF Cumulative Distribution Function. 11
CES Critical Embedded Systems. 3
CLT Central Limit Theorem. 13
CRTES Critical Real-Time Embedded Systems. 3
CV Coefficient of Variation. 32

ES Embedded Systems. 3
ETCS European Train Control System. 24
EVI Extreme Value Index. 6
EVT Extreme Value Theory. 5

GEV Generalized Extreme Value. 6
GPD Generalized Pareto Distribution. 6
GPUs Graphics Processing Units. 21

HEMs Hardware Event Monitors. 4
HRM HEM-Reading Merging. 63

i.i.d independent and identically distributed. 12

MBPTA Measurement-Based Probabilistic Timing Analysis. 5
MBTA Measurement-Based Timing Analysis. 4
MPSoC Multi-Processor System on Chip. 4
MUCH MUlti-Correlation HEM reading and merging. 78

PMCs Performance Monitoring Counters. 5
PMU Performance Monitoring Unit. 5
PoT Peaks over Threshold. 5
pWCET probabilistic Worst-Case Execution Time. 5

r.v. random variable. 11

STA Static Timing Analysis. 4

V&V Validation and Verification. 3

WCET Worst-Case Execution Time. 3

Part I

Introduction, Background, and Experimental Methodology

Chapter 1

Introduction

Historically, computer systems have been used mostly for automation and entertainment. This trend has changed in recent years with computers being increasingly used in a number of critical domains of our life including transportation and health, among others. In these domains, computers are embedded into devices like a car, a train, an airplane, and medical machinery. Unlike other Embedded Systems (ES) in which a malfunctioning of the computer causes mostly discomfort, a malfunction in the so-called Critical Embedded Systems (CES) can cause serious harm to the environment, or people, even resulting in fatalities. In order to ensure that these systems are safe to use in critical scenarios, it is imperative that their design satisfies functional correctness [61] tested according to a well-defined Validation and Verification (V&V) process. Functional correctness ensures that the system functions as intended without anomalies. On the other hand, timing correctness ensures that the performed tasks complete their execution within the assigned time budget or *deadline*. In this thesis, we focus on CES whose correctness also depends on their timely response, which are categorized as Critical Real-Time Embedded Systems (CRTES). In the domain of CRTES, the functional and timing correctness have equally important weight through the V&V process, given that not meeting the pre-assigned time budget, or *deadline*, can lead to catastrophic consequences. Thus, in timing analysis deriving a conservative estimation of the Worst-Case Execution Time (WCET) is essential. The interest in keeping up with the challenges of timing analysis is due to the increasing growth in CRTES in services like healthcare and industries like avionics, railway, and automotive vehicles [72]. Furthermore, the systems are growing exponentially in complexity: the car manufacturer Volvo stated that in 2020 their cars needed around 100 million lines of code to implement the key functionalities of a car such as electronic fuel injection, transmission control, navigation systems, etc. Their measurements show that since 2000, the automotive software increases an order of magnitude every decade [9]. Their functionalities are also way more complex too. In the case of autonomous cars, the system must handle data coming from radars and LIDARS and process it using state-of-the-art Neural Networks algorithms in real-time, requiring very high levels of performance. To satisfy those needs, CRTES need to rely on high-performance hardware such as multi-level cache memories design and multi-core architectures. While powerful, this kind of design tends to be harder to analyze which makes them more challenging to work with towards an accurate WCET estimation.

1.1 Certification with Higher Performance Requirements

In past years, the prevailing architecture paradigm for the design of CRTES was the *federated architecture*. These architectures comprise multiple subsystems, each with critical functionalities, which are physically separated from each other. The increasing complexity of new functionalities in recent years challenge the scalability of federated systems [96]. The higher performance needs of today's functionalities would rise the cost, weight, and size of CRTES. In that regard, the CRTES industry shifted towards *integrated architecture paradigms* that integrate several functionalities in the same computing hardware. Integrated architectures can also naturally leverage the parallelism of multicore systems and the capabilities of high-performance features like multi-level cache. To that end, software solutions like *resource partitioning* are used to contain the impact that software partitions can generate on each other. While the adoption of integrated architectures has aided the cost-weight-size problem and provided with satisfactory performance, the certification on multicore systems becomes notably more challenging. The loss of guarantees in terms of independence between subsystems within integrated architectures affects their *timing predictability*. Depending on the degree of integration and partitioning the dependency, and therefore predictability, will be affected differently. This usually results in a trade-off between performance and degree of integration. In terms of certification for CRTES, new designs trend towards favoring timing predictability [140] instead of increased average performance.

1.2 Measurement-Based Timing Analysis

Increasingly complex software functionalities in CRTES require high-performance multicore processors (computing hardware). However, more complex processors have drawbacks in relation to the V&V process. Complex CRTES should comply to the same safety standards as the simpler architectures and given the interconnected nature of integrated architecture and the complexity of multicores, the V&V process for timing is very challenging in terms of effort and accuracy [5]. A high-quality WCET estimation in this domain must be affordable in terms of costs and trustworthiness, so that the produced execution time estimate tightly upper-bounds the real execution time.

Timing analysis techniques can be broadly categorized into two groups: Static Timing Analysis (STA) and Measurement-Based Timing Analysis (MBTA) [172]. STA does not need to execute code on real hardware and instead it relies on creating abstract models of the architecture and analyzing possible execution paths for the task under analysis. This approach faces limitations due to the limited information about the hardware provided by the hardware manufacturer and its accuracy [1]. In fact, some processor manufacturers do not disclose full information about their systems' microarchitecture in order to protect intellectual property. Even assuming the validity and completeness of the information disclosed, the technical manual of a modern Multi-Processor System on Chip (MPSoC) like the Zynq UltraScale+ is more than a thousand pages long [179]. The methods that worked for simpler single core processor architectures do not scale well with the complexity that CRTES have nowadays. In less complex architectures, the approaches for V&V were usually guided by previous engineering experience [41].

MBTA considers a different approach, in which timing measurements of a given task in a given hardware guide the analysis. The CRTES system engineer is responsible for designing experiments with stressful initial conditions for the task running in real hardware or simulator. Those experiments should reflect the timing behavior at operation so that the bounds derived from it are reliable. However, measurements in themselves are not treated as bounds [173], given the general impossibility of reproducing and forcing those execution conditions that lead to the worst-case scenario. Industry standard solutions to derive bounds from measurements rely on adding safety margins. For instance, on top of the high-water mark, a safety margin (e.g. 20%) can be added. The logic behind it is to compensate the unobserved effects that can impact observed values. However, for complex hardware deriving a safety margin is significantly more complex.

1.3 Hardware Event Monitors

The execution time of a program in a complex computing platform is the product of many interactions happening within the different hardware blocks. In complex CRTES these interactions are hard to control and model. For instance, the functionality of multi-level cache memories is deeply connected to the variability of the program's execution time. Smaller level 1 caches directly connected to the core have low latency and their data can be quickly accessed. Due to cost and convenience, caches at a higher-level (e.g. L2) are bigger in size, more distant from the core, and the time required to access them increases. A program that tends to perform accesses to the higher-level caches, or the off-chip memory, will suffer higher latency to fetch data and instructions than the one which mostly operates at low-level caches.

Many of these events describing the usage of hardware computing resources made by a task can be tracked via Hardware Event Monitors (HEMs) which can be hence used for software timing analysis purposes. In fact, several MBTA techniques build on HEMs to model and control contention for instance by limiting the maximum number of accesses a task is allowed to do to a given resource [32, 125].

HEMs are programmed and accessed by the Performance Monitoring Unit (PMU) in the processor, which is also referred to as statistics unit. The PMU offers a set of visible registers called Performance Monitoring Counters (PMCs) which can be configured to read the number of occurrences of a given HEM.

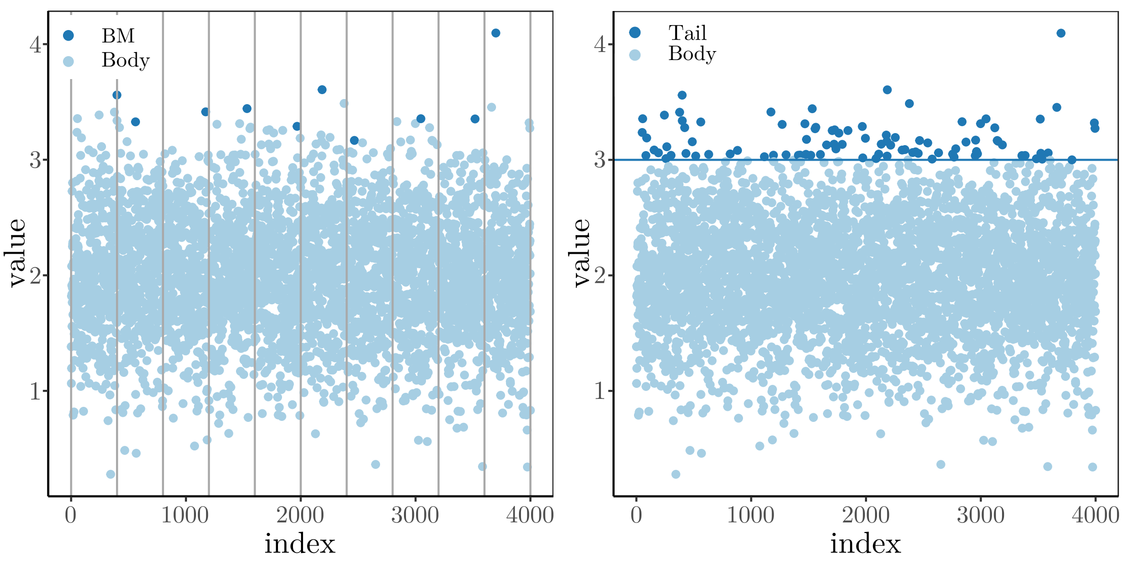
1.4 MBPTA and Extreme Value Theory

In an effort towards relieving some of the burden on the application of MBTA to complex systems, Measurement-Based Probabilistic Timing Analysis (MBPTA) approaches have been gaining traction [3, 21, 23, 44] in CRTES over more than a decade ago. The variety of execution conditions complex CRTES can produce causes huge differences between runs of the same task and great variance in the distribution of execution times. Hence, instead of aiming for a fixed and likely pessimistic WCET, a better alternative is to provide a probabilistic view into timing analysis [36]. Instead of fixing a single value that upper-bounds all executions, MBPTA provides a distribution that upper-bounds the probability of exceeding any specific execution time, although we are interested in high values. For each arbitrarily rare occurrence, an MBPTA tool provides a probability of exceeding it, thus a probabilistic Worst-Case Execution Time (pWCET) estimate. That is, on MBPTA the results are given in the form of a *distribution*.

When dealing with the distribution of arbitrarily rare occurrences, Extreme Value Theory (EVT) is the theoretical best approach [40]. In particular, the appeal of extreme value analysis is to estimate the probabilities of events which are rarer than the ones observed. With EVT one can assess for instance, the likelihood of an earthquake bigger than the ones we have observed so far. Given the generality of its results, EVT has been applied in multiple fields. EVT is still present in fields which have very little in common between them like traffic safety [79], finance [27], earthquake analysis [31], biophysical science [120] and even musical analysis [146]. The different nature of each application field requires to adjust the approach to each problem instance, but the underlying laws are still the same. Of course, for instance, in case of earthquakes one cannot generate more data so it makes sense that EVT is used in this domain. Nonetheless, EVT is also applied in fields where data is readily available, but the cost or even the chance to obtain it is barely possible or feasible. In the case of WCET estimation, the execution time of a program may be relatively short, but obtaining extreme execution times may take weeks or even months. Therefore, one must resort to construct a pWCET model with smaller yet representative set of data. And it is in the word *representative* where a big part of the challenges arises in MBPTA. Obtaining stressful benchmarks for the CRTES at analysis that generates the most extreme execution conditions in the system is a very challenging task. In my view, there are two sides to the MBPTA problem. The first is the technical aspect of computer science that is concerned about designing hardware that is compliant with MBPTA experimentation, and working towards obtaining representative extreme conditions [18, 117]. The second one, the focus of this thesis, is the modelling aspect of the estimation of unseen extreme execution times under the upper-bounding condition.

There have been several mentions to extreme values without proper definition. In EVT there are two classical ways of thinking about extremes. Since this thesis is concerned about WCET estimation, we consider extremes as the high-end values of our measurements, although EVT also work for the lowest values.

- The first way is considering that the extremes are only the maximum value on your sample. Imagine we have a sample of size n and we divide it into separate groups of size S. Then, for each group, we obtain the maximum value on it. This is called the Block Maxima (BM) approach.
- The second way is considering extremes as the group of values above a certain threshold. This is the Peaks over Threshold (PoT) approach. To illustrate them, Figure 1.1a and 1.1b represent the set of values deemed as extremes using the BM and PoT approach respectively. In this example, the sample of data is the same, yet the set of values deemed as extreme is different. There are values classified as extremes in Figure 1.1b which are classified as belonging to the body of the



(a) *Block Maxima* **(b)** *Peaks Over Threshold*

Figure 1.1: *Both classical EVT ways of considering extremes.*

distribution in Figure 1.1a. Not only that, but one can also see how changing the size of the block S or the location of the threshold will change the group of extreme values. This is not inconsistent within EVT and both approaches yield good models for extremes. Yet, the selection of the block size or the threshold is a well-known problem with many different approaches as we will see in Chapter 3. The key finding, and the core of EVT, is that these extreme values have a distribution of their own. The resulting extremes from the BM approach tend to a family of distributions called the Generalized Extreme Value (GEV) distribution [65], while the extremes from PoT converge to another family of distributions, named the Generalized Pareto Distribution (GPD) [15]. These results, under certain conditions, are general to any random variable; which allows us to model rainfall annual data [91] and execution times from CRTES under the same theoretical umbrella. These theoretical family of distributions are named so because there are different distributions within them depending on the Extreme Value Index (EVI) parameter represented as ξ. The EVI is what characterizes the shape of the most extreme values of a distribution.

For the PoT model the EVI determines the shape of the *tail* of the distribution in three ways. We refer to the tail as the probability mass at the upper end point of the distribution. Figure 1.2 shows these shapes for different distributions and values of the EVI. If the EVI is 0, the shape of the tail will be exponential, meaning that the probability of greater values decays as an exponential function. If the EVI is negative, the probability of the extremes will decay faster than the exponential. This kind of tail is named a *light tail*. Otherwise, positive values of the EVI will make the probability decay slower than the exponential, which makes for a *heavy tail*. The methods to estimate the EVI used in this thesis will be explained in Chapter 2.

1.5 EVT for pWCET Analysis

The integration of EVT in MBPTA has received increasing attention in the last decade [44, 82]. The research community working on MBPTA has been debating on which kind of tails are more appropriate for pWCET. It has been shown by argument and empirically that exponential tails are a safe upper-bound in general [3, 149]. Nonetheless, exponential tails can be too pessimistic to be usable, especially for very high quantiles. The pessimism is in part due to the very nature of the distribution. The exponential tail assumes that the execution time may grow to infinity. However, in CRTES this not the case. In the design process of CRTES, each task has an assigned deadline. Therefore, the estimation of a WCET is of utmost importance to ensure the task is executed within its time budget. Then, the theoretically best bound would be one that takes into account the finiteness

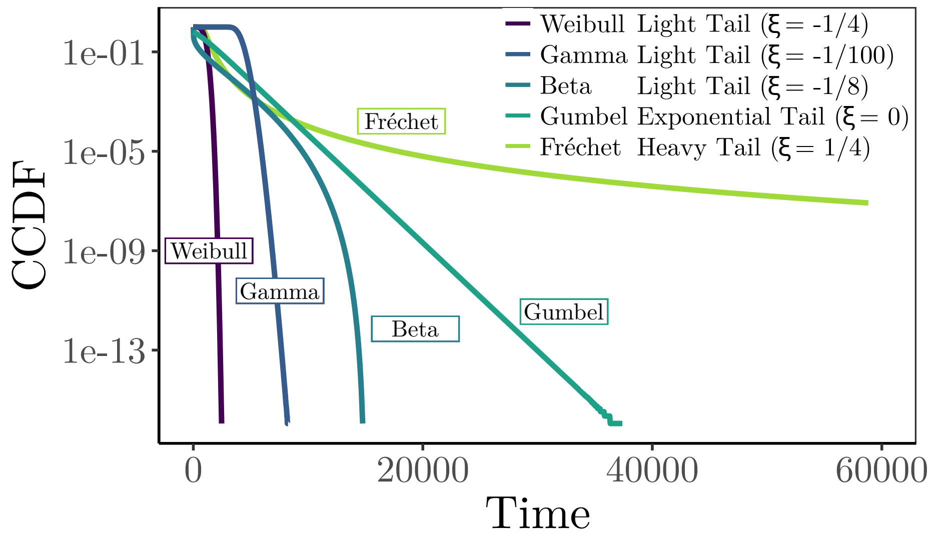


Figure 1.2: *Shape of distribution tails with different EVI ξ.*

of execution times in CRTES programs. Light tails in EVT fulfill this purpose as their domain is bounded with a maximum value and decay faster than the exponential. This is however, both a blessing and a curse. When using a light tail model to fit our execution time samples, it will take the maximum of those samples as the reference for the bounding. If in our samples we do not get an execution time close to the WCET, the light tail model will underestimate eventually. In reality, while light tails models tend to be very accurate in practice, in order to overestimate the WCET, they need to be fed with values in the sample that are very close to the real WCET, which is the very same problem we are trying to solve. In summary, while classical EVT and its two fundamental theorems offer the best theoretical solutions to the pWCET problem, neither the GEV or the GPD cannot prevent the uncertainty in estimating sky-high quantiles which are outside of the observed sample, thus in general estimating a tight and trustworthy upper-bound for the WCET with EVT is challenging. In this thesis, we identify some of these limitations and implement solutions that combine EVT with other statistical tools, and also find new tools outside EVT to estimate extreme values.

1.6 MBPTA with Limited Data

Frequently, CRTES engineers cannot afford an exhaustive set of benchmarks or test vectors to stress the system towards the WCET scenario. That is the case of timing analysis for Autonomous Driving (AD) systems, where the demonstrations via driving test serve as the benchmarks for providing guarantees over the system behavior at operation. However, designing benchmarks that are representative of real-life driving cases is a very challenging task [80]. As mentioned before, the difficulty in assessing WCET scenarios is the complexity of the hardware and software. In the case of AD the software complexity is at the core of its functionality, with Machine Learning algorithms deciphering what the perceptive modules receive in real-time. In these systems, there can be different tasks accessing concurrently the same hardware resources on the system, which causes timing interference or contention impact. In this thesis, we take another point of view in the TA of these systems. Instead of trying to provide with a representative set of benchmarks, we model the intrinsic dependency that occurs within these systems. We aim at modelling the time dilation that occurs when similar tasks are running at the same time. A model of this kind can give us an estimation of a WCET scenario that may occur in very stressful situations with extreme competition among tasks. Ultimately, this is expected to lessen the burden of the benchmark selection for the V&V.

1.7 Thesis Contributions

The contributions of this thesis are towards the design of new and improved methods for the timing analysis of software running on complex CRTES. It is a great challenge to improve on the classical fundamental theorems of EVT because they provide with the theoretical best solution for predicting extreme values. However, MBPTA's characterizing trait is the upper-bounding constraint, which is difficult to achieve both in theory and practice. Also, in CRTES there is a lot of data coming from the monitoring support in the hardware, which would allow creating more sophisticated MB(P)TA techniques accounting for varying contention scenarios. Such data is typically not used owing to inherent hardware limitations, constraining to sampling only a restricted number of variables at once. We provide solutions that allow merging monitoring data from multiple separate experiments as if they were extracted from the same execution conditions. Last but not least, apart from (p)WCET estimation for system design, there is a need for validation (testing) campaigns to assess that the system is properly designed and WCET bounds are not exceeded. However, the lack of controllability of how tasks overlap in multicores precludes from having sufficiently exhaustive test campaigns.

In summary, we can divide the contributions in three blocks: i) improving the current MBPTA methods in tightness and safety for the pWCET estimation problem, ii) creating new methods to analyze the monitoring data of CRTES by putting it in an orderly way that can be used by MBPTA tools, and iii) devising new methods to produce execution time observations for tasks overlapping in arbitrary (and worst-case) ways that test (validation) campaigns fail to produce. More specifically, the contributions are divided in three parts:

1. Sky-high quantile estimation techniques.
2. Hardware Event Monitor data merging techniques.
3. Contention modelling techniques.

1.7.1 Sky-high Quantile Estimation

The estimation of very rare occurrences for the WCET is still an ongoing problem. Those extreme values, a.k.a. sky-high quantiles, are typically in the exceedance probability range of $p = \{1-10^{-6}, 1-10^{-12}\}$. In order to fulfill MBPTA requirements, the pWCET estimation should seek to upper-bound which is an added difficulty on the estimation of extreme values. In that regard, in this thesis we make two contributions towards improving the WCET estimation problem using measurement-based techniques which aim at: i) Improving the accuracy of the estimation of sky-high quantiles, and ii) keep providing with the same safe guarantees of the estimation while needing less requirements.

1. On our first contribution towards sky-high quantile estimation, we develop a new parametric model with analytical expression which can produce safe and accurate estimates while overcoming some limitations of the usual EVT approaches. This technique was devised by making use of both EVT and Risk Analysis, which helps us to make safe guarantees. The proposed parametric model for tail estimation is the tail of a Weibull distribution, or as we named it *tailW*. The solution is tested against real case-study hardware data from a CRTES and improves on the current best EVT solution. This methodology is publicly available in an R package on the official repository (CRAN).

2. Our second contribution towards the pWCET estimaion problem is making use of Markov's Inequality to upper-bound sky-high quantiles. Unlike EVT, and even *tailW*, Markov's Inequality does not try to fit the data but rather upper-bounds data using simple probabilistic tools. We modify the simplest version of Markov's Inequality with the power-of-k function, which provides with much tighter upper-bounds. This works theoretically because one needs information about the distribution of the sample at analysis. The challenge of this problem is in working out a practical solution that allows to take an unknown distribution and still being able to provide trustworthy and tight bounds. We validate this method against reference theoretical distributions and real case-study hardware data from a CRTES, which improves on the current best EVT solution.

1.7.2 Hardware Event Monitor Data Merging

Multicore processors do not allow to read all the HEMs at once. HEMs which provide information about the system at operation, can be read in small groups. Multicore processors have only few (4-6) PMCs while the number of HEMs is in the order of hundreds [67, 68, 69]. Therefore, not much information from the hardware can be extracted at once at operation under the same execution conditions. To make things worse, conditions greatly vary from run to run, making the comparison of runs a relevant concern. This presents a challenge for the use of HEMS in MBTA since, while information from the system at operation can be obtained, only a small subset of the HEMs can be measured at the same time. In this thesis we explore methods to capture as much data as possible that can be used to the advantage of more robust V&V techniques.

- The first contribution makes use of statistics of order to merge HEMs. We do so by selecting a common HEM that will be present in all collections of HEMs to read on a given run. With the use of this common HEM, called the *anchor* HEM, we can proceed to perform a merge based on the order statistics of the *anchor*. We evaluate if the HEMs keep the relationship they have with one another after the merge.
- The second contribution makes use of Copula Theory in order to preserve the relationship between HEMs after the merge. This methodology uses a more data expensive procedure to merge HEMs because one needs to measure the relationship between all HEMs before reading. Then, modelling said relationships with a Multivariate Gaussian copula, we can perform the merge and obtain a final dataset were all values are sorted on similar execution conditions.

In both methods we use the T2080 platform for validation purposes.

1.7.3 Contention Modelling

High-performance processors used in CRTES provide the required computing capacity, but this comes at the cost of additional challenges to the timing analysis of the systems. The fact that multiple cores access shared resources at the same time can produce delay in the execution time of tasks. Therefore, when multiple tasks are executed at the same time, they tend to create contention on each other. In this final contribution, we design a model for inter-task contention that occurs in multicore systems. At system operation we have the timeline of the execution times of different tasks, although they are rarely executed in isolation from other competing tasks. In this work, we show how the contention model we constructed allows us to estimate isolated (solo) execution time and WCET scenarios in systems were this information cannot be obtained. Here we work on a mathematical model of contention to estimate the general impact of contending tasks in CRTES. This model allows us to estimate less pessimistic WCET conditions when working with limited information. We validated this model using execution times from a demo of the Apollo autonomous driving software.

1.8 Structure of the Thesis

- In Chapter 2 we introduce some background to understand each of the contributions made on this thesis. First, a deeper understanding of the limitations of EVT for the pWCET estimation problem based on theoretical arguments is presented. Then, we explain in detail the monitoring data coming from CRTES that cannot be extracted at once. Finally, we explain in detail the contention suffered in MPSoCs and the need to model it for the pWCET problem.
- Chapter 3 shows all the practical and theoretical tools that were used in this thesis which are divided in several blocks. The first block is the statistical one. The final one shows the benchmarks used to validate our solutions.
- Chapter 4 is devoted to the first solution to the WCET problem, the usage of Weibull tails to estimate sky-high quantiles. In this chapter we combine EVT with Risk analysis to provide a tight yet trustworthy solution to the WCET problem.
- In Chapter 5 another solution for the WCET problem is shown. We explain how Markov's Inequality, which upper-bounds by construction, can be the foundation for tight and trustworthy

pWCET estimation.

- Chapter 6 is the first one devoted to the HEM data problem where we want to merge different HEM readings into a single dataset using statistics of order.
- Chapter 7 is the second one devoted to the HEM data problem. The solution provided in Chapter 6 is a simple yet satisfactory one that is easy to implement and execute. Here we use Copula Theory to model the relationship between all HEMs and provide a final merge of the data that keeps their intrinsic relationships.
- Chapter 8 shows the contention model we devised that allows us to estimate isolated execution time and WCET in systems were this information cannot be obtained.
- Finally, in Chapter 9 we show the conclusions of this thesis, its overall contribution, and future lines of work.

1.9 List of Publications

Sky-High Quantile Estimation

1. **Software Timing Analysis for Complex Hardware with Survivability and Risk Analysis.** S. Vilardell, I. Serra, J. Abella, J. Del Castillo and F. J. Cazorla, In *International Conference on Computer Design (ICCD)*, 2019, pp. 227-236, doi: 10.1109/ICCD46524.2019.00036.
2. **Using Markov's Inequality with Power-of-k Function for Probabilistic WCET Estimation.** S. Vilardell, I. Serra, E. Mezzetti, J. Abella, J. Del Castillo and F. J. Cazorla. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2022, pp. 20:1–20:24, doi: 10.4230/LIPIcs.ECRTS.2022.20, **Best Paper Award**.

Hardware Event Monitor Data Merging

3. **HRM: Merging Hardware Event Monitors for Improved Timing Analysis of Complex MPSoCs.** S. Vilardell, I. Serra, R. Santalla, E. Mezzetti, J. Abella and F. J. Cazorla. In *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020, vol. 39, no. 11, pp. 3662-3673, doi: 10.1109/TCAD.2020.3013051.
4. **MUCH: exploiting pairwise hardware event monitor correlations for improved timing analysis of complex MPSoCs.** S. Vilardell, I. Serra, E. Mezzetti, J. Abella, and F. J. Cazorla. In *Symposium on Applied Computing (SAC)*, 2021, ACM, New York, NY, USA, 511–520. doi: 10.1145/3412841.3441931.

Contention Modelling

5. **CleanET: enabling timing validation for complex automotive systems.** S. Vilardell, I. Serra, H. Tabani, J. Abella, J. Del Castillo, and F. J. Cazorla. In *Symposium on Applied Computing (SAC)*, 2020, ACM, New York, NY, USA, 554–563. doi: 10.1145/3341105.3373871.

Chapter 2

Background

2.1 Measurement-Based Probabilistic Timing Analysis

In order to face the challenges of performing Timing Analysis on complex CRTES, lately, MBPTA techniques have been adopted by the community. Their utility on certification has been assessed [153] and even in avionics real-case studies [170].

MBPTA techniques build on collected measurements of the task under analysis and produce a probability distribution function which upper-bounds the execution time of the task under analysis, also called a pWCET. Usually, the pWCET is shown in terms of the Cumulative Distribution Function (CDF), which is the probability that a random variable (r.v.) is smaller than a certain value. Alternatively, and the one we will use throughout this thesis, is the Complementary Cumulative Distribution Function (CCDF) (or $1 - \text{CDF}$), which represents the probability of the r.v. exceeding a certain value, that is, the exceedance probability p. We say that the CCDF of an execution time pWCET, C_{bound}, see Figure 2.1, is said to upper-bound the real CCDF of the task under analysis, C_{real}, – so being a pWCET bound – when for any exceedance probability p the execution time of the former, $et_{bound}(p)$, is higher (or equal) than that of the latter, $et_{real}(p)$. It also holds that, for any given execution time, et, the exceedance probability of the former, $p_{bound}(et)$, is higher (or equal) than that of the latter, $p_{real}(et)$. This can be expressed as $tightness(p) = \frac{et_{bound}(p)}{et_{real}(p)}$.

Ideally, the tightness in pWCET should be as close to 1 as possible, specifically for small p, i.e. large execution times. However, one should aim at ensuring that the tightness is never under 1. Estimations of the WCET which are lower than the real WCET are said to be *optimistic*; and carry risk because one could allocate a smaller time budget to a critical task than it may need. On the other hand, a pWCET cannot be exceedingly pessimistic. A pWCET which estimates execution time 50% higher than the real ones causes a wasteful allocation of computational resources. The upper-bounding constraint is a difficult one to overcome. Typically, predictive models and fitting techniques aim at minimizing the error, or maximizing the likelihood, between the data and the model; however, usually there is not a concern about approaching the data from below or above. Furthermore, the biggest challenge is in upper-bounding data that has not been seen yet. EVT helps a great deal in that regard given that it provides with limit laws (i.e. parametric models for the most rare occurrences) on the range of probabilities that MBPTA is interested in, typically $p = \{10^{-6}, 10^{-12}\}$.

The values of a distribution given a probability, the quantiles, can be obtained with different methods. In general, from a sample of size n one can obtain with relative ease a satisfactory estimation of quantiles with probability $p > 1/n$ [151]. One can easily see the problem if we need to make estimations on very low probabilities. In order to compute a trustworthy estimation of quantiles with probability $p = 10^{-12}$, one cannot simply run a task $n = 10^{12}$ times. In practice, the range of probabilities of interest for V&V purposes for WCET estimation, will in general be beyond the observed sample. That is why initially MBPTA adopted EVT to assess this problem.

As one can see the problem that MBPTA proposes to solve is not an easy one. Even though modelling techniques like EVT have been proposed to estimate very rare occurrences not seen in the data, the requirements of MBPTA are challenging. The timing validation on CRTES are very strict and demand precision and more importantly, *trustworthiness*, in the pWCET models.

2.1.1 MBPTA Requirements

MBPTA builds on a sample of the execution time of the task under analysis. This sample should be independent and identically distributed (i.i.d) so that EVT can be applied. MBPTA-compliant architectures satisfy these requirements by construction [99]. In order to have i.i.d. input data in general one would reset the system each time one run finishes so that each run can be performed under the same initial state. Alternatively, one can choose randomly within an appropriate pool of initial states. Although in practice producing truly independent measurements in CRTES is hard, due to the impossibility of fully controlling the execution conditions. In order to apply EVT the input data must be independent on the extremes, although if this dependence is weak, or the dependence is in the body of the distribution, EVT can still be applied [40].

The i.i.d. requirement is not the only important condition. The last requirement, which is general to all modelling approaches, is the representativeness. One should ensure that the input data that is being fed to the MBPTA model, is as close as possible to the execution time at system operation. In the CRTES domain we have certain advantage with respect to other fields like climate or economics where EVT is applied, in that we are able to generate samples from the system that will be on operation. Nonetheless, it is in the hands of the engineer to produce representative scenarios of the system at operation. In that regard it is not enough simply to run the program many times in order to generate a representative sample. The Operating System is the one in charge of allocating the code, as well as the stack and the heap, into the memory. The way the Operating System performs the allocation will affect timing behavior. The different timings of multiple tasks accessing the same shared resources (shared buses, data buffers, and data caches) can change greatly the overall performance. Thus, it becomes necessary to produce tests with as much coverage of all possible execution paths as possible [186]. In that regard, this representativeness can be achieved by means of randomizing memory-placements at the code, stack, and heap level [70].

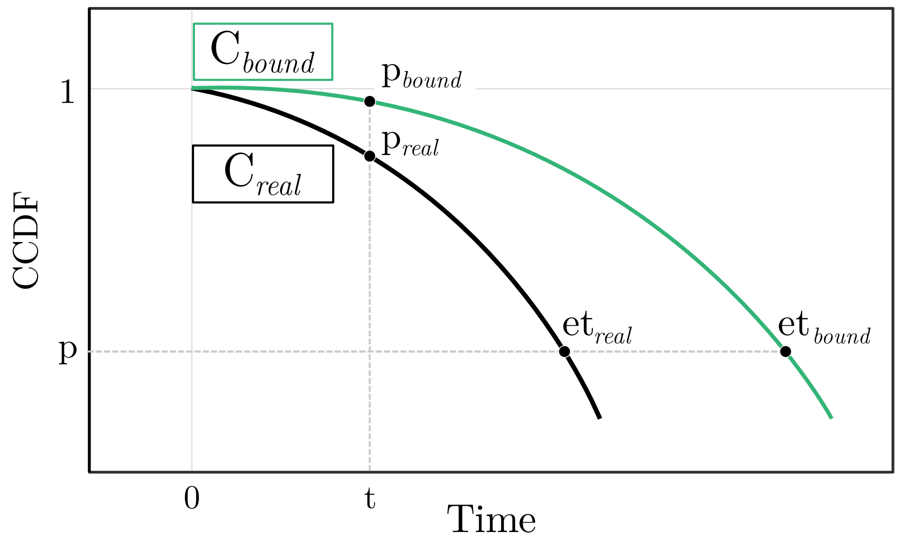


Figure 2.1: *Generic representation of the tightness of pWCET estimations using the CCDF.*

2.1.2 State-of-the-art on MBPTA

EVT relies on i.i.d. observations in the input sample [65, 74, 82]. The difficulty of achieving this depends on the underlying hardware platform [85, 99, 109, 142] and the software support used [104]. For instance, the commercial off-the-shelf hardware platforms considered in [109, 142] pose difficulties to enforce i.i.d. measurements, and some dependencies across measurements may exist, thus requiring special consideration, as shown by Santinelli et al. [142].

The source of dependencies has been investigated by Melani et al. [114] concluding that pWCET estimation is still possible despite dependencies. Other authors build upon alternative methods for measurement collection to get rid of dependencies. In particular, Yue et al. [109] consider retaining only maxima for that purpose. Lima and Bate [104] show that mitigating the impact of discrete data may also help to mitigate the impact of dependencies.

MBPTA brings several challenges [71] including a sound tail modelling application, the representativeness of the input samples [1, 76, 78, 99, 142], and how to interpret the obtained results. These three challenges have been presented in [71], while [35] makes a deep survey of the existing works addressing each of these challenges. In this context, [6, 98] have shown how to interpret EVT results from a certification point of view.

A different challenge to pWCET reliability relates to representativeness of the observations [4, 12, 78, 118] which corresponds, in the timing domain, to guaranteeing the observations are actually representative or upper-bound the execution time distribution, which in fact, is an inherent trait for any sampled-based approach. Statistical and model uncertainties in the scope of EVT are discussed in [26] where robustness in tail estimation is achieved by considering, together with GEV, a family of *plausible* probability models (i.e., close to GEV) and selecting the most conservative estimated probability value among all models.

2.2 Extreme Value Theory

EVT has been one of the most important and useful theories of modern mathematics. There are many areas where there is a need for modelling extreme events that have not been observed yet. CRTES is one of them; not only in order to comply with safety standards, but also for efficiency and energy consumption reasons [157].

In this section we show the two classical fundamental theorems of EVT. Also, we will dive into the techniques to put the theory into practice.

2.2.1 Law of Large Numbers: the Central Limit Theorem

Let us start with a basic definition that will be useful to discuss for this thesis. Throughout this thesis we model the execution time of complex CRTES as a r.v.. In this thesis, a r.v. is devised as a mathematical object in which we attribute a stochastic behavior, responsible of generating *outcomes*, from a set of possible events. Formally speaking, the outcome of a r.v. will be the result of a measure of some event, ω, in an experiment which belongs to the set of all possible outcomes $\omega \in \Omega$, and this event will take form in the measurable space. In this thesis, the measurable space will be the one where execution times belong, the positive real-values bigger than zero, in our $\mathbb{R}_{>0}$. As a toy example, a six-sided dice is a r.v. responsible for generating one of the possible outcomes from the set $\Omega = \{1, 2, 3, 4, 5, 6\}$.

It is needed to stress out that a r.v. is a *function* responsible of generating the *outcomes*. Considering this definition of a r.v. will makes us thing about them in a more abstract way which will be useful throughout the thesis.

The Central Limit Theorem (CLT) helps to introduce the concept of limit laws. In every discipline, modelling probability distributions is a challenging task. More often than not, the distributions of real data do not take the shape of a defined parametric distribution which eases the modelling. This, though, does not prevent performing satisfactory probabilistic models. One of the most useful tools in statistics is to approximate the empirical distribution of real data to a simpler parametric distribution using a *limiting argument*.

Definition 1 (CDF). *The CDF of a r.v. X is a function, F_X, given by*

$$F_X(x) = P(X \leq x), \tag{2.1}$$

where P denotes the probability that the r.v. X takes a value smaller or equal than some x.

Definition 2 (Limit law [40]). *A sequence of r.v.s $X_1, X_2, \cdots$, having distribution functions $F_1, F_2, \cdots$, is said to converge in probability to the r.v., X, having distribution function F, denoted $X_n \xrightarrow{\mathcal{L}} X$, if*

$$\forall x, \quad F_n(x) \to F(x) \quad as \quad n \to \infty. \tag{2.2}$$

Whatever the form of $X_1, X_2, \cdots$, might be, there can be transformations which takes those distributions to a limiting law which is much easier to assess for our modelling. Note that, limiting laws consider that the number of r.v. tends to infinity ($n \to \infty$). In practice, this is not possible to achieve. Some limiting laws may be able to achieve this limit with a different number of r.v. and this may carry problems in the modelling.

Let us use the most famous example of limiting law as introduction, the CLT.

Definition 3. *Let X be a discrete r.v. with probability function $f_X(x)$, where x are the particular values that X can take. The expected value of X is:*

$$E(X) = \sum_{x \in X} x f_X(x) = \sum_{x \in X} x P(X = x). \tag{2.3}$$

Theorem 1 (CLT [40, 168]). *Let $X_1, X_2, \cdots$, be a sequence of independent and identically distributed r.v.s with finite $E(X) = \mu < \infty$ and positive variance $0 < V(X) = \sigma^2 < \infty$. Then the weighted sum of the r.v.s converges in probability to:*

$$\bar{X}_n = \frac{X_1 + \cdots + X_n}{n} \xrightarrow{\mathcal{L}} \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right) \quad as \quad n \to \infty, \tag{2.4}$$

where $\mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$ is the Gaussian distribution with mean μ and variance σ^2.

This is a powerful and fundamental result of statistics and its simplicity explains why the Gaussian distribution seems to appear everywhere. One must be aware of the details of the theorem. Here the CLT is not stating that we are modelling the unknown distributions of this r.v. as a normal distribution. The result states that the *mean* of several r.v., regardless of their parent distribution, can be modelled as a normal distribution.

Bringing this to our domain, usually execution times of complex tasks in CRTES have unknown distributions. If we were interested in knowing the average execution time in our system, a first approach could be model the average of the distribution of our execution time as a Gaussian distribution. Which is a very powerful approach because we do not need to take into account more information than the one that is presented in the samples. Therefore, limiting laws allow us to avoid making ad hoc models for each specific platform or task. Changing platforms or changing tasks will only change the parameters μ and σ^2 of the Gaussian distribution, but the underlying model will stay the same.

2.2.2 Law of Extremes: Block Maxima

Let us show in this section how we can apply the limit laws into the most extreme values of a distribution. As we explained in Section 1.4, we have two ways of thinking about extremes. Let us start by the first one of considering as extremes the maximum value of a sample.

Block Maxima.

We can define,

$$M_n = \max\{X_1, \cdots, X_n\}, \tag{2.5}$$

where $X_1, \cdots, X_n$ is a sequence of independent and identically distributed r.v. with common distribution F. One can think of this sequence as different sets of runs from the same task in a given system, then M_n is the maximum of each set of runs. It is actually rather easy to describe the limit law for M_n:

$$
\begin{aligned}
P(M_n \leq y) &= P\{X_1 \leq y, \cdots, X_n \leq y\}, && (2.6)\\
&= P\{X_1 \leq y\} \times \cdots \times P\{X_n \leq y\}, && (2.7)\\
&= \{F(y)\}^n. && (2.8)
\end{aligned}
$$

Equation 2.6 of the demonstration is just writing the CDF of M_n in terms of all the components. Then, in Equation 2.7 we use the fact that each r.v. $X_1, \cdots, X_n$ is independent, and thus the CDF of M_n can be written as the product of the CDF of all X_i. Finally, because each X_i has the same CDF, F, the CDF of M_n can be neatly summarized as $\{F(y)\}^n$. As clean as this result is, it is not very useful at first because we do not know F, thus we look for approximations on F^n.

When looking at the behavior of F^n when the sample size tends to infinity, $n \longrightarrow \infty$, one can see that for any $y < y^+$, where y^+ is the upper end-point of Y, the CDF of M_n degenerates to zero.

An appropriate linear normalization of M_n allows to stabilize the scale and location of the distribution instead of degenerating to a point mass on y^+. This is the approach that was used to derive the first fundamental theorem of EVT.

Theorem 2 (Fisher–Tippett–Gnedenko Theorem [65]). *Let us suppose that there exists sequences of real-valued constants $\{a_n > 0\}$ and $\{b_n\}$ such that,*

$$P\{(M_n - b_n)/a_n \leq y\} \rightarrow G(y) \quad as \quad n \longrightarrow \infty, \tag{2.9}$$

where G is a non-degenerate[1] *distribution function, then G belongs to the GEV distribution families,*

$$G(y) = \exp\left\{-\left[1 + \xi\left(\frac{y-\mu}{\sigma}\right)\right]^{\frac{1}{\xi}}\right\}, \tag{2.10}$$

which is defined on $1 + \xi(y - \mu)/\sigma > 0$ where $\mu, \xi \in \mathbb{R}$ and $\sigma > 0$.

This model has three parameters. We call μ the location parameter, σ the scale parameter, and ξ the shape parameter. Here we can differentiate three cases that produce the distribution families. The distinction among them is given by the values that the extreme value index, ξ, takes. The first two are the cases where $\xi > 0$ which is the *Fréchet* family and the case where $\xi < 0$ which is the *Weibull* family. As seen in Figure 1.2, the Fréchet distribution is a Heavy Tail distribution, and the Weibull is a Light Tail one. The final case is when $\xi \longrightarrow 0$ which delivers the *Gumbel* family,

$$G(y) = \exp\left\{-\exp\left[-\left(\frac{y-\mu}{\sigma}\right)\right]\right\}, \tag{2.11}$$

which is an Exponential Tail.

Finally we have a solution for the distribution of maxima M_n. Nonetheless, one should recall that this is a limit law. When the number of samples n tends to infinity this is the distribution of the maxima. This puts a burden on the estimation of the parameters of the GEV, more importantly on the shape ξ which is crucial in making a good assessment of the distribution. Being a limit law implies that there will be some uncertainty in the model regardless of the method used to estimate the parameters because our sample size cannot be infinite. We regard this phenomenon as *model uncertainty*.

2.2.3 Threshold Law: Peaks Over Threshold

The second classical fundamental theorem of EVT parts from the second way to look at extremes. Instead of fixing the maxima of each sample, one can consider the probability mass of the upper end-point from a distribution, namely the *tail*.

[1]A Degenerate distribution function is one such that all the mass is located at one point m_0 with probability 1 and the rest of the values are 0.

Peaks Over Threshold (PoT).

Given a r.v. X with a CDF, F, and a threshold, $u > 0$, such that $y = x - u$, the excess r.v., X_u, defined as $(X - u \mid X > u)$ is given by the CDF, F_u, defined as:

$$F_u(y) = P(X - u \leq y | X > u) = \frac{F(u + y) - F(u)}{1 - F(u)}, \quad y \geq 0. \tag{2.12}$$

Then F_u is the excess distribution function, which is the distribution of the excess value over a threshold u. A PoT model is a semi-parametric model where the law of X for $x < u$ is described by the empirical distribution, and for $x > u$ is defined by a parametric model for the tail. It is semi-parametric because, if we want to apply EVT models, we are concerned with the tail of the distribution. Therefore, for the values of the r.v. higher than the threshold $X > u$ we use a parametric model $S_u(x_u)$, with $x_u = x - u$, and for the cumulative probability up to the threshold u we use the empirical probability mass given by the sample. More formally,

$$P(X > x) = S(x) = S(u)S_u(x_u), \tag{2.13}$$

where $S(x) = 1 - F(x)$ and $S_u(x_u) = 1 - F_u(x_u)$ are the CCDFs.

Theorem 3 (Pickands–Balkema–de Haan theorem [15]). *Let F be a distribution function such that $F(x) < 1$ with F_u being its conditional excess distribution function. Then, F_u converges in probability to the Generalized Pareto Distribution (GPD) for large u. That is, $F_u \xrightarrow{\mathcal{L}} G(y; \sigma, \xi)$ as $u \longrightarrow \infty$, where:*

$$G(y; \sigma, \xi) = \begin{cases} 1 - \left(1 + \xi\left(\frac{y-\mu}{\sigma}\right)\right)^{-\frac{1}{\xi}} & \text{if} \quad \xi \neq 0, \\ 1 - \exp\left(-\frac{y}{\sigma}\right) & \text{if} \quad \xi = 0. \end{cases} \tag{2.14}$$

With $\sigma > 0$, and $y \geq 0$ when $\xi \geq 0$ and $\mu \leq y \leq \mu - \sigma/\xi$ when $\xi < 0$. This result is crucial for tail estimation. If the conditions for the theorem are met, the GPD family of functions results in accurate estimates for the most extreme values on the tail. However, the conditions are met when the threshold $u \longrightarrow \infty$, which brings uncertainty in the implementation of estimates with the GPD since finite values of u need to be used.

2.2.4 State-of-the-art for EVT in MBPTA

Extreme Value Theory, a consolidated approach for modelling and predicting the occurrence of rare events, has emerged as the preferred option for modelling the WCET of software programs in CRTES. EVT has been considered particularly fit for probabilistic modelling of WCET as the latter is normally considered a rare event in the program's timing behavior. EVT is at the foundation of several MBPTA approaches [3, 20, 35, 82, 149, 165], which have been already positively assessed in some industrial use cases [170]. Probabilistic approaches building on sample and population sizes have also been built for overlapping concerns across task scheduling and timing analysis [127], hence being orthogonal to WCET estimation. Several works assess the necessary conditions for a correct application of EVT to the timing problem since an inattentive application of the EVT statistical tools can severely affect both trustworthiness and quality of the derived pWCET bounds [71, 105, 118].

Several studies focus on EVT applicability preconditions on the (timing) observations being i.i.d. r.v.s [35, 44]. EVT has also been shown to be applicable also to stationary data preserving extremal independence [142]. Different statistical tests have been assessed for that purpose in the real-time literature [3, 35, 44, 137]. Also, platform randomization [97] or data sample randomization [104] have been used to meet the i.i.d. requirement. However, even in case such statistical preconditions are met, the reliability of the obtained pWCET bounds is still affected by the choice in the EVT inputs (i.e. selection of samples belonging to the tail) and parameters of the fit distribution. Several methods have been proposed for selecting and assessing the quality of EVT parameters and sample selection and, in turn, their impact on the trustworthiness of the computed bounds [3, 12, 138].

Several authors have considered general EVT distributions, without restricting them to the exponential law [76, 104, 105], whereas others build on the particular characteristics of the problem modelled – finiteness of the execution time of critical real-time programs – to fit exponential tails only [3, 44, 59, 82], which have been shown to provide usable bounds and confidence intervals [149].

Probabilistic and statistical approaches [35, 48] have been increasingly considered as a promising solution to cope with the rise in complexity of hardware and software systems, as determined by unprecedented increasing computing performance requirements [54].

2.3 EVT Alternatives

In this thesis we propose alternative models for estimating very rare occurrences in the context of MBPTA. Our aim is to propose new methodologies that maintain safety guarantees while needing less requirements of application.

2.3.1 Chebyshev's Inequality

Additionally, the other front is to mitigate EVT's model uncertainty. Given that EVT has uncertainty in the modelling process that cannot be avoided by construction due to the threshold estimation, we want to resort to methods which can produce safe upper-bounds which do not have uncertainties embedded into them. In that regard, there is a probabilistic inequality called *Chebyshev's Inequality* which upper-bounds the cumulative probability of exceeding a given value of the r.v. at study. This inequality upper-bounds by construction any positive r.v. which makes for a suitable candidate for starting a pWCET methodology. Let us introduce this probabilistic tool.

Theorem 4 (Chebyshev's Inequality [156])**.** *Let X be a non-negative r.v., $b > 0$, and f a non-negative and increasing function. Chebyshev's inequality states that:*

$$P(X \geq b) \leq \frac{E(f(X))}{f(b)}. \tag{2.15}$$

For WCET estimation, X corresponds to the execution time distribution to be bounded, and b to an execution time for which we want to find its upper-bound probability.

Regarding f, it needs to be defined to realize the general Chebyshev's inequality into a specific upper-bound function. The function f can be any non-negative function so that for a particular domain D, Property 2.16 holds.

$$\forall x \in D, \quad f(x) > 0, \tag{2.16}$$

f must also be an increasing function so that for a given interval I, Property 2.17 holds.

$$a, b \in I \quad | \quad a < b, \quad \Rightarrow f(a) \leq f(b). \tag{2.17}$$

Interestingly, Chebyshev's inequality does not require determining where the actual tail distribution starts but instead works with the entire distribution, hence removing EVT's model uncertainty for tail selection. In fact, Chebyshev's inequality carries no model uncertainty.

Observation 1. *Chebyshev's inequality is a model uncertainty free general model for pWCET estimation.*

Chebyshev's inequality also applies to continuous and discrete distributions regardless of their characteristics (e.g. shape, variance, kurtosis, etc.). Also, it is non-parametric, i.e. it makes no assumption on parameters for the studied distribution.

2.3.2 Markov's Inequality

Markov's inequality is a specific instantiation of Chebyshev's inequality.

Corollary 1 (Markov's Inequality [111])**.** *Let $X > 0$ and let the function f be the identity function $f(X) = X$. Hence, Markov's inequality yields:*

$$P(X \geq b) \leq \frac{E(X)}{b}. \tag{2.18}$$

As for the baseline Chebyshev's inequality, Markov's inequality holds for any real-valued r.v. with a finite expected value and positive value b. Also, it (i) is a trustworthy upper-bound, by construction, of the underlying distribution; (ii) has no model uncertainty; (iii) is non-parametric; and (iv) can be applied to discrete and continuous distributions.

In Chapter 5, we will assess Chebyshev's Inequality viability for an MBPTA methodology.

2.3.3 State-of-the-art for EVT Alternatives for MBPTA

Markov's and Chebyshev's inequalities have been historically applied in a wide variety of fields such as Engineering [160], Big Data sampling [143], and radiative transfer [163]. In the context of real-time computing, moment-based bounds on tail probabilities have also been considered in the scope of probabilistic schedulability analysis [37, 38, 56, 167]. Chernoff bounds [37, 38] and generalizations thereof [167] are exploited to compute the cumulative distribution of the interference caused by higher-priority tasks on a task response time, ultimately delivering a probability for deadline misses. While building on application of Markov's inequality to the timing dimension, these approaches do not address the problem of computing probabilistic bounds to tasks' execution time, which are instead assumed to be available, but offer a scalable alternative to the computational complexity of convolutions. Some works [106, 135] consider the use of Chebyshev's inequality for WCET and/or cache hit and miss rates estimation. However, those works consider Chebyshev's inequality only to estimate the impact of the variance on those metrics, and focus on the analysis of statistical uncertainties. Other works consider using higher moments to improve concentration inequalities similar to Markov [101]. Finally, authors in [185] consider a similar approach to those works for WCET estimation, but discard Chebyshev's inequality altogether given the pessimism expected for high quantiles.

2.4 Hardware Event Monitors in MBTA

Powerful measurement-based timing analysis solutions have been proposed for multicores for critical applications building on hardware and software profiling [24, 57, 144]. In this context, the PMU in MPSoCs offers information relevant for timing analysis [73, 115], enabling V&V and software time budgeting of time-critical applications. For instance, the usage of some shared resources can be budgeted, monitored, and enforced using event quotas building on HEMs reached through the PMU [46, 126, 131, 183]. HEMs are used to monitor when tasks exceed their usage quotas which are suspended. Furthermore, HEM data has been shown to aid sampling representativeness [33]. Indeed, HEM information is already used as a pillar to certify critical avionics systems [134], so PMUs, and the HEMs they allow monitoring, become the basis of industrial-quality multicore interference mitigation and estimation techniques.

However, HEM information is not straightforward to use. Due to the impossibility of controlling execution conditions in complex CRTES. A program run several times and performing the same number of instructions, thus maintaining functionality, will yield different HEM results. In this section we analyze the variability of HEM data within the context of the NXP T2080 Reference Board.

We show that several of the 262 HEMs in the T2080 present significant variation (Section 6.2.1) and follow different distributions (Section 6.2.2). We also dig down into some of the reasons behind the observed variation (Section 6.2.3).

2.4.1 Disproportion Between the Number of HEMs and PMCs

Despite the increase in number and specialization of HEMs, their observability and accessibility in reference CRTES platforms is typically constrained by the availability of a relatively (but consistently) lower number of PMCs. The latter represent, in fact, the most natural way of making HEM information available to the user. The number of PMCs available in modern MPSoCs typically ranges between 4 and 8 per core, which inherently clashes with the number of HEMs an analysis would need to track. In the case of ARM A53/A57/A71 cores, the number of PMCs is 6, which also matches the number of PMCs in NXP cores e500mc/e6500.

Overall, the clear imbalance between the large number of HEMs and available PMCs calls for approaches that reliably merge HEM readings across different experiments. It is also worth mentioning that MPSoCs include an increasing number of complex shared resources. This will naturally result into more HEMs tracked by timing analysis techniques to capture the effect of contention.

Hence, several runs are required to read all the HEMs of interest, which are later 'merged' off-line to analyze the program behavior and reason about contention. For instance, to decide whether some tasks can be scheduled concurrently, we need to budget how much each one is expected to access each shared resource, which requires consistent reads of a large number of HEMs.

To make things worse, several runs of the same experiment in an MPSoC can result in inevitable variations in the timing behavior of the program, though its functional behavior is the same. This is due to the impossibility to control the entire hardware and software initial state in each run. In practical terms, this translates into variability in HEM readings (as high as 59% for processor cycles in our target system for relevant HEMs), with no variability observed in instruction count (as analyzed in Section 6.2.3). The engineer is confronted with a set of values (readings) for each HEM, that need to be merged to allow reasoning about multicore contention. Unfortunately, since HEM values from different runs to be merged can be subject to different (large) noise, it is challenging to merge them consistently so that merged HEM vectors – those where all HEMs of interest are included – resemble the values that would have been obtained if they could have been read all of them simultaneously in the same run.

2.4.2 State-of-the-art on HEM Analysis

In the CRTES domain, several works build on HEMs for the estimation of bounds to software timing. Paulisch et al. [126] create an analysis and runtime monitoring solution for limiting task contention in multicores by tracking and controlling HEM. In the same vein, Diaz et al. [55] build on HEM to produce an ILP-based contention model for an AURIX automotive microcontroller. Likewise, Santinelli et al. [77], build on the HEM of a multicore system to derive probabilistic WCET estimates. Griffin et al. [75] derive a method to select the HEM with highest contribution to software timing and predict execution time under unseen configurations.

More recently, information from HEMs has been exploited as the cornerstone of industrial-quality approaches [134, 162] for providing the necessary evidence for supporting the certification of multicore CRTES, in conformance with the requirements from domain-specific certification authorities [62].

Several works in the mainstream (high-performance) domain reason on the sources of variability in HEM values when executing several times the same piece of software. This covers from the operating system noise [119], application variability [7, 121] and the particular HEM-Reading library, to the complexity of the hardware [184]. For instance, [119] focuses on the cycle count HEM and shows that its variability is often related to the executable layout and operating system issues. Also, at software level, [184] assesses the accuracy of various high-level counter APIs with focus on cycle count and total retired instruction HEMs. In our work, we use no operating system and access directly, with no library, the HEMs (via the PMCs) so they are not subject to software-induced variability. In [121], authors focus on task-parallel programs in high-performance environments with highly-dynamic execution conditions, including dynamic task scheduling, that cause tasks to execute in different orders and in different cores across executions. Authors propose techniques to determine which HEM readings belong to each task and hence, combine them to derive all HEMs for a task. Interestingly, the reading of each group of HEMs is performed once, so authors do not assess the impact of variability in HEM readings due to hardware and software related variability. We, instead, focus in much more predictable environments, as needed for CRTES and consider the variability of HEM readings.

HEM sampling or multiplexing consists in time-sharing the PMCs over a set of HEMs: at each interval boundary, whose duration is a configuration parameter, the PMCs are reprogrammed to read a different set of HEMs. HEM sampling is, for instance, adopted by the Linux kernel's perf event subsystem. The potential inaccuracies introduced by the interpolation made by sampling techniques have been studied elsewhere [103, 180]. Other works rearrange samples derived from HEMs from long executions to improve performance analysis [147].

At hardware level, other works [171] focus on specific HEMs (e.g. retired instructions, branches, loads/stores) and develop low-level hardware hypotheses on the reasons behind some of these HEMs suffering from various forms of under and over count. Authors do several recommendations on the hardware design to reduce the observed variability. Our goal, instead, is managing at software level HEM variability and limitations to read HEMs on existing boards (e.g. NXP T2080).

Incomplete sets of data have been considered with Matrix Completion (MC) methods [83, 136]. There are fundamental differences between HEM merging and MC.

1. MC requires input data be a random matrix, where values in all rows and columns belong to one measure with its own distribution for each value, which does not hold for the problem at hand (e.g. the outcome of a HEM measure is a column, thus with its own distribution).
2. MC aims at producing synthetic data to complete missing data, thus bringing risks due to inferring the distribution of real data to produce new data, which may match some characteristics of real input data but miss others.

In fact, since input data (HEM values) do not match requirement (1), and the fraction of missing values is large, MC populates the data array with values whose mean and standard deviation differs drastically from those for real (observed) data. Overall, MC does not fit the needs of the problem at hand by construction since its prerequisites are not met.

Several solutions have been proposed to master multicore interference in industrial contexts. Some approaches attempt to reduce or fully remove interference across tasks running concurrently in multicores by segregating accesses to different hardware blocks. These approaches have been devised for on-chip and off-chip memories, including banks of shared caches, as well as banks and ranks of DDR memories [94, 107, 110, 128, 155, 182].

Other approaches perform segregation over time, rather than over space, by splitting execution of tasks into memory and computation phases, thus letting schedulers guarantee that memory phases from different tasks do not occur simultaneously [25, 51, 129]. However, segregation over time is not always doable due to the characteristics of the hardware or the application itself (e.g. application semantics cannot be changed due to overwhelming V&V costs). Therefore, if time segregation is not feasible, even if space segregation is used, multicore interference can still occur in hardware resources not visible at software level, such as interconnects, as well as buffers and queues internal to shared caches for instance [161]. In those cases, solutions are needed to master interference in multicores and account for it during timing analysis.

2.5 Timing Validation in Automotive Systems

The need for deriving time budgets for safety-related software components in automotive emanates from safety requirements, as described in ISO 26262 automotive functional safety standard [89]. Safety requirements determine the maximum response time affordable for a given functionality and the fault tolerant time interval, which stands for the time allowed since a fault occurs until an action is taken to recover or to bring the system to a safe state. Both determine the end-to-end time for a given functionality to be performed (including sensing and actuation time). Once discounted the time devoted to interfacing with physical components, the remaining time is the time budget allocated for the software component to complete its execution.

Automotive systems are designed and verified following appropriate practices to achieve safety compliance. For instance, WCET estimation tools may be used, together with appropriate response time analysis methods for task scheduling. Nevertheless, a validation step is needed before deploying the system to detect system faults due to, for instance, the violation of some assumptions caused during integration. Whether an application adheres to its timing constraints during the validation process is normally assessed in hardware-in-the-loop testing environments, where the complete application can be run. Appropriate test equipment is used to collect information on the execution of the application under analysis in general, and each of its tasks in particular, thus with a much higher observability than in the system during operation. However, due to the high coupling between the different tasks, the operating system, and the input/output interface of the application, individual tasks cannot be run in isolation or at externally-controlled time instants. Hence, although start and end times can be obtained for the different jobs of each task (observability), it is not possible to enforce specific release times at will (controllability). Overall, tasks overlap with each other in an intricate manner depending on release times dictated by input data and timers, as well as by their duration.

Execution time measurements, therefore, reflect arbitrary-looking overlaps across tasks, representative for the input data used during tests. Whether other overlaps are possible and whether those can lead to higher execution times is, in general, unknown and hard to control in practice. Hence, other than considering an engineering margin on top of the high watermark execution time to account for scenarios not triggered by the tests used, end users lack tools to use those execution time measurements in a more informed manner for validation purposes. While engineering margins set based on experience have worked for hardware platforms with low execution time variability, the increasing use of Graphics Processing Units (GPUs) and other high-performance hardware to match the performance needs of Autonomous Driving frameworks brings much higher performance variability due to contention in the use of shared resources (e.g. shared caches and main memory bandwidth). Hence, end users lack means to quantify whether unobserved scenarios could lead to timing violations.

One of the focus of this thesis is to aim the timing validation methodologies by providing with an analysis tool for the contention derived from overlapping tasks. Continuing the trend of resorting to measurement-based analysis our contention modelling builds on a state-of-the-art autonomous driving framework.

2.5.1 State-of-the-art on MBPTA with Dependency

The impact of dependencies in WCET estimation has been mostly considered for the particular case where they exist across jobs of a given task. In particular, probabilistic WCET estimation was originally formulated on top of the assumption of i.i.d. execution time observations for a task [59]. Therefore, solutions for pWCET estimation have often built on top of Extreme Value Theory for i.i.d. processes [65].

However, some researchers noted that execution time dependencies may exist across jobs of a given task, thus jeopardizing i.i.d. properties, so methods considering those dependencies would be more convenient [40, 100]. Amongst those works, Bernat et al. [22] pioneered in the area of execution time dependencies and analyzed them at the scope of program components.

Santinelli et al. [142] considered a particular type of dependencies across jobs – stationary processes – and concluded that they could lead to WCET underestimation if not accounted for carefully. Similarly, Melani et al. [114] showed that appropriate statistical tests (mostly correlation and independence tests) can be used to account for those dependencies satisfactorily. Lima and Bate [104] proposed a solution to mitigate the impact of dependencies across jobs to facilitate WCET estimation. Finally, Abella et al. [2] have recently shown that the source of those dependencies across jobs imposes different constraints on task scheduling.

Chapter 3

Experimental Methodology

The workflow used in this thesis is carefully designed in order to ensure the results obtained can be trusted; from the generation of the data until the end goal. The diagram in Figure 3.1 shows the general workflow of all experimental methodologies followed in this thesis. In Section 3.1 we present the *Data Generation* process which in this thesis has been performed in two ways.

- The first option is to design a set of programs that stresses the hardware (board) or simulator used to obtain execution times or HEM data for the analysis.
- The second option, described in Section 3.2, is to use analytical parametric distributions as benchmarks for pWCET methodologies' validation. Gathering a set of parametric distributions with similar tail profile than the most extreme conditions of CRTES at operation is useful to validate pWCET estimation methodologies.

Once we have chosen the benchmarks for data generation, we perform the *Data Gathering* using sampling techniques. In Section 3.3 the *Data Validation* step is detailed. After sampling, the samples coming from real hardware or simulator should pass a validation process with an i.i.d. test to ensure that the methodologies can be trustingly applied. In contrast, the data sampled from parametric distributions is obtained using standard statistical libraries from R [133], in which the i.i.d. condition is given by construction. Finally, Section 3.4 shows the *Modelling* part of the experimentation. Once the data is ready and validated, we apply the models and methods we propose in this thesis and also state-of-the-art methodologies like EVT. The application of EVT models will be further explained in this chapter. After the confidence intervals have been computed, we obtain our *End Result*, be it a pWCET or other results from our methodologies. Even if the methodologies we investigated in this thesis are different in nature, the core of the experimental methodology remains the same to ensure trustworthy results and methods.

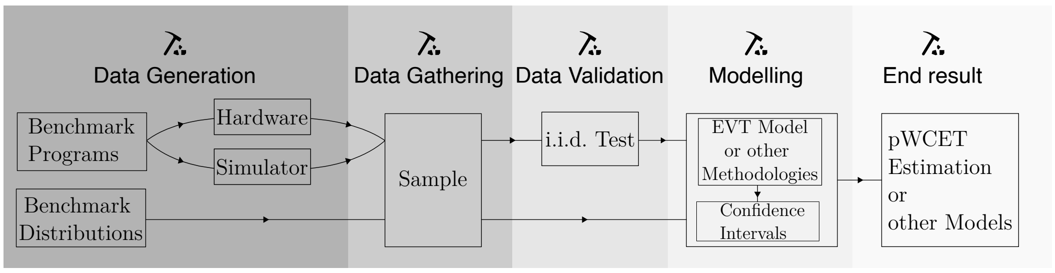


Figure 3.1: *Diagram of the experimental workflow with the tools used in this thesis.*

3.1 Benchmarks and Platforms

In this section we describe the benchmarks used in this thesis to generate the data that will be used to validate and test our methodologies. We distinguish between a real-case study on real hardware,

Table 3.1: *Workloads on the T2080 for validation purposes.*

	Core0	Core1	Core2	Core3
W1	IMUL_UL2	FADD_MEM	FMUL_MEM	LMUL_MEM
W2	LADD_DL1	LMUL_UL2	FADD_UL2	FMUL_UL2
W3	LMUL_MEM	LMUL_UL2	FADD_UL2	DMUL_UL2
W4	LADD_UL2	FADD_MEM	FMUL_MEM	LADD_MEM
W5	FADD_MEM	FMUL_MEM	LADD_MEM	LMUL_MEM
W6	FADD_UL2	FMUL_UL2	LADD_UL2	LDIV_UL2
W7	FADD_UL1	FMUL_UL1	LADD_UL1	LMUL_MEM
W8	FMUL_UL1	FADD_MEM	LMUL_L1	LMUL_MEM
W9	FMUL_MEM	FADD_DL1	FADD_MEM	LMUL_DL1
W10	LADD_MEM	LADD_DL1	LADD_UL2	LADD_MEM
W11	FADD_DL1	FADD_UL2	FADD_MEM	FMUL_UL1
W12	FADD_UL2	LADD_UL2	LDIV_DL1	LMUL_UL2
W13	LADD_UL2	FADD_MEM	FMUL_MEM	LADD_MEM
W14	LADD_UL2	LMUL_UL2	FADD_MEM	FMUL_MEM
W15	FADD_MEM	FMUL_MEM	LADD_DL1	LDIV_DL1
W16	LADD_DL1	LDIV_DL1	FADD_MEM	FMUL_MEM

benchmarks that are run on real hardware boards to collect HEM data, a representative autonomous driving software framework, and a set of reference analytical distributions which are representative of tail profiles in CRTES.

3.1.1 Railway Case-Study on a LEON3+ Platform

The European Train Control System (ETCS) is a safety-critical application (SIL 4) responsible of signaling and control in the European Rail Traffic Management System (ERTMS) framework.

ETCS protects the train motion by constantly monitoring traveled distance and speed, and is programmed to activate the emergency brake system whenever unauthorized speed values are detected. The ETCS subsystem comprises three main tasks that are executed sequentially to provide the required safety function: the Odometry module, estimating a set of parameters based on the inputs collected from the train environment (e.g., estimated position); the Service module, managing the Service braking system; and the Emergency module, actually controlling the Emergency braking system. This function has strict real-time requirements and needs to be certified at the highest integrity level in the corresponding railway safety standards. While all three tasks do exhibit strict real-time requirements, we focus our evaluation on the Emergency module, the core of the ETCS CRTES module.

The ETCS validation suite, which is made available with the application, includes 10 different input vectors (TEST0 to TEST9) corresponding to the operating conditions for functional and timing validation regarded as relevant by the application owner. These benchmarks are used for the techniques proposed in Chapters 4 and 5.

Platform. The platform for this experiment is an Field-Programmable Gate Array (FPGA) implementation of a LEON3+ architecture comprising first level instruction and data caches, and a unified L2 cache, where the sources of execution time variation have been conveniently controlled by hardware means to guarantee the representativeness of the measurements collected at analysis with respect to the timing behavior during operation [85, 159]. In particular, this processor builds upon the concepts of time upper-bounding and time randomization to enforce representativeness.

3.1.2 Microbenchmarks on an NXP T2080 Platform

In general, programs can have built-in sources of non-determinism (e.g. time- or input-dependent values). Also, they may easily be subject to variability due to minimal variations in the Operating System [171]. In order to reduce these sources of variability, we construct specific test-cases, which

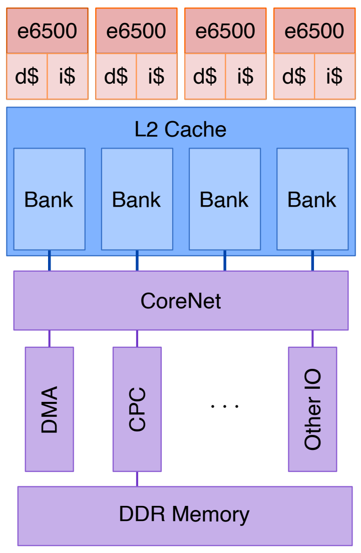


Figure 3.2: *Block diagram of the T2080.*

also aim at triggering a wide set of HEMs. To that end, we have created different benchmarks comprising different core and cache (memory) patterns. At core level, we create 3 benchmarks using intensively the integer pipeline, using integer (I) and long (L) operands, and the floating point (F) pipeline. We use short latency addition (ADD) operations and long-latency multiplication (MUL) operations. At the cache hierarchy level, benchmarks operate on a vector whose size we vary so most load/store operations hit in the data cache (DL1), the L2 (UL2) or memory (MEM). From these 18 benchmarks $(I, L, F) \cdot (ADD, MUL) \cdot (DL1, UL2, MEM)$, we have generated 16 workloads, as shown in Table 3.1.

These benchmarks are used for the techniques presented in Chapters 6 and 7. In both chapters, we target a NXP T2080 Reference Design Board [68, 69] increasingly considered in the avionics domain, with Airbus already achieving multi-core certification on this board [113], and with Rockwell Collins pursuing such certification [134]. The T2080 equips 4 e6500 cores (see Figure 3.2), each comprising private instruction and data cache as well as a private MMU. A second level cache is shared between all the cores. A "CoreNet" coherence fabric provides access to the memory controller as well as other peripherals present in the board. Some features are deactivated in our setup for predictability reasons, such as SMT (Hyperthreading in Intel terminology) and the CoreNet Platform Cache.

We have run our tests in a bare-metal setup, using the software development kit provided by the board manufacturer (NXP) to configure the platform and load images to it through a JTAG debugging interface. In the bare-metal setup, we access PMCs directly without the use of a specific library, e.g. PAPI, to minimize the impact of readings.

In each experiment, we run one benchmark per core. The task in core0 is the reference task on which we perform the analysis, for the tasks in the other cores would be performed analogously. In each run of every experiment, we collect measurements when the task in core0 finishes its execution. We consider single-path benchmarks to isolate platform-level variability, so that in all runs the number of instructions executed (`INSTRUCTIONS_COMPLETED`) in core0 is exactly the same. Across any two runs of an experiment, we reset the state of caches, TLBs, and Branch Target Buffer. To that end, we execute a micro-benchmark that generates a massive number of misses in all those stateful blocks. While ISA-specific solutions exist that allow obtaining the same effect with specific instructions, we considered the micro-benchmark solution to be more platform agnostic.

3.1.3 Apollo Autonomous Driving Framework on an NVIDIA Jetson Platform

In Chapter 8 we present a methodology to assess contention in CRTES. There, we target Apollo [11] AD framework, as the largest existing AD project with more than 120 partners including top-tier AI and tech companies, and car manufacturers.

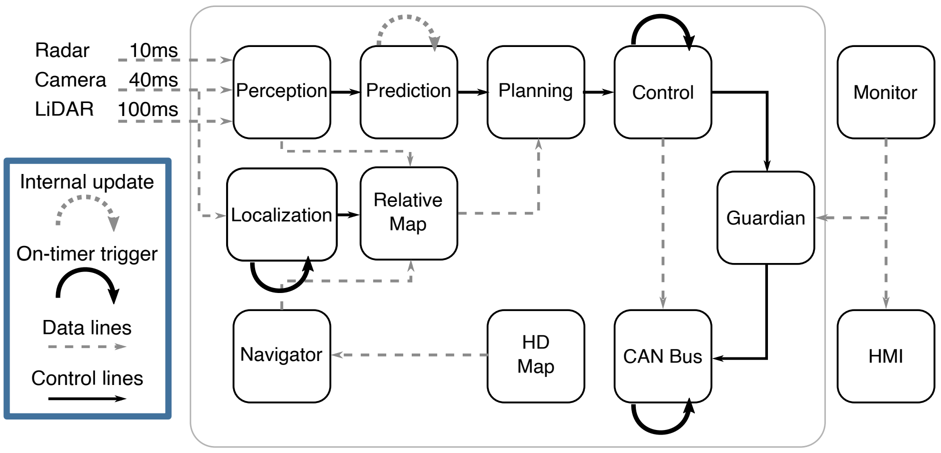


Figure 3.3: *Apollo 3.0 software architecture pipeline.*

AD systems drive the vehicle towards a specified destination as safely and efficiently as possible. To this end, the AD system includes complex combinations of various input sensors such as cameras, short-range and long-range radars and LiDARs to scan the surrounding area around the car and track the moving objects. The system features sophisticated navigation schemes to locate the position of the vehicle with centimeter-level precision. Based on this information, the system plans the future paths, predicts the trajectory of moving objects, and controls the vehicle to follow the specified paths. These are the main stages of almost all practical state of the art AD systems [11, 13, 158]. Figure 3.3 shows the main modules and architecture of our reference AD system, Apollo 3.0. The main modules are the following:

(1) *Perception* module identifies the obstacles in the surrounding of the autonomous vehicle.

(2) *Prediction* anticipates the future motion trajectories of perceived obstacles/objects.

(3) *Localization* leverages information received from different sensors to estimate the location of the vehicle precisely.

(4) *Navigator (Routing)* module tells the vehicle how to reach the specified destination.

(5) *Planning* plans the spatio-temporal trajectory of the vehicle.

(6) *Control* generates control commands such as accelerating, braking and steering based on the outcome of these modules.

(7) *CANBus* passes all the control commands to the vehicle hardware and also provides some information back to the AD system.

(8) *HD Map* module is a library that provides detailed structured information about the roads.

(9) *HMI* (Human Machine Interface) is a module for viewing the status and controlling the functions of the vehicle in real-time.

(10) *Monitor* is a surveillance system to check all the software and hardware modules.

(11) *Guardian* is a safety module responsible to intervene whenever *Monitor* detects a failure.

Inter-Module Functional Dependencies: The dependencies between different modules of Apollo can be triggered in three different ways:

- Periodically, based on a timer (On-Timer).
- Once a module produces data to be served by its successor or successors.
- Whenever a module receives a *request* message from another module. Accordingly, a *response* message should be computed and published.

As shown in Figure 3.3, each module may be triggered by either two or just one of these ways.

- *Perception* receives sensor input data such as Radar data, LiDAR point-cloud data, and camera data. Based on these inputs, this module detects objects of interest and traffic lights and tracks them inside consecutive frames.
- *Localization* module has two modes: a RTK-based[1] mode with a timer-based function (On-Timer), and Multiple Sensor Fusion (MSF).
- *Prediction* receives output data of both *Perception* and *Localization*. It updates its internal status whenever it receives a *Localization* update. However, the main *prediction* function is triggered once *Perception* publishes an output message.
- *Navigator* module is triggered whenever a routing request is received.
- *Planning*, *Control* and *CANBus* modules are all triggered periodically and also when they receive *Prediction* output, human commands and *Control* commands respectively.

In Figure 3.3, control lines, On-Timer triggers, and inter-module updates are shown with different arrow types.

Platform. Since our target are real automotive systems, we have ported it to the NVIDIA's latest automotive platform, Xavier MPSoC, which is integrated in the Jetson AGX Xavier [148] platform. The Xavier MPSoC consists of an Octa-core ARM-based CPU, a 512-core NVIDIA Volta GPU, and several accelerators such as specialized deep learning accelerators (NVDLA) to meet the needs of automotive systems.

3.2 Analytical Distributions

In general, parametric models with analytical expressions are very useful to validate pWCET methodologies. While case-study benchmarks allow us to validate the pWCET with data that will be similar to the one at system operation; parametric models allow to validate the pWCET for exceedingly low probabilities. With parametric models, one can contrast the pWCET at probabilities as low as $p = 10^{-15}$, thus assessing how the model works for quantiles which are very far from the ones observed. This kind of validation cannot be done with real hardware data due to the impossibility of capturing such quantities of data. Therefore, when working in tandem, analytical distributions and case-study data make for a more complete validation set.

In the scope of timing analysis of real-time programs, it is reasonable to assume that the target program will always terminate. Hence, a WCET value upper-bounding all possible program's executions always exists. It therefore follows that, the tail of its execution time distribution is necessarily a light distribution, which has been shown to be trustingly and tightly upper-bounded with light and exponential tail distributions [3, 44, 149]. For this reason, we focus on light and exponential tails.

While heavy tails are, therefore, unnecessarily pessimistic and hence, left out of our set of validation distributions, an execution time sample could apparently correspond to a heavy tail distribution. This could be, for instance, the case when the sample size is not large enough to provide sufficient representativeness in a mixture distribution. In the general case, this concern relates to the sampling process (sample size in particular), shared across all applications of statistics, and therefore, beyond the scope of our methodology. Nevertheless, nothing precludes the use of our methodology for heavy tails.

The different types of tails are illustrated with the CCDF of several example distributions in Figure 1.2. All three GEV distributions have location $\mu = 0$ and scale $\sigma = 1000$: the Weibull distribution (light tail) with shape $\xi = -1/4$ has a sharp slope and a maximum value (2500 in the example); Gumbel (exponential tail) with $\xi = 0$ has also a relatively sharp slope but it has no maximum; and the exceedance probability for the Fréchet distribution (heavy tail) with $\xi = 1/4$ decreases polynomially. The Beta (light tail) distribution has a similar profile to the Weibull distribution and it is commonly used to model high quantiles of random variables with a finite defined interval [10, 92, 116], which

[1]Real-time kinematic (RTK) positioning is a satellite navigation technique used to enhance the precision of position data derived from satellite-based positioning systems (global navigation satellite systems, GNSS).

Table 3.2: *Distributions used for the analysis and their respective parameters.*

Acronym	Type	Parameters	Probability Density Function
Gaussian1 Gaussian2	Gaussian Gaussian	$\mu = 100, \sigma = 10$ $\mu = 100, \sigma = 50$	$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$
Weibull1 Weibull2	Weibull Weibull	$\alpha = 4, \lambda = 80$ $\alpha = 8, \lambda = 80$	$\frac{\alpha}{\lambda}\left(\frac{x}{\lambda}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\lambda}\right)^{\alpha}\right)$
Beta1 Beta2	Beta Beta	$\alpha = 8, \beta = 1/4$ $\alpha = 8, \beta = 1/8$	$x^{\alpha-1}(1-x)^{\beta-1}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$
Gamma1 Gamma2	Gamma Gamma	$\alpha = 100, \lambda = 1$ $\alpha = 150, \lambda = 1$	$\frac{x^{\alpha-1}}{\Gamma(\alpha)\lambda^{\alpha}} \exp\left(-\frac{x}{\lambda}\right)$
Mixture1	Mixture of Gaussians	$\mu = \{5, 50, 100\}$, $\sigma = 10$, $w = \{0.60, 0.39, 0.01\}$	$\sum_{i=1}^{3} \frac{w_i}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$
Mixture2	Mixture of Gaussians	$\mu = \{50, 100, 400\}$, $\sigma = 50$, $w = \{0.60, 0.39, 0.01\}$	-
Mixture3	Mixture of Weibulls	$\lambda = \{5, 50, 100\}$, $\alpha = 4$, $w = \{0.60, 0.39, 0.01\}$	$\sum_{i=1}^{3} w_i \frac{\alpha}{\lambda_i}\left(\frac{x}{\lambda_i}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\lambda_i}\right)^{\alpha}\right)$
Mixture4	Mixture of Weibulls	$\lambda = \{5, 50, 100\}$, $\alpha = 8$, $w = \{0.60, 0.39, 0.01\}$	-

fits WCET modeling. And the Gamma (light tail) distribution is also typically used in EVT and WCET [12].

All former distributions are unimodal, which is a common way to represent execution time profiles. However, it has also been shown that the execution time of many programs presents "clusters". That is, the program's execution time varies around two or more central values rather than vary around a single mean. This results in *mixture distributions* that can arise both in sequential applications and parallel applications [3, 177, 178]. As an example of the former, let us assume a program whose execution time profile is influenced by the latency of its load/store operations which can hit or miss the data L1 cache and L2 cache across different runs depending on the program's inputs. This results in a mixture distribution with 3 clusters (peaks) around the data L1, L2 and memory latencies, respectively. An example of mixture distribution is represented by the black line in Figure 4.1.

Overall, we use a solid set of reference distributions that are in line with the state of the art [12, 34, 82, 137]. In particular, we use unimodal (Gaussian, Weibull, Beta, and Gamma) and challenging multi-modal distributions (Gaussians and Weibulls) with different tail profiles to increase representativeness. In Table 3.2 we show all the reference distributions and their parameters used in this thesis. Summarizing, we fixed this set of reference distributions based on these criteria:

- Working on the assumption of non-heavy tails, we choose four parametric distribution families to reflect on different tail profiles. Thus, Fréchet distribution is left out. Also, Gumbel distribution is replaced by the Gaussian distribution, which has exponential tails.

- Execution time profiles of complex CRTES working with complex software produce mixture distributions. Each component of the mixture is related to the latencies produced by misses in the different cache levels. We choose two families of mixture distribution with exponential (Gaussian) and light tails (Weibull).

- Each distribution family is present with two different parameterizations to stress the starting point of the threshold from an EVT point of view.

3.3 Statistical Tests and Techniques

In thesis, we resort to the frequentist approach to statistical testing [123]. Frequentist inference has proven to be a trustworthy approach to hypothesis testing and to compute confidence levels comparable to the Bayesian approach in critical domains like healthcare [169]. In the literature there are a plethora of tests that tackle these problems and we try to use the most powerful ones from the point of view of frequentist inference. Before discussing statistical testing, let us throw some light to the p-value discussion.

The p-value in hypothesis testing is mistakenly thought as the probability of your hypothesis being correct or not. This interpretation is quite common due to the multiple layers in between your sample and the p-value that are not explored enough. The definition of p-value is the conditional probability to the hypothesis associated to an statistic of the observed sample. In the next section we further explain this definition.

3.3.1 Hypothesis Testing

Hypothesis testing is commonly used to assess some characteristic in our data. If done right, it is very useful to guide us on our assumptions on the data.

First, one starts by fixing a *null* hypothesis, H_0, which is the hypothesis that one wants to test on the data, and then an *alternative* hypothesis, H_1, in order to fix an alternative option. For instance, a null hypothesis can be that the data follows a Gaussian distribution. Then, one decides which test is appropriate. It is important to highlight that the goal of hypothesis testing is gather enough information about the sample to either reject the null hypothesis or not reject it, but the null hypothesis will never be accepted as such for reasons we will explain later. The statistical test works as the following. One can devise a new random variable, t, attempting at describing some property of the data, also called a *test statistic*. To put it formally, the test statistic will be devised in such a way that, *if* the null hypothesis is true, *then* the test statistic t will follow a certain distribution T. Now, let us assume we have our sample and computed our test statistic t, it is time to assess whether t follows the distribution T. This is a complicated problem of its own, but in hypothesis testing what one does is assess whether a value like t is likely to happen *if* it comes from distribution T. In the case of unilateral testing, the conditional probability on this statistic is given by:

$$p = P(T \geq t|H_0), \tag{3.1}$$

which is the definition of the p-value for the unilateral test. Note that, there are some tests where the test statistic, t, is directly related to the null hypothesis. For instance, if we want to test whether our data can be adjusted to a given expected value, we will perform a *t-test* [154], in which under the null hypothesis the statistic follows the Student's t-distribution. In other cases, the test statistic might just be a consequence of the null hypothesis. If our null hypothesis is that our data follows a Gaussian distribution, because it is a symmetric distribution, we can test whether our data has zero skewness. In this case, zero skewness is a property derived from the null hypothesis, but on its own it is not enough proof to assess if we are under the null hypothesis. In practice, one should aim at finding, or devising, a statistical test in which the statistic is directly related to the null hypothesis (e.g. Lilliefors' test for the Gaussian distribution [102]).

Now that we have developed the intuition on the p-value, we need to assess how to interpret the usual critical factor, α, used in practice. Even if the probability of having a value as big or small as the statistic, t, coming from T is very unlikely, they still can happen. As we show, there is a whole branch of statistics dedicated to extreme values. Therefore, the assessment of whether or not t belongs to T is determined by a critical factor α set by the user. In the case of unilateral testing, if $p < \alpha$ the null hypothesis is rejected, which means that the user thinks the value is so unlikely they *assume* it does not come from T. We put emphasis on the assumption because test are not a truth, the user is the one that puts a critical threshold and decides how risky they want their assumptions to be. Furthermore, if one takes another look at Equation 3.3.1, it says that one *first* assumes that the null hypothesis is true, then checks whether or not the data reflects that. Usually one would think we are testing the opposite, that given our data, how likely is the null hypothesis to be true. In practice computing that is very challenging and thus we resort to these incomplete methods. That is why one can not accept a null hypothesis, because the tests are designed assuming the null hypothesis is true. Nonetheless, hypothesis and statistical tests are very useful and are used widely within the scientific community, but it is important that each and every scientist that implements them knows their limitations.

3.3.2 Confidence Intervals

It is common in any statistical estimation dealing with r.v. to provide with confidence intervals. One of the most common approaches is the Delta Method [42], or propagation of uncertainty. Let $\hat{\theta}_n$ be the estimator of a certain parameter θ. If the estimator is asymptotically normal then:

$$\sqrt{n}(\hat{\theta}_n - \theta) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma^2(\theta)), \tag{3.2}$$

the difference between the estimator and the true value of the parameter converges in probability to a Gaussian distribution which is fairly easy to compute. In general, the computation of confidence intervals is not so straight-forward given that the estimator may be inside a more complex function. For instance, if we devise a model for the execution time of a program, rarely will this depend directly on the estimator. Usually, it will be devised in a more complex way inside some function, $ET \sim g(\hat{\theta}_n)$. Let g be a differentiable function around θ with g' being the derivative. Then the Delta Method for the univariate case is

$$\sqrt{n}\Big(g(\hat{\theta}_n) - g(\theta)\Big) \xrightarrow{\mathcal{L}} \mathcal{N}\Big(0, \sigma^2(\theta)[g'(\theta)]^2\Big). \tag{3.3}$$

Suddenly, the computation of confidence intervals can become very complex. The derivative of g may be very hard to compute in practice. In those cases, one can resort to techniques which put the burden on computing power.

The Bootstrap [60] is a technique to estimate the sampling distribution of a r.v. coming from an unknown distribution F. Let us explain this with a simple case. We want to compute the mean from a sample X_n drawn from a population F. With the delta method one would estimate the mean using the sample mean $\bar{X}_n$ and estimate the confidence interval using Equation 3.2. In Bootstrap, one takes the sample $X_n = x_1, x_2, \cdots, x_n$ and resample it *with replacement*, meaning that when resampling any value x_i can be drawn many times. A sample without replacement would mean that once x_i is drawn, it gets taken out of the sampling pool. Now you have another sample X^*_m. In [60] the second Bootstrap method states that as one generates multiple samples X^*_m and computes the desired estimator with it $\hat{\theta}^*_m$, its distribution is approximately the one in the Delta Method. The power of this method is that this is true for any r.v., therefore it works as well for the case of the estimator being in a complex function $g(\hat{\theta}_n)$. In practice, when computing confidence intervals using Bootstrap one would generate many Bootstrap samples and take as confidence intervals the maximum and minimum value of the estimator produced by those samples. The Bootstrap method eases the computation of confidence intervals by a great deal and is used throughout this thesis.

3.3.3 Testing for Independence

The Ljung-Box test [28, 108] is a test to check the independence of a sample. The original idea from [28] was to find the distribution of the residuals in ARIMA models. One can define the autoregression of the residuals of an ARIMA as:

$$\hat{r}^k = \sum \frac{\hat{a}_t \hat{a}_{t-k}}{\hat{a}^2_{t-k}}, \tag{3.4}$$

where $\hat{a}_t$ are the residuals. Then for large n and assuming a's are uncorrelated random deviates, the distribution for the statistic,

$$Q = n(n+2) \sum_{k=1}^{m} \frac{\hat{r}^2_k}{n-k}, \tag{3.5}$$

follows a chi-squared distribution χ^2_m with m degrees of freedom. Now, we can use this reasoning to check the independence of a random sample. We take a random sample and compute the statistic Q, then if the r.v. is independent Q follows a χ^2_m.

3.3.4 Test for Identical Distribution

The Kolmogorov-Smirnov (KS) test [95, 150] is usually the go-to test for identical distribution of a sample. Given a r.v. X of size n one wants to assess whether the sample was drawn from distribution $F(x)$. For the KS test one computes the statistic:

$$D = \sup_x |F(x) - S_n(x)|, \tag{3.6}$$

where $S_N(x)$ is the empirical cumulative distribution function of $\mathbf{X}$. Under the null hypothesis and if $F(x)$ is continuous the statistic $\sqrt{n}D_n$ for $n \to \infty$ follows the Kolmogorov distribution,

$$P(x) = \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} \exp\left(-\frac{(2k-1)^2\pi^2}{(8x^2)}\right), \tag{3.7}$$

which does not depend on $F(x)$. For the usual case where $F(x)$ is unknown we use the *two-sample* test:

$$D_{m,n} = \sup_x |S_{1,m}(x) - S_{2,n}(x)|. \tag{3.8}$$

One rejects the hypothesis of $S_{1,m}$ and $S_{2,n}$ coming from the same distribution if the statistic $D_{n,m}$ surpasses some confidence level α, e.g. $\alpha = 0.05$. Note that, in order for the statistic to follow a Kolmogorov distribution, the parameters of $S_{1,m}$ and $S_{2,n}$ must be known. If one or more parameters must be estimated from the sample, alternative test based on Kolmogorov should be devised, as in the case of the Lilliefors' test for normality [102]

3.4 Estimating the Extreme Value Index

In this section we discuss some methods to estimate the extreme value index while indirectly inferring on the threshold for the tail.

3.4.1 Hill Estimator

Let us assume a sample $X_1, \cdots, X_n$ with CDF F, and let $X^{(1)} \geq X^{(2)} \geq \cdots \geq X^{(n)}$ be the order statistics. That is, $X^{(1)}$ denotes the highest value from the sample and $X^{(n)}$ denotes the lowest one. The Hill estimator [86] based on n upper order statistics is:

$$\hat{\xi}_k = \frac{1}{k} \sum_{i=1}^{k} \frac{X^{(i)}}{X^{(k+1)}}, \tag{3.9}$$

for $k = 1, \cdots, n-1$. The extreme value index has an estimator which only depends on the proportion of upper order statistics with some chosen threshold X^{k+1}. The particular choice of threshold is crucial for the estimation. The estimator is shown to be consistent when $k \to \infty$. As one can see the particular choice of k crucial for the estimation while remaining an unknown variable. The threshold k depends on the underlying distribution F which is unknown and therefore requires solutions which are obtained directly from the data which can provide asymptotic solutions like bootstrap [45]. This can still be computationally demanding given the need to use finite samples. In practice, one infers ξ with the use of the Hill plot. From a given sample, we compute all $\hat{\xi}_k$ from a set of thresholds $X^{(k)}$ such that $1 \leq k \leq n-1$. Then in a plot we draw in the x-axis all threshold values k and in the y-axis all the computed $\hat{\xi}_k$. We will find a region on the plot that is stable, and we will infer the value of ξ from there. However, this method is well-suited for the analysis of heavy-tails, given that the model is derived from the power law model. Given that in the context of this thesis we discard the use of heavy tails, we need to resort on other methods to estimate the extreme value index.

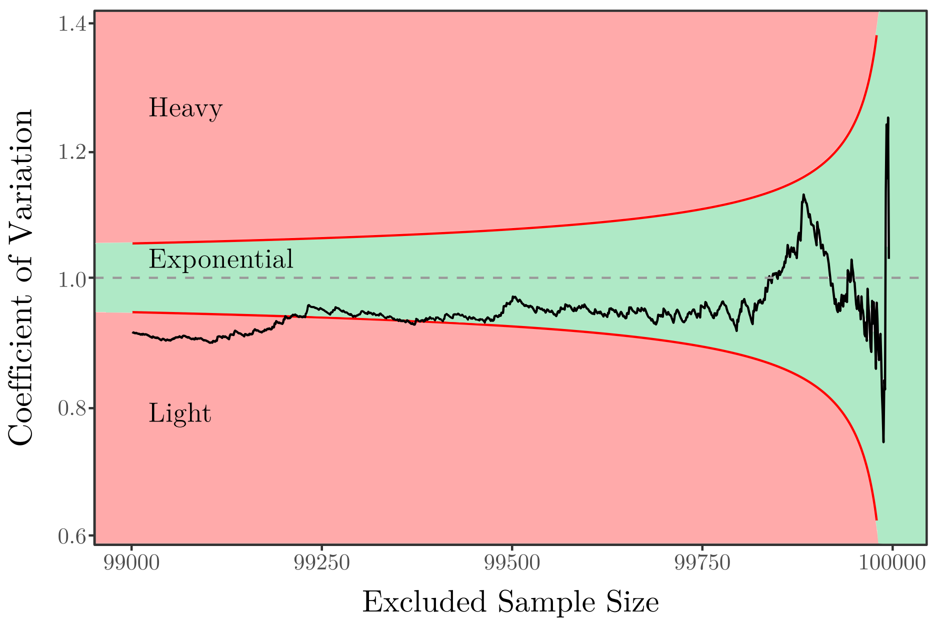


Figure 3.4: *Example of a CV plot for a sample from Gaussian distribution.*

3.4.2 Coefficient of Variation

The Coefficient of Variation (CV) is a powerful test to distinguish between polynomial (light and heavy) and exponential tails [90]. It uses the residual coefficient of variation. The theoretical residual CV from the excess random variable X_u from a threshold u is given by

$$\mathrm{CV}(u) \equiv \mathrm{CV}(X_u) = \sqrt{V(u)}/M(u), \tag{3.10}$$

where $M(u) = E(X_u)$ is the residual mean and $V(u) = \mathrm{Var}(X_u)$ is the residual variance. For a given sample we can compute the empirical residual CV with the standard deviation and mean of the sample. It was shown in [132] and [15] that the residual CV of a random variable, for a sufficiently large threshold, is almost constant and tends to the residual CV of a certain GPD. In the case of the GPD, the residual CV is constant and independent of the scale parameter,

$$C_\xi = \sqrt{1/(1-2\xi)}. \tag{3.11}$$

Therefore estimating the CV one can estimate ξ:

$$\xi = \frac{1}{2}\left(1 - \frac{1}{C_\xi^2}\right). \tag{3.12}$$

This serves as a distinguisher for the kind of tail one is working with. Consider the case when $C_\xi = 1$, then $\xi = 0$, therefore our sample, X, is in the domain of attraction of a Gumbel distribution (exponential tails). Similarly, if $C_\xi > 1$ we are dealing with heavy tails, and if $C_\xi < 1$ we are dealing with light tails. In the same way as with the Hill plot, we can draw a plot that help us distinguish which tail we are dealing with. In Figure 3.4 we show an example of the CV plot (with the residual CV being the black line) for a sample of size $n = 10^5$ coming from a Gaussian distribution. First of all, we only use those order statistics that are closest to the tail. In Figure 3.4 only the last thousand order statistics are shown, which are the ones closest to the tail in this case. There are three zones to remark here. The red line represent the statistical test developed in [90] to reject the hypothesis of exponential tail. In the green zone in between the red lines, there is the region where we cannot reject the hypothesis of exponential tails. Our example data shows that on the order statistic between 99200 and 99250 the exponential tail of our distribution begins. The red regions are the ones where we can reject the exponential hypothesis because we are either dealing with heavy tails (upper zone) or light tails (bottom zone). This technique has been used throughout the thesis as a first assessment on the kind of tails we are working with. The tools to implement this concept are in the CRAN package "ercv" [53].

Part II

Sky-High Quantile Estimation for CRTES

Chapter 4

Sky-high Quantile Estimation with Weibull Tails

4.1 Introduction

The increasing automation of CRTES, such as those in cars and planes, leads to more complex and performance-demanding on-board software and the subsequent adoption of multicores and accelerators. This causes software's execution time dispersion to increase due to variable-latency resources such as caches, networks on chip, advanced memory controllers and the like. CRTES undergo strict V&V processes against their corresponding functional safety standards (i.e. ISO26262 in automotive [89] and DO178B/C[139] in avionics). Timing V&V processes require collecting evidence supporting that software will execute correctly and *timely*. In particular, those processes provide evidence that the risk of violating deadlines for critical real-time software is residual, since functional safety standards acknowledge that risk cannot be completely avoided. Thus, they impose thorough V&V processes that allow deeming such risk as "sufficiently low" so that it can be neglected. In this line, pWCET estimates allow to quantify such residual risk.

EVT [38], appropriate for risk analysis, is used to model the right/upper tail of execution time distributions. In the context of pWCET estimation, exponential tails delivered by EVT, see Figure 1.2, have been shown by argument [7] and empirically [156] to provide a reliable tail model for pWCET estimation. However, tightness of those exponential tails is limited. Tails lighter than exponential ones (so with $\xi < 0$) can deliver tighter bounds, as discussed in [3] and later in Section 4.2. Yet, in the context of EVT, either GEV or GPD, distributions with $\xi < 0$ have a compact support, i.e. they have an absolute maximum value that cannot be exceeded. Hence, light tails in the case of EVT have an intrinsic risk of delivering optimistic tail distributions. This has some key implications in the fitting process, since a sufficiently large sample is needed to guarantee that light tail fitting is reliable for arbitrarily low exceedance probabilities. The target of this chapter is overcoming this limitation of the data and delivering a practical solution to obtain pWCET estimates tighter than those of exponential tails while preserving reliability. We do so by complementing EVT with survivability analysis as the theoretical ground for our hypothesis.

Analysis. We formally show that tail modelling in the context of *survivability analysis* [43] targets analogous questions to those of risk analysis. Then, we show that *log-concave*[1] distributions [112], inherited from survivability analysis, deliver the tightest distribution models, but existing fitting methods fail to model tails [14, 58], needed for pWCET estimation.

Proposal. We propose the use of a subset of log-concave distributions: Weibull tail distributions with increasing hazard rate, i.e. with shape $\beta \geq 1$, neither Reverse Weibull as in EVT nor full Weibull as in survivability analysis. Our approach provides analogous accuracy to that of log-concave distributions, without limitations to extend them to arbitrarily low exceedance probabilities, as needed for pWCET estimation.

[1]A function f is logarithmically concave log-concave for short, if the function $log(f(x))$ is concave.

Assessment. We compare our method and EVT alternatives with a bootstrap analysis on large data samples (10^7 measurements) from a railway case study as ground truth. Our results provide evidence on the reliability of our approach and significant pWCET reductions with respect to exponential tails obtained with EVT, thus allowing to trustingly increase system utilization noticeably.

4.2 EVT limits

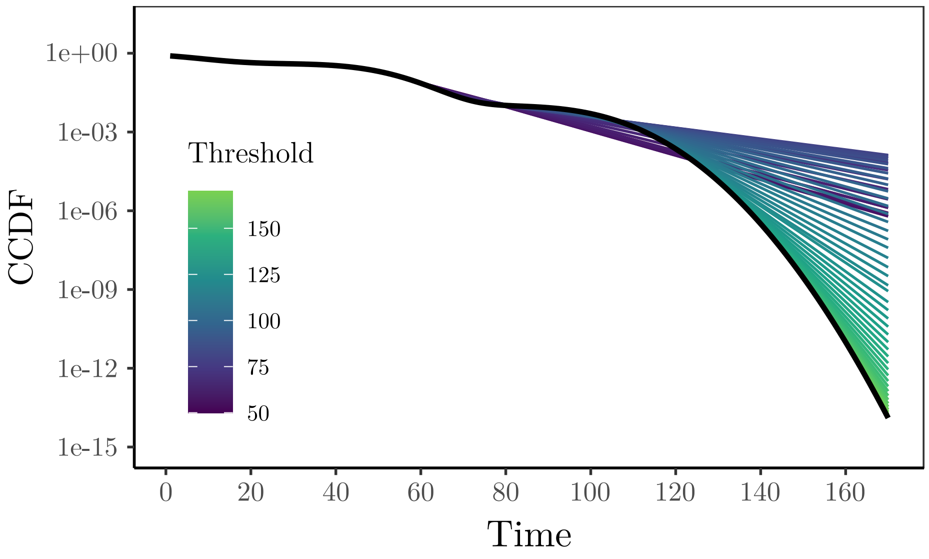


Figure 4.1: *EVT estimation on the mixture distribution (three Gaussian with parameters* $\mu = \{5, 50, 100\}$, $\sigma = \{10, 10, 10\}$ *and weights* $w = \{0.60, 0.39, 0.01\}$, *respectively.)*

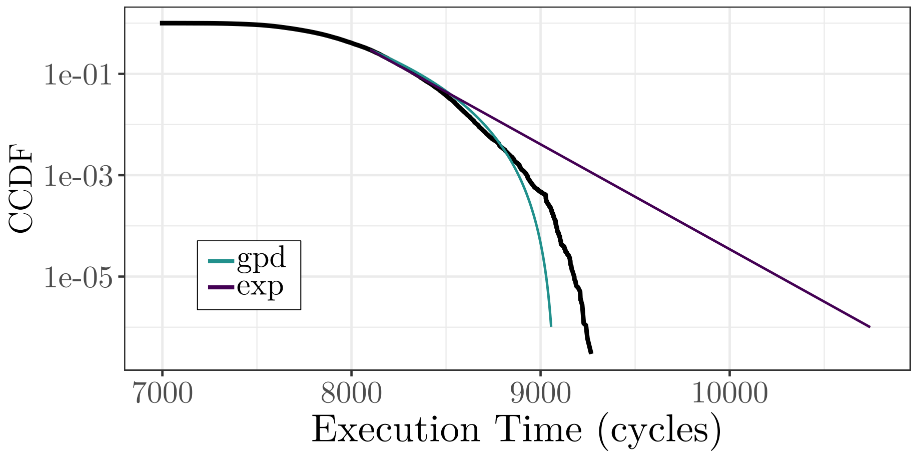


Figure 4.2: *CCDF for GPD* ($\xi < 0$) *and exponential tails from EVT for TEST8. Both models are fitted with a sample of* $n = 1000$ *observations out of all* $n = 10^7$ *observations made.*

CRTES programs have a finite WCET as discussed in Section 2.1. A distribution with a finite maximum value is said to have compact support. In that situation the most appropriate tail profile to model the input data with will be a light one, i.e. with extreme value index $\xi < 0$. In [44] it has been argued that heavy-tails, i.e. $\xi > 0$, are too pessimistic to model distributions with compact support. As an upper limit for the model, exponential tails, $\xi = 0$ can be proposed. As an example, we show the results of this reasoning. In Figure 4.2 we show the fit of an execution time profile from a program running in CRTES. The execution time profile contains $n = 10^7$ samples. Although for modelling purposes, we tried to fit a GPD with an exponential tail and a light tail using only a sample of $n = 10^3$ realizations. In the figure we can appreciate how the exponential model in this case is overly pessimistic for probabilities as low as $p = 10^{-6}$. On the other hand, the GPD model with light tails is closer to the real distribution, however the model underestimates, thus being unsafe in the context of MBPTA. The reason the GPD with light tails underestimates, is because its compact support nature. When trying to fit the small sample of $n = 10^3$ with a light tail, the maximum value on that sample will serve as the reference for the GPD. Therefore, if the maximum value for the sample is smaller than the maximum value of the whole sample, the GPD will underestimate. Nonetheless, the GPD with light tails is a good model in a vacuum for these kinds of distributions. However, in modelling the context always matters. Therefore, while in general the GPD with light tails is a good model, in the context of MBPTA it is a risky model.

We have just discussed how the exponential tail can be used as an upper-bound model for execution time profiles of CRTES. Recall that we discussed in Section 2.2.3 how the second fundamental theorem of EVT, Theorem 3, was fulfilled when the threshold $u \to \infty$ and that it could bring uncertainty to the estimation. Figure 4.1 shows a mixture of three Gaussians with $\mu = \{5, 50, 100\}$, $\sigma = \{10, 10, 10\}$ and weights $w = \{0.6, 0.39, 0.01\}$. Each mode around Time $\{5, 50, 100\}$ represents a Gaussian in the mixture. For each different tail threshold u such that $u > 60$, we fit an exponential tail, given that Gaussian distributions have exponential tails. In Figure 4.1 we see three scenarios that we exemplify with approximate ranges of u. For values of u around $[60, 80]$ (purple lines), we see how the exceedance probability of the mixture is underestimated in the Time range $[80, 120]$ (probabilities $10^{-3} - 10^{-5}$). For u in the range $[120, 140]$ (blue-green lines), GPD over estimates the reference distribution. For values of u above 140 (green lines), the estimate becomes increasingly tight. Depending on the threshold selected, even if the exponential is the appropriate model to fit the tail of a mixture of Gaussians, it can lead to optimistic pWCET.

In that regard, one of the goals of this thesis is to overcome the limits of EVT when working in the context of MBPTA. More specifically, in this work our goal is to devise a more flexible model for the tails than the GPD with light tails and the exponential by making use of Risk Analysis theory.

4.3 On the Use of Light Tails and Risk Analysis for WCET Estimation

4.3.1 Risk and Survivability Analysis

EVT is used in risk analysis to predict extreme (rare) events with the objective of proving that risk is below specific thresholds (e.g. financial risk). Survivability analysis, while it also focuses on predicting extreme events, it has opposite goals: proving that survivability is above specific thresholds (e.g. human life duration). Hence, both analyses target the modelling of extreme events, but with different objectives.

The type of distributions used to model those extreme events (i.e. tail distributions) differs across both analyses.

- EVT (either GPD or GEV) is used in the context of risk analysis, and it has been used so far in pWCET estimation building on the idea that exceeding a specific execution time bound is a risk.
- For survivability analysis tail distributions can be split into two types: Decreasing Hazard Rate (DHR) and Increasing Hazard Rate (IHR) distributions. The boundary between those two categories corresponds to exponential distributions, which can be regarded as part of both.

In the context of pWCET estimation, we focus on IHR distributions, since they include those distributions that, as values get higher, the probability of realization increases. In our context this means that, as the program runs, there is an increasing probability of finishing the execution, which is the case of real-time programs that need to have a finite execution time to meet their deadline. Formally stated, a random variable X is IHR if the hazard rate function is increasing, where the hazard rate function is defined as:

$$h(x) = \frac{f(x)}{1 - F(x)}, \quad x \in support(X), \tag{4.1}$$

where f and F stand for the Probability Density Function (PDF) and the CDF of X, respectively. In Equation 4.1, $support(X)$ is a function representing the subset of the domain in which the random variable (X) probability is defined (i.e. it is not zero). In fact, the hazard rate function which assesses the IHR property is equivalent to the convexity H function, where $H(x) = -log(1 - F(x))$, called cumulative hazard rate.

4.3.2 IHR Distributions in Survivability Analysis

Log-concave distributions, as they can have an arbitrarily large number of parameters, are one of the best tools to fit data. On the other hand, they are generally defined only for the range of data observed. Hence, they are unable to model the distribution beyond that range, which is not useful for pWCET estimation. Some methods smooth the distribution by means of convolutions with other laws (e.g. Gaussian) to better fit the mode of the distribution [14, 58, 112]. While such an approach delivers a distribution that spans beyond the range of the data observed, it is tuned to model central behavior (not the tails) and so inherits the original problem we aim to tackle: an appropriate law needs to be identified for tail modelling.

Weibull distributions, among others, have often been used to model IHR distributions. However, they are unable to fit all tail distributions, so they are used only in specific contexts where the problem at hand matches the shape of those distributions [112, 174]. Therefore, while survivability analysis opens new opportunities to model pWCET, this is yet unexplored and existing distributions, in general, may not fit the needs of pWCET estimation. In this chapter we address this challenge by contextualizing the needs of pWCET estimation and defining distribution families able to model pWCET distributions reliably, tightly and without incurring on the limitations imposed by log-concave distributions.

4.4 Equivalence Between IHR and Non-heavy Tails

In this section we introduce the connection between risk and survivability analysis in the context of pWCET estimation, which will lay out the ground for our hypothesis in further sections. pWCET estimates should be obtained theoretically under the assumption of the law of extreme events characterized by quicker decay in the tail than an exponential law, or equal, in the limit case. Exponential tails have been regarded as the appropriate (limit) model in practice [3, 149]. An exponential decay is a memoryless process where the probability of the process to complete is constant regardless of how long the process has been progressing. In the context of pWCET estimation, this corresponds to having a constant probability for the program to finish its execution despite the time elapsed since the program started running. Instead, the theoretical solution for pWCET estimation indicates that, the longer the program has been running, the higher the probability of finishing. That is, if X corresponds to execution time as a non-negative random variable, this assumption can be formally stated as follows if $s < t$ and $x > 0$:

$$P(X > t + x \mid X > t) \leq P(X > s + x \mid X > s), \tag{4.2}$$

In the context of extreme events, where u is the threshold upon which values belong to the tail of the distribution, s, t have to be large enough so that $s, t > u$, for some $u > 0$ [2].

In the general case of EVT (e.g. GPD), the tail decay can be described as follows for heavy, exponential and light tails respectively:

$$\begin{array}{llll} \text{if} \quad \xi > 0, & \text{then} \quad P(X > t + x|X > t) \geq P(X > x) & \text{for} & x > t. \\ \text{if} \quad \xi = 0, & \text{then} \quad P(X > t + x|X > t) = P(X > x) & \text{for} & x > t. \\ \text{if} \quad \xi < 0, & \text{then} \quad P(X > t + x|X > t) \leq P(X > x) & \text{for} & x > t. \end{array}$$

Note that the assumption described by Equation 4.2 for extreme events implies $\xi \leq 0$. Hence, the assumption for pWCET estimation matches the formulation above for light and exponential tails from GPD, inherited from risk analysis since, in the context of pWCET estimation, the longer the

[2]Note that the threshold u is not larger than the theoretical threshold corresponding to the asymptotic behavior of the tail from the second fundamental theorem in EVT [15, 132].

program has been running, the higher the probability of finishing. This matches the known concept in reliability modelling of IHR, which is therefore appropriate for pWCET estimation.

Building on Equation 4.2, and given a fixed x, we have the following equivalent assumption:

$$P(X < t + x|X > t) \geq P(X < s + x|X > s), \quad \text{if} \quad s < t.$$

Given that $s < t$, then $P(X > t) < P(X > s)$, and hence, we can elaborate the equation above as follows:

$$\frac{P(X < t + x) - P(X < t)}{P(X > t)} \geq \frac{P(X < s + x) - P(X < s)}{P(X > s)}.$$

If we use the CDF expressions instead, we have the excess distribution:

$$\frac{F(t + x) - F(t)}{1 - F(t)} \geq \frac{F(s + x) - F(s)}{1 - F(s)}, \quad \text{if} \quad s < t. \tag{4.3}$$

If x tends to 0, then the cumulative probability ranges, between t and $t + x$ and between s and $s + x$, reduce to the particular probabilities at t and s respectively:

$$h(s) = \frac{f(s)}{1 - F(s)} \leq \frac{f(t)}{1 - F(t)} = h(t), \quad \text{if} \quad s < t. \tag{4.4}$$

As shown, Equation 4.4 – which we derive from Equation 4.2 – builds upon the hazard rate function shown in Equation 4.1. Hence, it models IHR distributions for survivability analysis, analogously to the GPD formulation for risk analysis. Note that in Equation 4.4 the equality case corresponds to a constant decay rate, hence a constant hazard rate function.

Log Concavity: In order to use IHR distributions for pWCET estimation, we build upon the following theorem proven in [43] and [84]:

Theorem 5. *Given a non-negative random variable X, with f and F the PDF and CDF, respectively (where $H(x) = -\log(1 - F(x)), \quad x \in support(X)$),*

$$\log(f) \textit{ concave} \Rightarrow X \textit{ IHR} \Leftrightarrow H \textit{ convex}. \tag{4.5}$$

Note that X is IHR in the tail, i.e. $(X \mid X > u)$ is IHR for some threshold $u > 0$, if and only if Equation 4.2 holds for all $s, t > u$ and, therefore, X is log-concave. Thus, by using log-concave distributions, IHR holds by construction.

A non-negative function is log-concave if its domain is a convex set, and if it satisfies the inequality $f(\theta x + (1 - \theta)y) \geq f(x)^{\theta} f(y)^{1-\theta}$, for all x, y in the domain of f and $0 < \theta < 1$.

In order to test IHR one could make use of the log-concavity of the probability density function, which would give a convex H function and hence, IHR.

Given an appropriate threshold u so that $(X \mid X > u)$ is IHR (and log-concave), we can fit a log-concave density function to the tail by using the maximum likelihood approach, as detailed in [14, 58]. Regardless of whether we fit the best log-concave distribution or a distribution function family preserving log-concavity but with much fewer parameters, as we do in this work, the exceedance threshold (u above) must be estimated to use the appropriate set of tail values from the sample for fitting. In particular, we build upon the work by Hazelton [84] that provides a procedure for testing whether we can reject the hypothesis of log-concavity for a given threshold u.

4.5 Weibull Tails (TailW) for pWCET Estimation

In Section 4.2 we have concluded that light tails with compact support are likely optimistic (thus unreliable) for pWCET estimation. On the other hand, exponential tails are the limit distribution for appropriate pWCET distribution models, hence being reliable but likely pessimistic. In order to support this reasoning, we showed in Figure 4.2 how the GPD (with $\xi < 0$) and *exp* ($\xi = 0$) from EVT produce lower and upper bounds respectively to the pWCET distribution estimation for decreasing exceedance probabilities. Hence, we need an alternative model that has to satisfy the following properties:

1. Must have IHR in the tail (so H convex in the tail), thus having positive memory and *evi* < 0.
2. Must not have bounded (compact) support, not to suffer the same problems as GPD (with *evi* < 0).

The set of H-convex probabilistic models, i.e. with log-concave densities, satisfies the properties above and includes *all* probabilistic models satisfying those properties. However, as explained in Section 4.3, those distributions may have a large number of parameters, which reduces the number of degrees of freedom. Furthermore, the fitting process allows describing them only for the probability range where data exists, which is useless for pWCET estimation.

Weibull distributions[3] have been often used to model survival processes such as, for instance, the lifetime of processors [174]. Failure rates over processor lifetime are usually shown in the form of a bathtub curve, where the failure rate decreases during the beginning of the lifetime (so DHR), since failures due to infant mortality are frequent. However, the more the processor survives in this phase, the lower the hazard rate. Eventually, a near-flat phase is reached where the hazard rate is nearly-constant, until the end of life period is approached, when the hazard rate increases (so IHR) until the processor eventually fails. Obviously, such a distribution does not meet the properties indicated above since it should be IHR, and Weibull distributions may have DHR for at least part of their support. However, if $\beta > 1$, where β stands for the shape, Weibull tails (*tailW*) are IHR and allow covering all the spectrum between GPD with $\xi < 0$ and exponential tails. In fact, if we allow $\beta \geq 1$, the boundary case where $\beta = 1$ corresponds to the exponential tails.

4.5.1 Formal Definition of *tailW*

The *tailW* law is constructed using the excess probability function, shown in Equation 4.3. Thus, the CDF is:

$$F(x, \alpha, \beta, \nu) = 1 - \exp\left(-\alpha(x+\nu)^{\beta} + \alpha\nu^{\beta}\right), \tag{4.6}$$

for $x \geq 0$, $\alpha > 0$, $\beta \geq 0$ and $\nu > 0$. We consider *tailW* law with ν fixed and $\beta \geq 1$. The former reduces the cost of parameter estimation (only 2 parameters need to be estimated instead of 3) at the expense of delivering negligibly more pessimistic tail models. The latter ($\beta \geq 1$), as explained before, restricts *tailW* distributions to the domain of IHR. The likelihood ratio in the tail is described by:

$$l(x; \alpha, \beta, \nu) = n\left(\log(\alpha) + \log(\beta)\right) + (\beta - 1)\sum_{i=1}^{n} \log(x+\nu) - \alpha \sum_{i=1}^{n} \left((x+\nu)^{\beta} - \nu^{\beta}\right), \tag{4.7}$$

and the MLE to fit the *tailW* law to the tail is obtained with numerical methods. The full definition of the *tailW* and a numerical method based on MLE to estimate its parameters can be found in the R package *distTails* [133, 164].

Since the purpose of tail prediction is only modelling tails, for which we lack sufficient empirical data to rely on the empirical quantile, we need to fit an appropriate law for the tail. However, for the rest of the distribution we can simply rely on empirical data. Hence, we can resort to a semi-parametric model, where for a fixed threshold u, the law for $x < u$ is given by the empirical law and for $x > u$ the law is given by a parametric model (e.g. *tailW* or *exp*).

4.6 Fitting Protocol

In order to use *tailW* distributions, we define an application protocol that guarantees reliability and maximizes tightness. Consider a sample, X, $\{x_1, \cdots, x_n\}$, and a fixed threshold u to define the tail. We start by checking that the sample preserves the IHR property (log-concavity) for the considered exceedance threshold u as described in [84]:

[3]The Weibull law for X is given by the CDF $F(x, \lambda, \beta) = 1 - \exp\left(-(x/\lambda)^{\beta}\right)$.

1. Apply the bootstrap log-concavity test in the tail of the sample for the given u.
2. If the null hypothesis of log-concavity is rejected with risk 0.05, take another sample and restart.

Note that the significance level is 0.05, so that 95% of the bootstrapped samples must pass the test, is a common value for statistical tests. From this point onward, log-concavity in the tail is assumed since it has already been tested. Also, $u \geq u_{EVT}$, where u_{EVT} is the threshold such that the theoretical approach from EVT can be applied. Then, the protocol continues as follows:

3. Fix $\lambda = F_{emp}(u)$.
4. Consider the sample Y given by $y_i = x_i/u - 1$ for i such that $x_i \geq u$.
5. Fit the *exp*, *tailW* and *logc* by MLE. Then, the loglikelihood for each law can be denoted by (where $\widehat{\Theta}$ corresponds to the set of parameters for a *logc* distribution)

$$\begin{aligned} \widehat{l}_{exp} &= l_{exp}(\widehat{\psi}; Y), \\ \widehat{l}_{tailW} &= l_{tailW}(\widehat{\alpha}, \widehat{\beta}; Y, \nu = 1), \\ \widehat{l}_{logc} &= l_{logc}(\widehat{\Theta}; Y). \end{aligned}$$

6. Test the null hypothesis *tailW* $\sim$ *exp* through the Likelihood Ratio Test (LRT) [124, 175] with risk $\alpha = 0.05$:

$$2(\widehat{l}_{tailW} - \widehat{l}_{exp}) < \chi^{2,1}_{0.95}.$$

   Note that, since *tailW* has 2 parameters and *exp* 1, the χ^2 test is applied with 1 degree of freedom (the difference). If it is true, then the *exp* model (PoT with *exp* in the tail) must be considered for high-quantile estimation, and the fitting process finishes since the simplest model must be used if the models are not proven to differ[4]. Else, we continue with the next step.

7. Test the null hypothesis *logc* $\sim$ *tailW* with risk 0.05:

$$2(\widehat{l}_{logc} - \widehat{l}_{tailW}) < \chi^{2,\delta}_{0.95},$$

   where δ is the number of parameters in *logc* fit minus 2 (those for *tailW*). If it is true, then *tailW* model (PoT with *tailW* law in the tail) must be considered for high-quantile estimation, and the fitting process finishes. Else, *tailW* may not be a sufficiently good fit. Hence, either we continue searching for *tailW* fitting with larger samples or another valid value for u, or we resort to the *exp* model (computing u_{EVT} and fitting the new tail as indicated before), which is known to be pessimistic but reliable for sky-high quantiles[5].

Overall, this application protocol is reliable by construction and aims at maximizing pWCET tightness.

4.7 Evaluation

While theoretically *tailW* distributions meet the properties needed to model pWCET distributions tightly and reliably, we verify empirically such hypotheses in several ways:

- We compare *tailW* distributions against sky-high quantiles for large (ground truth) data samples.
- We compare *tailW* against GPD (with $\xi < 0$) and *exp*.
- We apply LRT to compare *tailW* and the reference *logc*.

[4]If an *exp* tail is used for modelling, then EVT should be used to improve the fit, computing u_{EVT} such that $u_{EVT} \leq u$, and fitting the new tail.

[5]Note that typically, high quantiles are defined at the range 0.9-0.9999. Since we target very small exceedance probabilities, in line with safety standards requirements, we define sky-high quantiles those reaching values around $1 - 10^n$, where typically $n \in [-6, -12]$.

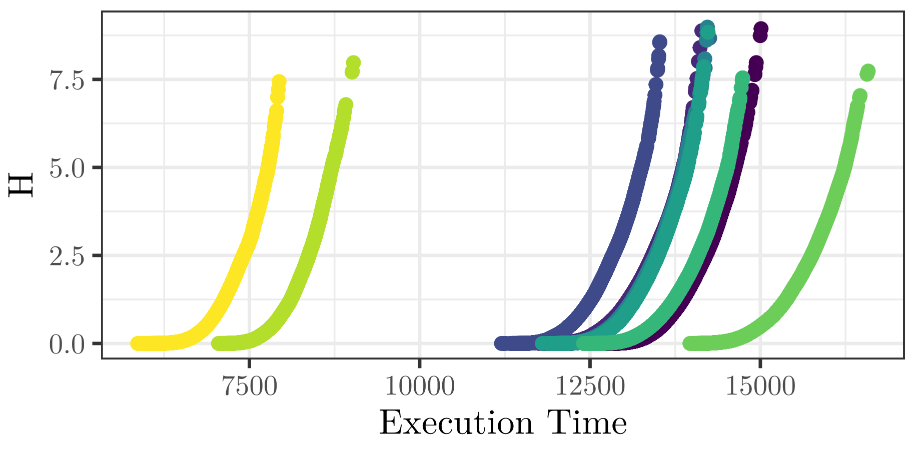


Figure 4.3: *H-plots for all tests for a bootstrap sample of* 10000 *observations. The x-axis shows execution times in processor cycles.*

The evaluation of *tailW* will be done using the 10 tests with size $n = 10^7$ each, from the ETCS validation suite run in an Field-Programmable Gate Array (FPGA) implementation of a LEON3+ architecture described in Section 3.1.1.

4.7.1 Assessing Model Hypotheses: H-Convexity and Light Tails

tailW relies on the convexity of the H-plot and lightness of the tail for the distribution modelled. By definition of our problem at hand (finite execution times), both properties must hold. Figure 4.3 shows the H-plot for the full samples of the railway case study (10^7 measurements). As shown graphically in the plots, all data sets are H-convex, thus in line with the hypothesis in Equation 4.2. The lightness of the tail can be assessed with the Coefficient of Variation (CV) statistic. Given that the CV in the context of the *gpd* for a certain threshold u is $CV(X_u) = 1/\sqrt{1-2\xi}$, where ξ is the *evi*, we are able to classify the nature of the tail. This method is properly developed and explained in the context of MBPTA in [3]. In Figure 4.4, we show the CV plot for all TEST traces, where the thick line corresponds to the trace for TEST6. The CV plot shows that, given that CV ≤ 1 in general, distributions have light tails. Small discontinuities in the data sample, which may happen due to random sampling, may create peaks when a low number of exceedances is considered; as in the case of TEST0 and TEST4, in Figure 4.4. Regarding TEST6, it is the only one with a slightly different behavior since the leftmost part of its CV is slightly heavy (CV ≥ 1), and it only becomes light after excluding half of the sample. We discuss TEST6 in more detail later.

4.7.2 Assessment with Large Data Sets

To assess the reliability and tightness of the *tailW* model, for each of the 10 TESTs, we conduct a bootstrap experiment consisting of generating 1000 random samples with 1000 observations each from the 10^7 observations collected for that TEST. Then, we fit the exponential (*exp*), the GPD with $\xi < 0$ and *tailW* models for each of those 1000 data samples. The pWCET value obtained for each of the three methods is assessed against the empirical distribution sky-high quantile $1 - 10^{-6}$ (so at an exceedance probability of 10^{-6}). Note that, by building on a data sample with 10^7 measurements, the highest quantile we could consider would be at $1 - 10^{-7}$, which would be fully dependent on the highest value in the sample. Relying on a single (randomly sampled) value may bring some instability, so we opted for considering a lower (still sky-high) quantile at $1 - 10^{-6}$.

Figure 4.5a shows the Quantiles Of BootStrap (QOBS) estimator for TEST0 in the form of a boxplot. All other TESTs except TEST6 have analogous behavior to TEST0, so we omit them for space needs. Results are obtained using different number of extremes (maxima) for tail fitting: we have used 500, 200, 100 and 50 extremes selected with PoT. The assumption formalized in Equation 4.2 lets us use the hypothesis on IHR for all data. Therefore, those numbers of extremes (between 50 and 500) can be reliably used for our analysis. As shown, *exp* provides highly pessimistic estimates with 500

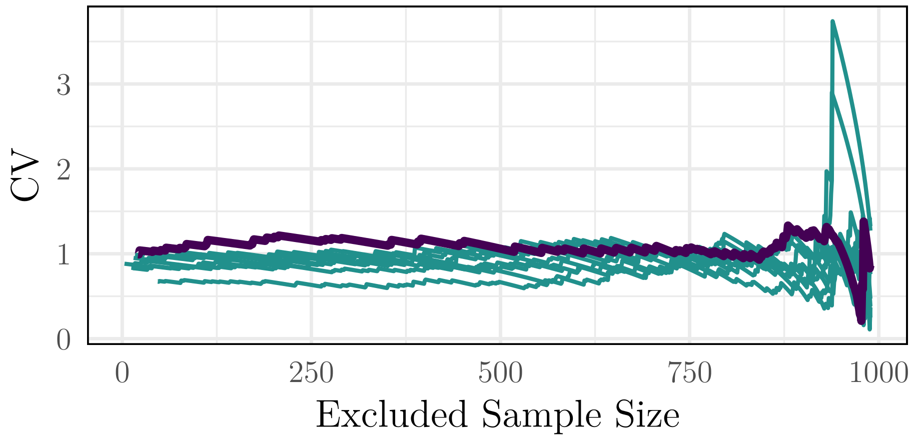
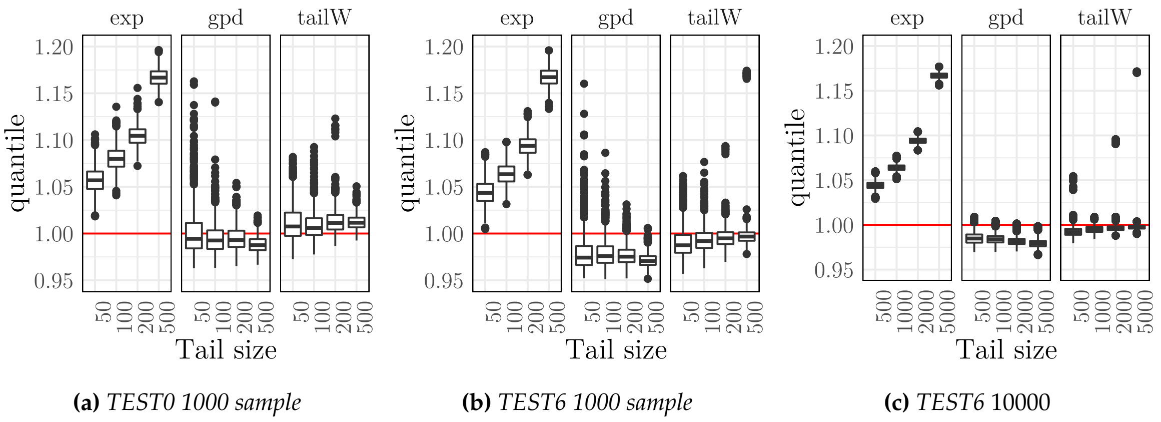


Figure 4.4: *Plot of the CV against the excluded sample size. The thick purple line corresponds to the TEST6, while the thin lines come from the rest of the tests. The CV plot was computed using the R package* ercv *[53].*



(a) *TEST0 1000 sample* **(b)** *TEST6 1000 sample* **(c)** *TEST6* 10000

Figure 4.5: *QOBS estimator distribution for TEST0 and TEST6 for 1000 and* 10000 *samples, under different models:* exp, gpd *and* tailW, *with different number of extremes: (500, 200, 100, 50) for samples of 1000 measurements and (5000, 2000, 1000, 500) for* 10000 *measurements.*

extremes, whose mean is in the range $[1.15, 1.20]$. Tightness improves as we decrease the number of extremes. However, fitting a distribution with a lower number of measurements brings increased uncertainty.

As expected, *gpd* tends to underestimate the sky-high quantile of the real data with a low number of extremes (50), and quantiles are wide. Hence, in this case uncertainty is relatively high and confidence intervals should be wide. By increasing the number of extremes, uncertainty rapidly decreases and quantiles narrow down. However, *gpd* further underestimates the sky-high quantile of the real data due to losing the asymptotic tail behavior by using values farther away from the maximum in the sample.

Finally, *tailW* tends to tightly and reliably estimate the sky-high quantile of the real data with few extremes (50), but with wide quantiles. As we increase the number of extremes up to 500, we observe that reliability and tightness are preserved, and quantiles quickly narrow down. Overall, *tailW* provides much tighter (and still reliable) pWCET estimates than *exp*, and does not suffer the underestimation problems of *gpd* due to the compact support of *gpd*, which should naturally worsen as we consider higher sky-high quantiles (a.k.a. lower exceedance probabilities). Note that, sporadically, *tailW* may lead to pessimistic sky-high quantile estimates (comparable to those for *exp*). As explained before, occasionally, *tailW* fitting may not be sufficiently good and then, our model

Table 4.1: *Lower Confidence Interval (LCI) for the reference values for all railway TESTs.*

Test	LCI	Test	LCI
TEST0	0.00%	TEST5	0.28%
TEST1	0.08%	TEST6	0.97%
TEST2	0.13%	TEST7	0.10%
TEST3	0.17%	TEST8	0.21%
TEST4	0.00%	TEST9	0.35%

resorts to exponential tail fitting to preserve reliability.

For TEST6 (Figure 4.5b), the larger the number of extremes considered (and so the higher the confidence), the closer the estimator to the reference value for *tailW*, but still most of the distribution is below the reference value. For larger samples of 10000 measurements – Figure 4.5c – estimates for TEST6 become more precise, but still slightly below the reference value (up to 1%). Since the reference value is obtained with point estimation, we have calculated its lower confidence interval (a.k.a. how much reference lines at 1.0 should be moved down in the y-axis). In particular, we use the *binomial confidence interval* [39], since it only requires the size of the sample used and the number of successes/failures. We compute the 95% confidence interval. Results for all TESTs, see Table 4.1, show that all confidence intervals are tiny except for TEST6, whose lower confidence interval is $\approx$1%, which means that *tailW* estimates are within the confidence interval of the ground truth value.

Since standards accept failure rates in the order of 10^{-5} to 10^{-9} failures per hour, and critical real-time tasks may run up to several thousands of times per hour, we consider exceedance probabilities of $1 - 10^{-6}$ and $1 - 10^{-12}$ per run for the pWCET estimates, thus showing the sensitivity of the different methods to the exceedance probability. In particular, we compare the only two reliable methods: *exp* and *tailW*. Given that *gpd* has been proven unreliable, we do not consider it for pWCET estimation. Therefore, we estimate the pWCET at such probabilities for *tailW* and *exp* with samples of 1000 execution time measurements. We consider, as in previous experiments, different numbers of extremes (50, 100, 200 and 500), and show the results normalized with respect to *tailW*.

As shown in Figure 4.6, *exp* delivers pWCET estimates between 5% and 20% higher than those of the proposed *tailW* method for the railway case study for an exceedance probability of 10^{-6} per run. As shown in Figure 4.5, tighter estimates are obtained for *exp* if fewer extremes are considered, whereas *tailW* accuracy is highly insensitive to this parameter. However, uncertainty increases if the number of extremes used is relatively low. Therefore, as the reliability required for the pWCET estimate increases, *tailW* provides higher gains.

Results for an exceedance probability of $1 - 10^{-12}$ per run are shown in Figure 4.7. Such a lower probability should be used for more critical tasks and/or for those tasks running more often. As shown, trends are similar to those of $1 - 10^{-6}$ exceedance probability, but at a higher scale, since *exp* pWCET estimates are typically between 15% and 50% higher than those of *tailW*. This increasing gap between the (tight) *tailW* model and *exp* model can be easily understood looking at Figure 4.2, where we see that the gap between *exp* and the actual distribution increases at decreasing exceedance probabilities. Note that achieving higher savings for tasks running more frequently implies that potential savings in system utilization can also be larger.

4.7.3 Comparing *exp*, *tailW* and *logc* Models

As explained before, *logc* distributions are the reference model, but they can only be used in the value range determined by input data. Nevertheless, we assess whether *tailW* delivers distributions that cannot be distinguished from *logc* ones statistically in the range where the latter are defined. For that purpose, we perform an LRT.

We have applied the LRT to the 10 TESTs on the large data sets. As shown in Table 4.2, the test is passed in almost all cases. In particular, the p-value is below 0.05 only for TEST1 (T1) with 100 extremes and TEST4 (T4) with 500 extremes. Hence, this result confirms that *M1* (so *tailW*) cannot be distinguished from *logc* distributions, thus supporting the high accuracy of the proposed model. In fact, in those cases where the test is failed, we only need to select an appropriate number of extremes

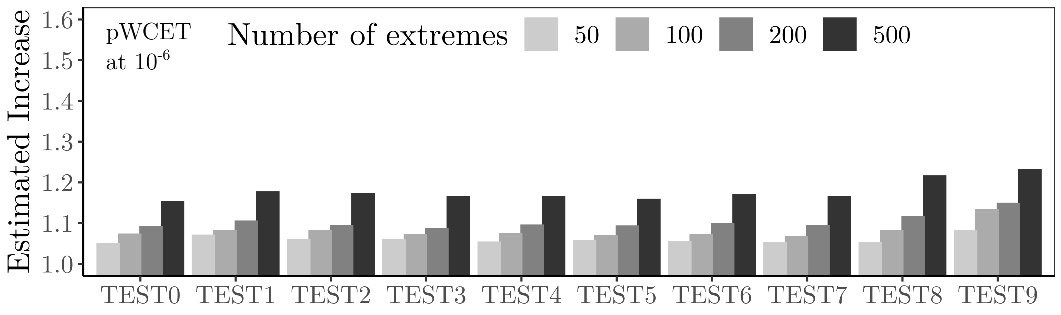


Figure 4.6: *pWCET estimate increase for* exp *with respect to* tailW *at an exceedance probability of* $1 - 10^{-6}$ *per run for the railway case study with different numbers of extremes for the tail.*

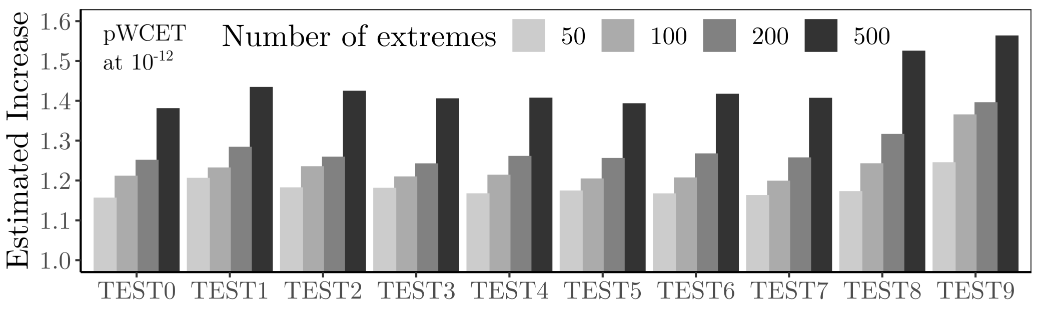


Figure 4.7: *pWCET estimate increase for* exp *with respect to* tailW *at an exceedance probability of* $1 - 10^{-12}$ *per run for the railway case study with different numbers of extremes for the tail.*

that ensures that *tailW* is indistinguishable. The default solution consists on collecting a new sample since both distributions are naturally indistinguishable. Note that, in general, α determines the ratio of false negatives (a.k.a. wrong hypothesis rejections). In our case, given that $\alpha = 0.05$, we would expect 2 rejections out of 40 tests, which is exactly what we obtained.

For completeness, we have applied the LRT to compare *exp* with *tailW*. A test pass would mean that the simpler model (*exp* has just 1 parameter whereas *tailW* has 2) should be used instead of the complex one. Our results (omitted due to space constraints) show that the test is failed in most of the cases, thus meaning that *exp* is unable to capture tail distributions with as much accuracy as *tailW*.

Table 4.2: *LRT p-values for the 10 TESTs comparing* tailW *and* logc *models.*

	tailW* vs *logc			
	500	**200**	**100**	**50**
T0	0.91	0.63	0.24	0.19
T1	0.07	0.08	0.04	0.72
T2	0.72	0.63	0.21	0.37
T3	0.21	0.57	0.30	0.94
T4	0.01	0.72	0.48	0.80
T5	0.98	0.25	0.67	0.19
T6	0.88	0.77	0.97	0.97
T7	0.75	0.85	0.34	0.59
T8	0.52	0.61	0.25	0.51
T9	0.11	0.88	0.21	0.92

4.8 Summary

The need for performing a proper analysis is crucial for the V&V of CRTES. While EVT has been proposed and applied successfully to derive WCET estimates, its results can be improved upon. The first alternative we proposed in this thesis to compute pWCET is the Tail of a Weibull (*tailW*), which combines characteristics from EVT and Risk Analysis. As shown in the results, *tailW* is less pessimistic than the exponential tail while not being optimistic like GPD with light tails model, thus producing tight pWCET models.

Chapter 5

Sky-high Quantile Estimation with Markov's Inequality

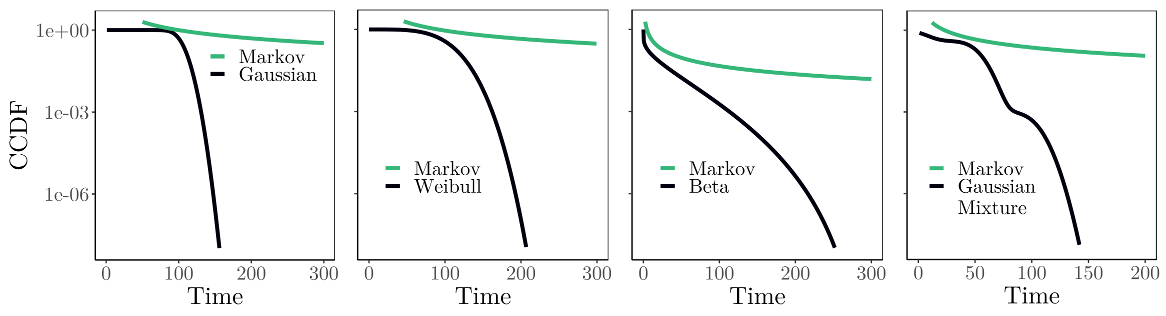
5.1 Introduction

Deriving WCET estimates for software programs with probabilistic means (a.k.a. pWCET estimation) has received significant attention during the last years as a way to deal with the increased complexity of the processors used in real-time systems. Many works build on EVT that is fed with a sample of the collected data (execution times). In its application, EVT carries two sources of uncertainty [26]: model uncertainty that is intrinsic to the EVT model and relates to determining the subset of the sample that belongs to the (upper) tail, and hence, is actually used by EVT for prediction; and statistical uncertainty that is induced by the sampling process and hence is inherent to all sample-based methods.

- Statistical uncertainty encompasses as the first aspect the testing conditions under which the experiments are performed in reference to those that can arise during system operation. Testing conditions that are representative or worse than operation conditions are the basis to attain representativeness of the sample data (execution time) [4, 12, 78, 118] so that the pWCET estimate holds during system operation. A second aspect of statistical uncertainty relates to the natural uncertainty of a sampling process that, in general, reduces as the sample size increases, and that is handled with confidence intervals. Sampling uncertainty impacts summary statistics (e.g. mean) and tail fitting methods, whose goodness – either of their hypotheses or outcome – is assessed with specific methods [12, 137].
- Model uncertainty, instead, relates to uncertainties intrinsic to the mathematical model used for tail prediction. In the case of EVT, model uncertainty relates to determining the threshold from which the upper tail starts. This threshold plays a key role on the trustworthiness (safeness) of EVT results since only samples above it (i.e. the maxima data set) are fed into EVT for pWCET estimation. There is not an exact mathematical method to derive this threshold. Instead, current methods estimate the tail of a distribution [30] based on plot inference [50, 52, 86] and regression analysis [29].

  As discussed in Section 2.2.3, the second fundamental theorem of EVT is satisfied when, for the excess distribution function F_u in Equation 2.12, the threshold $u \to \infty$. It is crucial to make a good estimation of this threshold because it will affect the pWCET as seen in Figure 4.1. Given that the computation of the threshold is not exact and there does not exist a closed form to compute it, one must rely on estimations to assess the tail of a distribution. This phenomenon is an example of model uncertainty. The model itself, in this case EVT, is the one bringing uncertainty in the calculations before dealing with the sampling process. To tackle this issue, we introduce Markov's Inequality, a probabilistic tool which produces provably safe upper-bounds by construction, which fits the needs for pWCET estimation.

In this chapter, we show that Markov's Inequality is a provably safe probabilistic tool which upper-



(a) *Gaussian distribution* **(b)** *Weibull distribution* **(c)** *Beta distribution* **(d)** *Gaussian Mixture*

Figure 5.1: *Markov's Inequality bound for the reference distributions.*

bounds by construction since it does not have any source of model uncertainty. The modification of Markov with the power-of-k function leads to tight upper-bounds even for very low probabilities. Finally, we show how to apply this probabilistic tool for any collection of samples coming from execution times of Real-Time systems.

We carry out the development of our methodology using analytical reference distributions which are representative of the tail profiles from execution time distributions of Real-Time systems. This reference distributions are described in Section 3.2 of the Experimental chapter. Furthermore, we assess the validity of our methodology with a railway case-study data coming from the European Train Control System (ETCS), which consists on 10 different test from the Emergency Module.

5.2 Chebyshev and Markov Inequalities for pWCET Estimation

This section analyzes the applicability of Markov's inequality as an alternative model to EVT for the problem of trustworthy pWCET estimation and shows that it is not subject to any model uncertainty, thus resulting in provably safe bounds for the analyzed distribution.

Recalling the discussion from Chapter 2 in Section 2.1; one upper-bounds the execution time profile of the task under analysis, when the resulting execution time at a given probability p for the pWCET , $et_{bound}(p)$, is higher (or equal) than the one of the task under analysis, $et_{real}(p)$. This can be expressed as $tightness(p) = \frac{et_{bound}(p)}{et_{real}(p)}$.

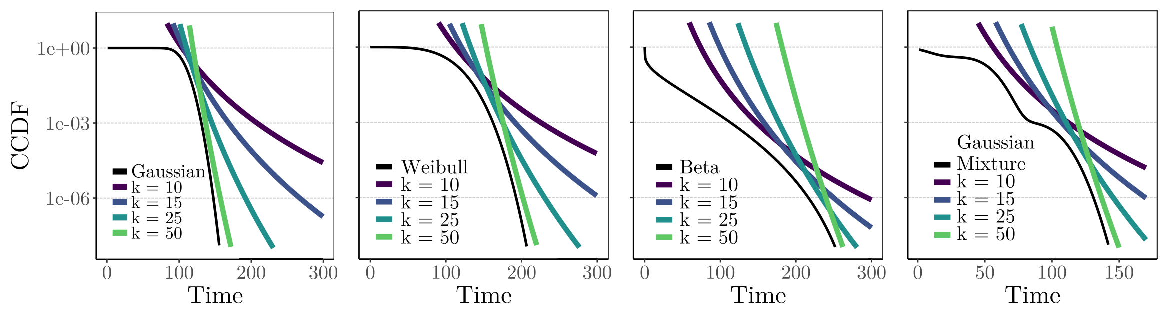
We introduced in Section 2.3.1 Chebyshev's Inequality, and one of its realizations – Markov's Inequality–, which are both probabilistic tools that satisfy MBPTA requirements for the pWCET problem because produce safe upper-bounds by construction. This chapter will be dedicated to study the viability and applicability of these inequalities to estimate sky-high quantiles.

5.2.1 Markov's Inequality on Low Probabilities

Besides trustworthiness, pWCET estimates are also required to be reasonably tight, especially for the range of relevant probabilities usually considered for pWCET estimation, e.g. $[10^{-6}, 10^{-15}]$. In this line, our analysis shows that Markov's inequality tends to be hardly useful for pWCET estimation.

This is better illustrated in Figure 5.1 which shows for all considered distributions the probability bounds given by Markov's inequality. We can observe that estimates are very pessimistic, orders of magnitude higher than the real probability. This includes the range of probabilities of interest for pWCET estimation. In fact, we see that Markov's inequality never goes below 10^{-2} for all distributions for the execution time value range plotted.

Observation 2. *Markov's inequality in its original form is too pessimistic to be usable in practice for pWCET estimation.*



(a) *Gaussian distribution* **(b)** *Weibull distribution* **(c)** *Beta distribution* **(d)** *Gaussian Mixture*

Figure 5.2: *MIK bounds for the reference distributions.*

5.3 Power-of-k functions for Markov's Inequality

One of the main insights of this work is that the key reason for Markov's inequality resulting in loose pWCET bounds lies on the fact that it builds on the identity function, $f(X) = X$, of a random variable. In this section, we show how a different function can lead to increased tightness on the produced pWCET estimates while preserving trustworthiness.

In particular, we contend that the power function for any $k \in \mathbb{R}_{>0} = \{k \in \mathbb{R} | k > 0\}$, i.e. $f(X) = X^k$, or power-of-k function for short, can be safely used instead of the identity function to obtain tighter and trustworthy pWCET bounds.

Definition 4. *Let X be a discrete random variable and k a positive real value. The expected value of X^k, also defined as the k^{th} (theoretical) moment, is:*

$$E(X^k) = \sum_x x^k P(X = x). \tag{5.1}$$

Corollary 2 (Markov's Inequality to the power-of-k)**.** *Let $X > 0$ and let the function f be the power-of-k function $f(X) = X^k$. Markov's inequality to the power-of-k yields:*

$$P(X \geq b) \leq \frac{E(X^k)}{b^k}. \tag{5.2}$$

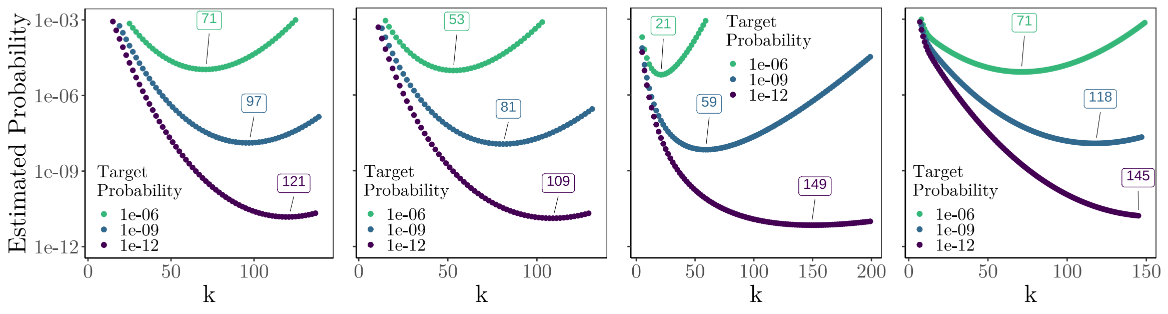
Hence, the probability that X takes a value greater or equal to b is bounded by $E(X^k)/b^k$. This makes Markov's inequality with $f(x) = x^k$ (**MIK** for short) a safe pWCET estimate when X represents the execution time of a program.

Proof. Theorem 4 holds true when Property 2.16 and Property 2.17 are fulfilled. The power-of-k function *does not fulfill those properties in general*. However, when the x domain is restricted to the positive real numbers $\mathbb{R}_{>0} = \{x \in \mathbb{R} | x > 0\}$, which in fact includes the domain of execution time profiles, the power-of-k function does fulfill Properties 2.16 and 2.17 since x is positive, so x^k is also positive and an increasing function. □

Overall, for this application scenario ($x \in \mathbb{R}_{>0}$), Equation 5.2 is an upper-bound when using the power-of-k function onto the reference distribution for any value of $k \in \mathbb{R}_{>0}$. Hence, it can be leveraged for pWCET estimation.

It is worth noting that other functions can exist that fulfill Properties 2.16 and 2.17. While exploring them is part of our future work, as shown in Section 5.3.1, MIK (i.e. $f(X) = X^k$) achieves very tight pWCET estimates which leaves small room for improvement.

Observation 3. *For every value of $k \in \mathbb{R}_{>0}$, MIK (i.e. Markov's inequality with $f(X) = X^k$) is a safe pWCET estimate when X represents the execution time of a program.*



(a) *Gaussian distribution* **(b)** *Weibull distribution* **(c)** *Beta distribution* **(d)** *Gaussian Mixture*

Figure 5.3: *Evolution of MIK bounds with the value of k.*

Note that there is no theoretical constraint on the maximum value of k, which can be any positive real number k.

5.3.1 Tightness of MIK for Increasing Values of k

Once we have established the safe use of MIK for pWCET estimation, we illustrate the impact of varying k on tightness. Figure 5.2 shows for several values of k, $\{10, 15, 25, 50\}$, and the reference distributions presented before, that MIK dramatically increases the tightness provided by Markov's inequality (Figure 5.1), while remaining a trustworthy upper-bound for every value of k. We also observe that MIK tightness remains for high exceedance probabilities, which hence makes it a promising model to provide pWCET estimates.

Observation 4. *Markov's inequality with $f(X) = X^k$ heavily reduces the pessimism of Markov's inequality.*

Intuitively, from Figure 5.2, higher values of k result in tighter estimates, i.e. minimizing the distance between the reference distribution and the upper-bound distribution. However, this is not always the case. For instance, if we take a closer look at the Beta distribution (Figure 5.2(c)), we see that at cut-off probabilities 10^{-3} and 10^{-6} the tightest MIK estimates are not obtained for the highest value of k evaluated (50).

Observation 5. *For a given threshold probability, higher values of k do not necessarily result in a tighter MIK bound.*

This is better illustrated with the examples in Figure 5.3 that shows quantitatively the evolution of the MIK bound obtained for varying values of k.

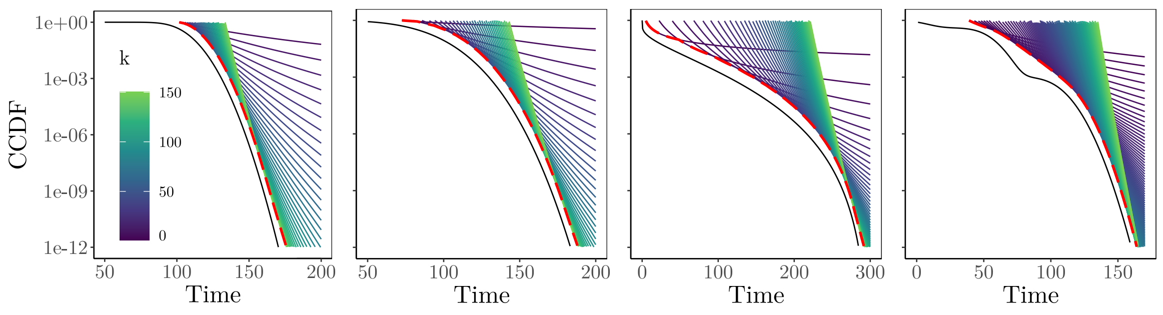
In this experiment, for the value of the target distribution at each probability, we evaluate MIK for different values of k. As it can be seen, for every threshold probability and distribution the value of k resulting in the tightest estimation is different. For instance, for the Gaussian distribution (Figure 5.3(a)) and target probability 10^{-6}, $k = 71$ produces the tightest estimate, while for 10^{-9} and 10^{-12} the best k is 97 and 121, respectively. As a general trend, we see that the lower the target exceedance probability, the higher the value of the best k is. Yet, the highest value of k evaluated for each target probability does not produce the tightest bound. Overall, for each probability there exists a value of k producing the tightest upper-bound, with the optimal value of k depending on the actual reference distribution.

Observation 6. *Increasing the tightness of MIK for each probability is an optimization problem on k which only increases accuracy and does not affect trustworthiness.*

In order to address this optimization problem, we propose **MEMIK** (Minimum Envelope for MIK, i.e. Markov's Inequality to the power-of-K). MEMIK combines the results of MIK bounds obtained for any value of k by keeping, for each point in an interval, the value of k producing the tightest estimate. This set of points form an envelope that is a provable trustworthy and tight tail bound by construction for any exceedance probability. Therefore, MEMIK improves the pessimistic upper-bounds of Markov's inequality (see Figure 5.1), with a much tighter envelope that is usable for

Algorithm 1 Compute an envelope using power-of-k

```
1: function MEMIK($t_{range}$, $t_{step}$, $p_{all}$, $max_k(p_{all})$, $k_{step}$)
2:     for $t \in t_{range}, t_{step}$ do
3:         for $p \in p_{all}$ do
4:             $mik_{best}(p) \leftarrow \infty$
5:             for $k \in ([1, max_k(p)], k_{step})$ do
6:                 $vpred \leftarrow EVAL(t, k, p)$
7:                 if $vpred < mik_{best}(p)$ then
8:                     $mik_{best}(p) = vpred$
9:                     $k_{best}(p) = k$
10:                end if
11:            end for
12:            $envelope(t, p) \leftarrow< mik_{best}(p), k_{best}(p) >$
13:        end for
14:    end for
15:    return $envelope$
16: end function
```



(a) *Gaussian distribution* **(b)** *Weibull distribution* **(c)** *Beta distribution* **(d)** *Gaussian Mixture*

Figure 5.4: *MEMIK bound (envelope) on the reference distributions.*

pWCET estimation. Formally, the MEMIK bound is defined as follows:

$$P(X \geq b) \leq \min_k \frac{E(X^k)}{b^k} \quad \text{for} \quad k > 0. \tag{5.3}$$

MEMIK, see Algorithm 1, which uses point-wise power-of-k Markov's inequality values, performs a simple complete exploration of MIK values over a given time range t_{range} and over a configurable range of k, determined by the maximum value max_k to be explored for each probability p in the set of probabilities of interest p_{all}. The granularity of MEMIK exploration over t and k is determined by the t_{step} and k_{step} parameters respectively. For each probability p in the interval of interest p_{all} (line 3) and k in the range determined by $max_k(p)$ (line 5), the algorithm estimates the value of the target distribution, *EVAL(t, k, p)* (line 6). To that end, we evaluate Equation 5.2 with $E(X^k)$, which corresponds to the theoretical k^{th} moment of the target distribution from Equation 5.1, obtaining *vpred* that we compare to the best MIK value so far for all considered k (line 7).

The minimum MIK value produced for a given t and across all k values ($mik_{best}(p)$) is stored, together with the corresponding $k_{best}(p)$, in the data structure *envelope* (line 12). Eventually, after iterating over the whole-time interval, the algorithm returns the *envelope* data structure (line 15) which holds the point-wise definition of the approximation envelope. Note that, if for any value t, the value of $k_{best}(p)$ matches $max_k(p)$, then $max_k(p)$ can be increased to find tighter bounds.

We applied MEMIK to our reference distributions, for which we can derive the theoretical moments. For this experiment, we varied the value of k up to 150 with $k_{step} = 1$. We obtained the envelopes depicted in Figure 5.4 which provides evidence that MEMIK produces tight and trustworthy estimates for all distributions, with an observed error of around 5%.

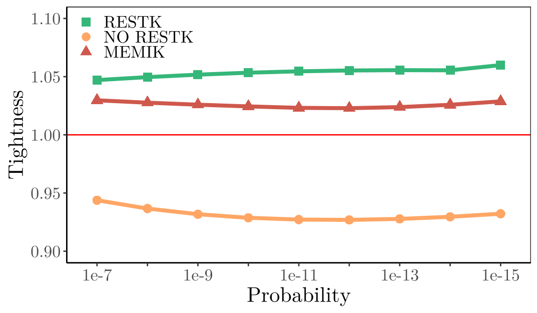


Figure 5.5: *MEMIK with sample moments (n = 1000 and n_{sims} = 100) on the reference Gaussian distribution with (RESTK) and without (NO RESTK) restricting k. Also, MEMIK evaluated with theoretical moments.*

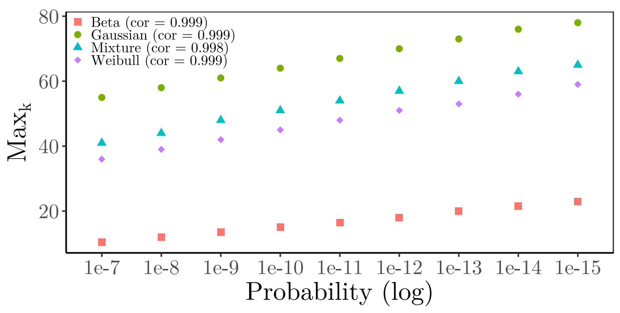


Figure 5.6: *Minimum max$_k$ values for the reference distributions used in this work.*

Overall, this section provides the key result that our proposal, Markov's inequality to the power-of-k (MIK), unlike EVT, suffers no model uncertainty at the theoretical level and hence, provides correct-by-construction pWCET estimates that are much tighter than those provided by the default Markov's inequality. This leaves sampling uncertainty as the problem to address.

5.4 Handling Markov Sampling Uncertainty

So far, we have been reasoning on examples for which we could compute the theoretical k^{th} moments for each distribution. This was possible since the distributions considered were known and, hence, we could compute exactly the value of each moment (i.e. $E(X^k)$ for each value of k) using its analytical closed form. However, we need to consider the scenario in which only samples are available. Hence, as for any other sample-based method, we need to deal with the underlying sampling uncertainty.

A commonality of sample-related methods like EVT [35], and something that we also assume, is that, input samples are independent and identically distributed [40] (i.i.d.) or at least exhibit extremal independence [142]. The i.i.d. property can be pursued with platform randomization [97] or data (time measurements) sample randomization [104].

For the case of the Markov's inequality, this translates into deriving the **sample moments**, referred to as $\hat{E}(X^k)$ (for each value of k). In particular, we need an estimator for high-order moments that can produce good estimates for any distribution.

5.4.1 Sample Moment Estimation

The k^{th} moment of a random variable X can be estimated as:

$$\hat{E}(X^k) = \frac{1}{n}\sum_{i=1}^{n} X_i^k. \tag{5.4}$$

In general, this estimator is the best one to deal with high-order sample moments [81], as it is asymptotically unbiased. Given that it asymptotically tends to a Gaussian distribution [16], the properties of the Central Limit Theorem's apply to it. However, the estimator is asymptotically unbiased [64] only when using large amounts of data. For instance, for a sample of a Gaussian distribution with $\mu = 100, \sigma = 10$ and $n = 10^3$, the difference between the 3rd exact moment, and the sample moment using Equation 5.4 is about 0.02%, it is between 1% and 3% for the 50th moment, and can be up to 160% for the 100th moment.

When the sample moment, i.e. the estimate of the k^{th} moment, is higher than the theoretical moment, there is a risk of underestimating the upper tail of the distribution by assigning to a certain probability a smaller quantile than it has in reality. The approach we propose to limit that risk consists in setting a maximum value of k allowed for each different probability, which we refer to as max_k. In order to illustrate the effect of not controlling max_k, Figure 5.5 assesses the tightness of MEMIK over a particular set of nine probabilities (from 10^{-7} to 10^{-15}). The red triangles represent the theoretical bound of MEMIK using Equation 5.3 with $E(X^k)$ being the theoretical moments, while the orange dots (NO RESTK) represent the application over multiple simulations of Equation 5.3 using the sample moment from Equation 5.4. On both applications we set up a high value for max_k, 150. We can see how the loss of consistency of the sample moment estimator on Equation 5.4 results in bad estimates.

An intuitive way to control the gap between the k^{th} sample and the theoretical moments is to vastly increase the sample size, which in our domain would require an unaffordable number of runs of the task under analysis. Alternatively, one can control the range of values of k explored in Equation 5.4. By doing so, we trade some tightness for trustworthiness. That is, if we explore values of k until a low max_k limit, we can see in Figure 5.3 that the theoretical bound is not optimal in terms of accuracy. On the other hand, small values of max_k also limit the inaccuracy of the sample moment estimator. Note that it is not possible to identify a general optimal value of max_k for any kind of data under analysis. The appropriate max_k value changes across distribution types, across the same distribution type with different parameters, and even across probabilities for a given distribution. For this reason, we propose the *restricted k method* (**RESTK**) that builds on the information gathered directly from the samples to derive max_k so as to produce trustworthy and tight results.

5.4.2 Understanding the Behavior of max_k

We gain insight on the behavior of max_k along 3 axes. We analyze i) whether for a given distribution there exists a pattern for max_k that can provide tight and safe results using the sample moment estimator; ii) whether this pattern can be predicted using only the information from the sample; and iii) whether the pattern can be generalized for any distribution.

We focus on the same example distributions used in previous sections. We fix the interval $[1, 150]$ as exploration range for k. In order to account for sample uncertainty, we perform $n_{sims} = 10^3$ Monte-Carlo simulations, each one considering a random sample of size $n = 10^3$.

We first compute Markov to the power-of-k (MIK) using Equation 5.2 with the sample moment estimator in Equation 5.4, for all selected k. In each simulation, we increase values of k and find the **first** (smallest) value of k that produces underestimation. This is computed by comparing the estimation with the actual value of the distribution. That is, we take the value of k for which the estimated quantile is smaller than the theoretical quantile. Then, we set $k-1$ as our max_k. For each Monte-Carlo simulation, we compute the max_k for all target probabilities. When all simulations are performed, we keep the smallest max_k for each probability. As a result, for each experiment of n_{sims} Monte-Carlo simulations, we obtain one value of max_k for each probability under study. We plot those values in Figure 5.6 from which we derive two main conclusions.

We observe that the values in Figure 5.6 follow a linear distribution. For each distribution we fit minima max_k values to a linear model and we derive the resulting correlation coefficient. The correlation coefficient quantifies the strength of the linear relation between two variables. It ranges between -1 and 1, with 1 or -1 indicating perfect correlation (all points would lie along a straight line).

The distributions we use in this work include 4 types of unimodal distributions, 2 multi-modal distributions and several parameters thereof (see Table 3.2 in Section 5.5). For all those distributions, Table 5.1 shows that the correlation coefficient is very high and steadily stays above 0.99. Even the empirical distributions derived from the case study analyzed in Section 5.6 result in a high coefficient of correlation (0.98 on average), despite they tend to produce more variability in the estimations. Hence, empirical evidence in support of linearity for the minimum observed value of max_k is strong for the distribution tails and range of probabilities representative for the WCET. Besides, the application of RESTK includes a method to assess whether the linearity property holds, building on the observed data. It is also noted that, similar empirical reasoning is used to support statistical arguments whether phenomena adhere to specific distributions builds on empirical tests. For instance, in the case of EVT, previous work uses QQ-plots to assess, based on observation, whether some tails can be considered exponential [105].

Table 5.1: *Correlation Coefficient for all the distributions used in this work.*

Gauss1	Gauss2	Weib1	Weib2	Beta1	Beta2	Gam1	Gam2	Mix1	Mix2	Mix3	Mix4
.999	.999	.999	.999	.999	.998	.999	.999	.998	.998	.997	.998

Overall, by restricting max_k, one can avoid under-estimating the upper tail of the modeled distribution. This is exemplified in Figure 5.5, where the green squares (RESTK) represent the estimates obtained for the Gaussian distribution when applying the max_k values in Figure 5.6. By restricting max_k, we address the lack of trustworthiness in Figure 5.5 (NO RESTK) and produce tight and trustworthy bounds. Analogous results are obtained for the other distributions.

5.4.3 Deriving max_k from Unknown Distributions

When deriving Figure 5.6, we built on the values of the theoretical quantiles so as to determine the value of k for which the sample moment starts underestimating. Given a sample of size 10^p, we can estimate confidently quantiles from exceedance probabilities bigger than $1/10^{(p-1)}$. In this case, based on the law of large numbers, it is very likely to see around 10 realizations whose probability is of the order of $10^{(p-1)}$ [151]. That is, on a sample of size $n = 1000$ we will see around 10 realizations whose probability is 0.01 (1%). Therefore, for a sample of size 10^4, quantiles corresponding to exceedance probabilities 10^{-3} and 10^{-2} can be estimated easily with the usual quantile estimation functions from statistical packages [88]. The computation of confidence intervals for quantile estimation can be done using distribution-free methods like Kaigh and Lagenbruch or bootstrap [152]; and in any case the accuracy of the estimation can be increased using a bootstrap technique to correct variability.

RESTK estimates at least three max_k points to construct its model and assess linearity. The latter is assessed by deriving the correlation coefficient for these three points. If such coefficient is above a threshold $th = 0.95$, we deem max_k boundary to be linear and vice-versa (in which case RESTK cannot be applied). For instance, the quantiles corresponding to exceedance probabilities 10^{-3}, 10^{-4}, and 10^{-5} can be estimated very accurately with a sample of size $n = 10^6$. These reference points allow us to assess when RESTK underestimates, and hence generate three max_k points, one for each probability. With those points, we can generate the regression line that projects max_k for any probability of interest (e.g., those in Figure 5.6) and assess that the correlation coefficient is above the desired threshold.

Algorithm 2 generalizes the RESTK process starting from a main sample of the distribution under analysis (size 10^p), selecting the range of k to explore and the number of n_{sims} to run. First, RESTK estimates the quantiles at given probabilities p_{test} from the main sample (Line 3), e.g. $10^{-(p-1)}$,$10^{-(p-2)}$, and $10^{-(p-3)}$. For each simulation, n_{boot} bootstrap samples of size 10^{p-3} are generated from the main sample (Line 5). For each of these samples, we compute the maximum k and maximum tightness for all the probabilities to test. The predicted value $vpred$ is obtained by computing MIK from Equation 5.2 with the sample moment estimator, $\hat{E}(X^k)$ in Equation 5.4, (Line 10). The tightness of the predicted value is computed (Line 11) by using as reference value $vref$ the estimated quantiles obtained before (Line 3). The algorithm finds the values of k in the considered range that produce the tightest estimate (Lines 15-16) and terminates its exploration as soon as a k that underestimates ($tightness < 1$) is found (Line 12). After exploring all selected probabilities for all k and all simulations, the algorithm returns the smallest max_k across all simulations (Lines 20-21). Once the max_k (e.g. for $10^{-(p-1)}$, $10^{-(p-2)}$, and $10^{-(p-3)}$) are obtained, RESTK builds a linear model for all possible probabilities (Line 24). The final check (Line 25) will ensure that the correlation of max_k is above $th = 0.95$, otherwise RESTK provides no pWCET estimate. As we show in Table 5.1, the correlation should always be close to 1 for max_k. The threshold $th = 0.95$ is a standard and stringent threshold for confidence in statistics, and used as a way to discard estimates that do not meet the safety criteria of finding a proper max_k.

RESTK enables the computation of a value for max_k that reduces the risk of underestimation for any probability of interest. This can be directly exploited by running MEMIK (Algorithm 1) on a predetermined range for k and for each probability p under study by using $max_k(p)$ as the upper bound for

Algorithm 2 Computing the boundary necessary to apply RESTK approach.

```
1: function RESTKBOUNDARY($sample, k_{range}, k_{step}, n_{boot}, n_{sims}, p_{test}, p_{all}, th$)
2:     for $p \in p_{test}$ do
3:         $q_{est} \leftarrow estimateQuantiles(sample, p)$
4:         for $sim \in n_{sims}$ do
5:             $sample_{boot} \leftarrow bootstrap(sample, n_{boot})$
6:             $max_k(p) \leftarrow \infty$
7:             $tightness_{best}(p) \leftarrow \infty$
8:             for $k \in k_{range}, k_{step}$ do
9:                 $vref \leftarrow q_{est}$
10:                $vpred \leftarrow MIK(k, sample_{boot}, p)$
11:                $tightness \leftarrow vpred/vref$
12:                if $tightness < 1$ then
13:                    break
14:                end if
15:                if $tightness < tightness_{best}(p)$ then
16:                    $current_k(p) = k$
17:                end if
18:            end for
19:            if $current_k(p) < max_k(p)$ then
20:                $max_k(p) = current_k(p)$
21:            end if
22:        end for
23:    end for
24:    $max_k(p_{all}) \leftarrow buildLinearModel(max_k(p_{test}), p_{all})$
25:    if $corr(max_k(p_{all})) < th$ then
26:        return no pWCET estimate
27:    end if
28:    return $max_k(p_{all})$
29: end function
```

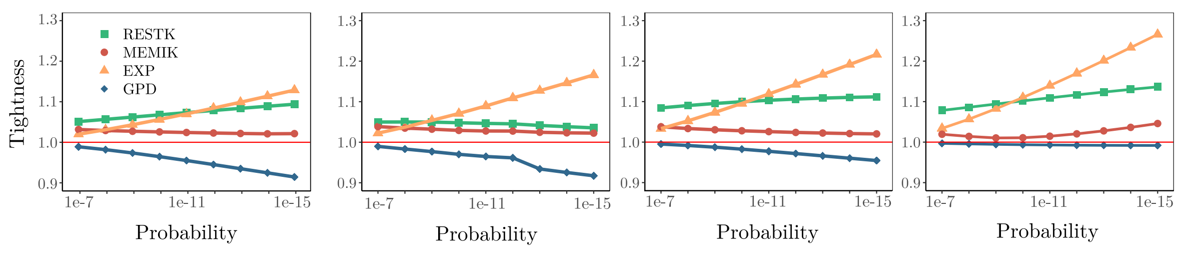
k, instead of considering an arbitrary range. Also, note that with RESTK, we use Equation 5.2 with sampled moments ($\hat{E}(X^k)$) as $EVAL(t, k, p)$ function in MEMIK. The rest of the MEMIK algorithm remains unchanged when using the RESTK method.

5.5 RESTK and EVT PWCET Estimates on Distributions

Our implementation of RESTK and MEMIK is programmed in *R* [133]. We run experiments on an Intel Core i5-7600K CPU clocked at 3.8GHz. The maximum execution time required per experiment was very short, 50 milliseconds or lower. We analyze values of k in the range $k \in [1, 150]$ with $k_{step} = 1$ to estimate max_k for all reference distributions, which, as shown in Figure 5.6 is a wide enough range to find the best max_k across distributions and probabilities. For all methods compared in this section, we use a sample size of $n = 10^6$. For RESTK, we set the number of bootstrap simulations to $n_{boot} = 2000$.

For EVT, we use the PoT methodology to fit tails and the CV Plot [52] to find the appropriate threshold for the PoT model. We use two EVT models fitted for pWCET estimation, namely exponential and light tails models. For each specific model, a different threshold using the CV plot will be found to ensure the best possible fit.

- *Exponential*: with an exponential model, the shape of the GPD is fixed to $\xi = 0$, which only leaves us to estimate the threshold u and the scale σ. The threshold is estimated with the CV Plot fixing $\xi = 0$, which finds where the exponential tail begins. Once we find the tail, we separate it from the rest of the sample and estimate the scale σ with it.
- *GPD light tails*: for the light tails model, we need the value of ξ, with $\xi < 0$, best fitting the data. Using the CV Plot we find the threshold u where the light tail begins. Then, we separate



(a) Gaussian (b) Weibull (c) Beta (d) Mixture

Figure 5.7: *MEMIK with sample moments ($n = 1000$ and $n_{sims} = 100$) on the reference distributions, hence restricting k (RESTK). Also MEMIK evaluated with theoretical moments (MEMIK), and EVT evaluated with exponential tails (EXP) and with a GPD with light tails (GPD).*

the tail from the sample and estimate the shape ξ and the scale σ.

Reference Distributions. We start the comparison with Figure 5.7 that depicts for the reference distributions the results of PoT with exponential and light tail models (EXP and GPD). It also shows the results obtained with MEMIK, which we can obtain as we have the actual distributions, and RESTK. Note that MEMIK provides the theoretical bound achievable with RESTK – it produces a safe bound and the tightest estimates. RESTK, EXP, and GPD build on a sample (the same one for a fair comparison) of the distribution. Following common practice, we show in Figure 5.7 the bias of our estimator, which is the expected value (mean) of RESTK output. It is noted that in RESTK application process all the distributions of this section fulfilled the linearity assessment (line 25 in Algorithm 2).

As we can see in this initial set of results, GPD tends to underestimate while EXP increases overestimation for high exceedance probabilities. RESTK produces values that are more consistent across all probabilities, improving EXP specially for higher exceedance probabilities. For 10^{-12} overestimates are 8%, 13%, 24% and 11% for the four reference distributions, respectively. The values increase to 13%, 20%, 37% and 17% for 10^{-15}. It can also be observed that the overestimation introduced by RESTK with respect to MEMIK to handle sampling uncertainty is limited: at 10^{-12} the difference is 4.75 percentage points (p.p) on average with a maximum of 8 p.p across all four reference distributions, and at 10^{-15} the overestimation difference is 5.25 p.p on average with a maximum of 10 p.p.

Extended set of Distributions. We consider a wider set of parameters for each distribution as listed in Table 3.2, resulting in 12 different distributions. The first distribution of each type (but the Gamma) is the reference distribution with the parameters used in previous sections. The rest of the distributions of each type encompass a different set of parameters to increase representativeness. The set of values explored for each parameter aims at showing the capabilities of RESTK under different scenarios. To that end we modify the following parameters.

1. the variance of the distribution for Gaussian1 and Gaussian2; and Mixtures1 and Mixture2
2. the shape of the tail of the distribution for Beta1 and Beta2, Weibull1 and Weibull2, Gamma1 and Gamma2, and Mixture3 and Mixture4.

This covers **all** possible variability scenarios as the scale and location do not affect the results for Markov's Inequality. As shown in [111], a change of location does not affect the inequality if the shift keeps the random variable positive, $P(X - a \geq b - a) \leq \frac{E(X-a)}{b-a} < \frac{E(X)}{b}$. Also, scaling the random variable X as λX where λ is real-valued, does not affect Markov's Inequality as $P(\lambda X \geq \lambda b) \leq \frac{E(\lambda X)}{\lambda b} = \frac{\lambda E(X)}{\lambda b} = \frac{E(X)}{b}$.

Looking at the results for the broader set of experiments in Table 5.2, we observe the following:

- GPD is always close to the true quantile, but in all cases it produces optimistic results. Furthermore, the higher exceedance probability, the more optimistic the estimate is. For instance, GPD on the Gaussian1 has a tightness of $\{0.93, 0.90\}$ at $p = \{10^{-12}, 10^{-15}\}$ respectively. This

Table 5.2: *Tightness of the different models (MEK stands for MEMIK and RES for RESTK).*

	probability 10^{-12}				probability 10^{-15}			
	GPD	**EXP**	**MEK**	**RES**	**GPD**	**EXP**	**MEK**	**RES**
Gaussian1	0.93	1.08	1.02	1.06	0.90	1.13	1.02	1.06
Gaussian2	0.90	1.20	1.05	1.14	0.86	1.28	1.03	1.11
Weibull1	0.91	1.13	1.02	1.09	0.87	1.20	1.02	1.09
Weibull2	0.96	1.09	1.03	1.04	0.94	1.14	1.05	1.04
Beta1	0.98	1.24	1.03	1.18	0.98	1.37	1.03	1.20
Beta2	0.98	1.17	1.03	1.11	0.98	1.26	1.02	1.13
Gamma1	0.93	1.09	1.03	1.07	0.89	1.11	1.03	1.07
Gamma2	0.95	1.09	1.02	1.06	0.92	1.13	1.02	1.07
Mixture1	0.92	1.11	1.03	1.03	0.88	1.17	1.02	1.02
Mixture2	0.94	1.16	1.03	1.07	0.90	1.23	1.03	1.05
Mixture3	0.88	1.25	1.03	1.15	0.83	1.37	1.02	1.13
Mixture4	0.94	1.15	1.02	1.15	0.91	1.23	1.03	1.16

behavior is observed for all reference distributions.

- EXP follows the opposite pattern. In general, we can see in Figure 5.7 that, for an exceedance probability $p = 10^{-7}$, the estimates across distributions are always safe and quite tight. However, for higher exceedance probabilities, EXP tends to give more pessimistic estimates. For instance, EXP on the Gaussian1 has a tightness of 1.01 at $p = 10^{-7}$ that increases to 1.13 at $p = 10^{-15}$. At this probability and across all distributions EXP overestimation is 21.8% on average, 37% in the worst case.
- The estimates with MEMIK, i.e. with theoretical moments, are very tight, below 6% for all distributions.
- RESTK achieves results similar to MEMIK, preserving tightness and trustworthiness. Even for very high exceedance probabilities, RESTK is able to produce consistent estimates. Building on the Gaussian1 distribution, we see that while EXP can achieve a tighter estimate for low exceedance probabilities (e.g. $p = 10^{-7}$), EXP suffers from increased pessimism for higher exceedance probabilities whereas RESTK stays stable. This behavior is more striking in distributions harder to analyze like mixtures. For instance, for Mixture1 RESTK not only maintains tightness stable across probabilities, $\{1.03, 1.02\}$, but it gets also a tighter bound than EXP for probabilities $p = 10^{-9}$ and beyond as seen in Figure 5.7. At 10^{-15} RESTK estimates across all distribution overestimate by 9.4%, far below the 21.8% of EXP. In the worst case it is 20% (with respect to 37% of EXP).

Overall, experimental results show the ability of RESTK to produce bounds suitable for pWCET estimation, being those trustworthy, tight and stable across probabilities and distributions, as opposed to existing models, which fail to meet all three goals simultaneously. The benefits of RESTK increase as the cutoff probability decreases to $10^{-12} - 10^{-15}$, which are the main range of interest for pWCET estimation considering maximum failure rates of 10^{-9} per hour and tasks running thousands of times per hour.

5.6 Railway Use Case

In order to evaluate the effectiveness of RESTK, we use an industrial critical real-time use case. In particular, we focus on the central safety processing unit of the European Train Control System (ETCS) reference architecture described in Section 3.1.1 where 10 different tests were conducted on a Cobham Gaisler LEON3 platform.

Ground truth. For real programs, for which the actual distribution is not known, common practice consists in using as 'ground truth' the observed quantiles for samples as large as reasonably possible (e.g. 10^4 [105] and 10^8 [149]). We follow the same approach and consider the quantile observed for

Table 5.3: *Tightness of the estimates for the ETCS case study.*

	probability 10^{-6}						
	GPD	**EXP**	**RESTK**		**GPD**	**EXP**	**RESTK**
TEST0	1.01	1.21	1.10	TEST5	0.98	1.12	1.07
TEST1	0.98	1.12	1.06	TEST6	0.94	1.06	1.01
TEST2	1.00	1.13	1.11	TEST7	1.01	1.16	1.09
TEST3	0.98	1.20	1.10	TEST8	1.01	1.23	1.12
TEST4	0.99	1.17	1.13	TEST9	0.98	1.11	1.10

the 10^7 sample as the reference value. More than two weeks of execution were needed to complete the execution of all the 10^7 runs per input vector (TEST).

Setup: The setup and parameters used to run RESTK are the same used for the reference distributions. As the number of runs we have is $n = 10^7$ per input set, we make projections for $p = 10^{-6}$, for which the observed frequencies closely match the actual probability (i.e. 95% confidence intervals are within 0.1% of the mean). In this case, the estimated quantile at probability $p = 10^{-6}$ is our ground truth. Only TEST6 is an exception to this and, due to the variability observed for that quantile, we use a 95% confidence interval, which is 1% off the mean. In this section, we use a sample size of $n = 10^4$ for GPD, EXP and RESTK. Note that MEMIK with theoretical moments cannot be used since the actual distribution is unknown and we only have sampled data.

Results: As part of the RESTK application process, all the 10 distributions fulfilled the linearity assessment (line 25 in Algorithm 2). As shown in Table 5.3, pWCET estimates are similar to those presented in the previous section. In particular:

- GPD achieves extremely tight estimates for 4 tests, with tightness up to 1.01, but on the other 6 tests it produces optimistic results. In general, while potentially very tight, GPD can easily underestimate the bounds.
- On the other hand, EXP never underestimates although it produces pessimistic results, as high as 23% for such a relatively low quantile.
- RESTK consistently produces tighter estimates than EXP in all tests for the use case. On average, EXP exhibits 15.1% pessimism, whereas RESTK reduces it to 8.9%.

Discussion: Overall, with the combined results over a wide set of distributions, shown in the previous section, and the results for the ETCS case study presented in this section, we conclude that RESTK consistently provides tighter estimates than EXP and improves for lower exceedance probabilities.

5.7 Summary

In the previous chapter, we showed how *tailW* improves pWCET estimates with respect to exponential tails while preserving trustworthiness. However, both approaches have some model uncertainty due to the need to determine where the tail starts. The other EVT alternative we propose in this thesis for the WCET problem is the use of Markov's Inequality. The key characteristic of Markov's Inequality is being an upper-bound for the probability of exceeding a given value by construction, hence without any model uncertainty. In this work, we managed to make the upper-bound very tight with the use of the power-of-*k* function, which yields the moments of a distribution. With said function, we can move from computing the moments theoretically to a sample moment estimator that allows us to compute the inequality for any sample. We devised the RESTK method to avoid the underestimation produced by the inconsistency of the sample moment estimator and, in our experiments, it provides tight and safe estimates.

Part III

Merging Hardware Event Monitor Data for CRTES

Chapter 6

Merging Hardware Event Monitor Data for Complex MPSoCs Using Order Statistics

6.1 Introduction

The Performance Monitoring Unit (PMU) in MPSoCs is at the heart of the latest measurement-based timing analysis techniques in Critical Embedded Systems. In particular, hardware event monitors (HEMs) in the PMU are used as building blocks in the process of budgeting and verifying software timing by tracking and controlling access counts to shared resources.

As the complexity of MPSoCs in modern CRTES platforms in domains like avionics and automotive continues to increase, so does the number of HEMs that can be tracked. In fact, current MPSoCs, already comprise hundreds of HEMs, e.g., the ARMv8(-A) architecture defines over 280 micro-architectural events [67]. This architecture is implemented by a set of processors such as the A53 (used by the NXP LayerScape family and the Xilinx ZynqUltraScale MPSoC), the A57, the A72, and NVIDIA's Carmel processor. Each processor implements a subset of those HEMs. For instance A53 implements 63, while more modern A57/A72 implement 92/85 respectively. We see a similar increasing trend in the NXP eXXX family with 180 HEMs in the e500mc and 262 in the e6500.

While the number of HEMs in current multicores is in the order of hundreds, they can only be read in small groups of 4-8 via user-visible performance monitoring counters (PMCs). This limitation relates to the hardware cost of routing the HEMs via long wires and multiplexors to access PMCs that can be accessed via software. Hence, several runs are required to read all the HEMs of interest, which are later 'merged' off-line to analyze the program behavior and reason about contention. For instance, to decide whether some tasks can be scheduled concurrently, we need to budget how much each one is expected to access each shared resource, which requires consistent reads of a large number of HEMs.

To make things worse, several runs of the same experiment in an MPSoC can result in inevitable variations in the timing behavior of the program, though its functional behavior is the same. This is due to the impossibility to control the entire hardware and software initial state in each run. In practical terms, this translates into variability in HEM readings (as high as 59% for processor cycles in our target system for relevant HEMs), with no variability observed in instruction count (as analyzed in Section 6.2.3). The engineer is confronted with a set of values (readings) for each HEM, that need to be merged to allow reasoning about multicore contention. Unfortunately, since HEM values from different runs to be merged can be subject to different (large) noise, it is challenging to merge them consistently so that merged HEM vectors – those where all HEMs of interest are included – resemble the values that would have been obtained if they could have been read all of them simultaneously in the same run.

In this chapter we introduce a HEM-Reading Merging (HRM) approach to guide the merging of HEM

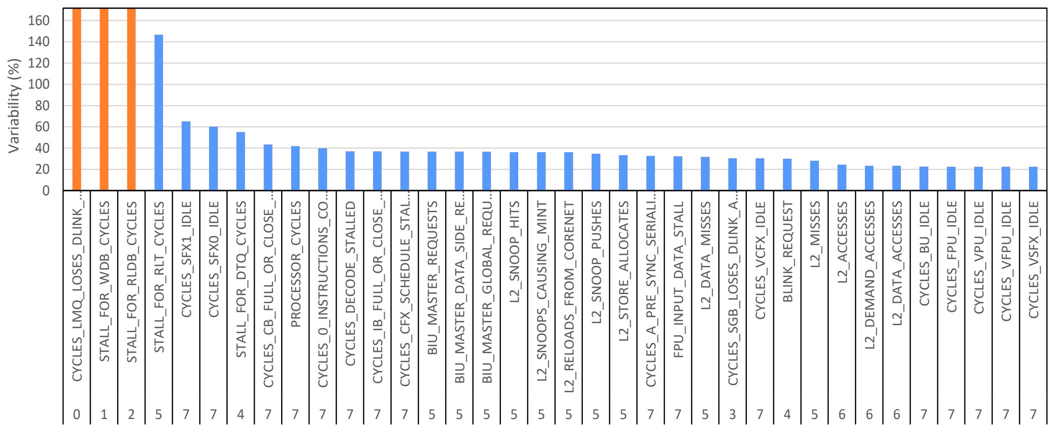


Figure 6.1: *Observed variability for several HEMs in the T2080.*

values subject to different noise. HRM identifies an anchor HEM, and defines groups of HEMs, each group with *PMCs - 1* HEMs plus the anchor. HRM performs several runs for each group of HEMs, ranks HEM values in each group using order statistics on the anchor HEM, and merges those HEMs with the same rank in different groups. Order statistics are non-parametric and hence, can handle the different distributions of the HEM values observed. Since noise-free HEM values cannot be obtained in general in complex MPSoCs, we evaluate HRM comparing the correlation across HEMs merged by HRM against their correlation when those HEMs are measured in the same run, thus under identical noise. Our results show that HRM captures accurately the correlation between HEMs, as opposed to blindly merging HEMs read in different runs.

6.2 Motivation

We motivate this work by showing how several of the 262 HEMs in the T2080 present significant variation (Section 6.2.1) and follow different distributions (Section 6.2.2). We also dig down into some of the reasons behind the observed variation (Section 6.2.3).

6.2.1 HEM Variability

On the NXP T2080 [69] we run a four-task workload with each task pinned to one of its e6500 cores. Each task performs integer and floating operations at the core level over several large vectors so that data operated is fetched from main memory, causing frequent misses in all cache levels, and thus exercising several HEMs. For this experiment, as well as the remaining ones throughout this chapter, we run on baremetal to remove potential interference coming from the operating system (the specifics of our experimental framework are described in Section 3.1.2). We divide the experiment into several sub-experiments, in each of which we read 6 HEMs (the total number of PMC available in the T2080). Hence, reading all 262 HEMs requires 44 sub-experiments, each of which we repeat 100 times to capture the impact of noise on HEM readings. In all runs we focus on the HEMs for core 0. Each run finishes when the task in core 0 finishes.

Figure 6.1 shows the maximum relative variability observed for several HEMs, i.e. $var = (max - min)/min$, with bars in the figure sorted from higher to lower. Each bar is tagged (see bottom part of the figure) with the order of magnitude m of the value of each HEM in the experiment. For instance, for $m = 3$, $10^3 \leq hem^i < 10^4$. This information allows assessing the potential impact of the variability on execution time, whose magnitude for this experiment is tens of millions (10^7) of cycles. So is that for the number of committed instructions.

In the NXP T2080, the maximum duration of an event triggered by an instruction can be in the order of hundred cycles (10^2). Hence, HEMs below 10^4 arguably have low impact on performance. This, of course, is related to events not involving the execution of system software, e.g. a TLB miss, whose impact is not covered by multicore contention timing analysis but instead captured by the system-level timing analysis. We differentiate some cases for our experiments.

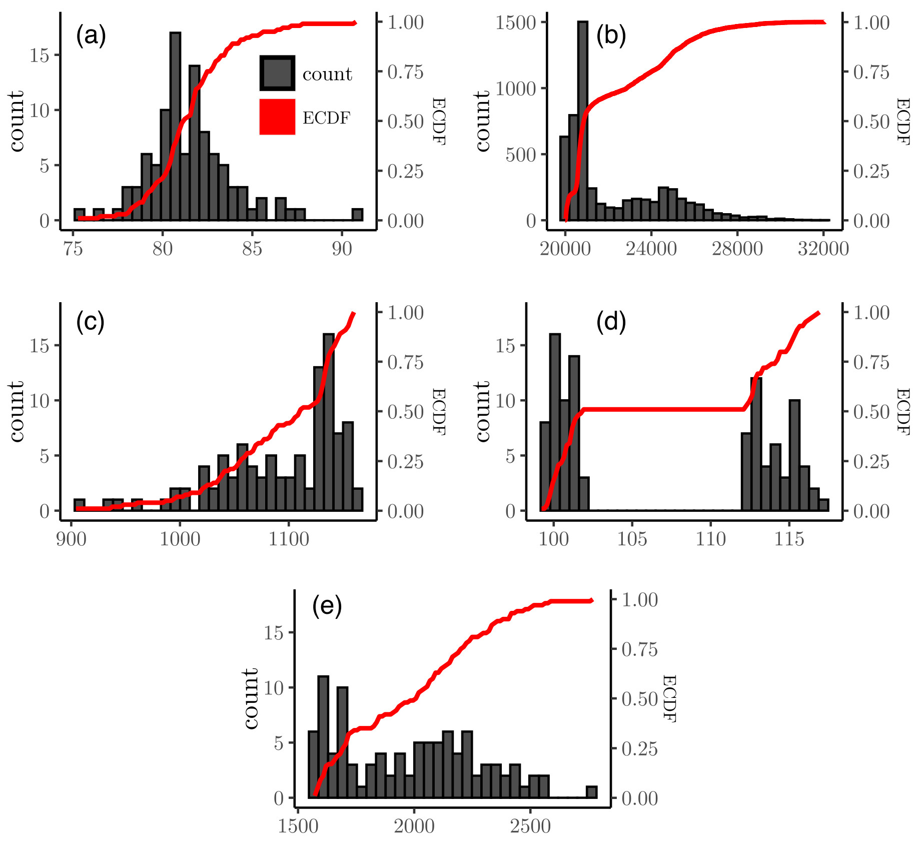


Figure 6.2: *Histogram and empirical CDF (ECDF) types: (a) Normal, (b) Concave, (c) Convex, (d) Clustered, (e) hard-to-fit.*

- **Relevant high variability**. Some HEMs present high variability while their magnitude is relevant, 10^4-10^7. These HEMs are the focus of our study as they can significantly impact the timing of the application and hence, the bounds that can be derived to it. In this category we find `PROCESSOR_CYCLES` with a variability of 45% from $3.6 \cdot 10^7$ to $5.2 \cdot 10^7$ (in other experiments the variability of this HEM reached 59%). 37 HEMs fall in this category if we set 1% as threshold for low-variability.

- **Irrelevant or low variability**. Other HEMs have low variability in absolute terms, thus having little impact on performance. There are 70 HEMs in this category, including the three on the left of Figure 6.1 whose variability is over 180% but their value is below 300, hence, insignificant with respect to the cycle count. Other HEMs, 5 in total for this experiment, while exhibiting values above 10^4, incurred less than 1% variability, with limited impact on performance.

- **Not exercised**. Finally, other HEMs, 150 in our case, were not exercised by the program under analysis making both the minimum and maximum value be zero. As the set of HEMs exercised can change across different experiments, the particular non-exercised HEMs will like vary. In fact, this is the motivation behind having different benchmarks in the experimental evaluation.

The observed HEM variability does not depend on the particular subset of HEMs that are enable/disabled when collecting observations. Interestingly, the 'low variability' category comprises HEMs presenting no variability. Those are related to the functional execution of the program capturing the number of completed instructions including SFX, CFX, store, load, stores, taken and non-taken branches. For instance, the total number of executed instructions (`INSTRUCTIONS_CMPLTD`) of the first task is exactly the same in all runs (15,646,749). This leads us to conclude that the observed variability does not come from the software that always performs the same function (e.g. it traverses always the same execution path), and instead the variability is induced by the hardware.

6.2.2 Distribution

Focusing on the *relevant high variability* HEMs, we identified their variable behavior falls into five main classes of distribution. These types are depicted in Figure 6.2 that shows the histogram (bars) and the cumulative distribution function or CDF (line) of observed values for one HEM in each category for illustrative purposes. The x-axis shows HEM value, the left y-axis the frequency of occurrence for the histogram (for a 500 observations sample), and the right y-axis the fraction of observations for the CDF.

a. **Normal**. HEMs in this category show a symmetric behavior that resembles a normal distribution.

b. **Concave**. The distribution resembles a uniform with leaning towards the smallest values, which gives a concave cumulative distribution function.

c. **Convex**. Distribution with the probability mass concentrated on the highest values of the distribution, giving a convex cumulative distribution function.

d. **Clustered**. HEMs in this category show a clustered behavior around two values or more values. Distributions with more than one clear mode also fall into this category.

e. **Hard to fit**. Finally, some distributions follow no obvious distribution (hard to fit category) apparently characterized by having two modes and a long tail.

Out of the 37 relevant HEMs in our experiment, their distribution is as follows: 2 (5.4%) Normal, 13 (35.1%) concave, 19 (51.4%) convex, 1 (2.7%) clustered, and 2 (5.4%) hard-to-fit. The case of the HEM `PROCESSOR_CYCLES` is particularly relevant for a two-fold reason. It is the main HEM used in timing analysis for making predictions, and it presents a hard to fit distribution (see Figure 6.2(e)). This HEM presents 59% variability from around $2.0 \cdot 10^7$ to $3.2 \cdot 10^7$.

6.2.3 Reasons Behind the Observed Variability

The T2080 implements a complex architecture with an aggressive core (the e6500), so some form of hardware-induced HEM variability is therefore expected. We have observed that the HEMs with relevant high variability capture the activity in a wide range of hardware units, from the (on-core) integer issue queue to the internal queues of the L2 cache. High variability can be due to the complex nature of the T2080 and its sources of multicore interference: specific hardware scheduling choices in the multiple shared queues and buffers in the core-to-L2 interconnect, internal to the L2, the CoreNet Coherence Fabric, and the memory controller, may lead to variable latencies for specific requests. Specific and controlled execution scenarios allow narrowing down the sources of execution time variability. As an example, we have performed some bare-metal experiments where all cores hit L2 cache sustainedly, with a task τ overlapping its full execution with the others. The intent is that interference occurs solely in the L2 cache. Variability observed across executions (up to more than 40%) could be attributed to minor initial processor state differences causing slight time shifts between L2 accesses across runs, and leading to cascade effects in L2 queues. In general, however, the limited information about the internal functioning of some of these resources, e.g. CCF, simply prevents identifying some of the reasons behind the observed variability. Also, as the programs used in this study typically perform the same activities repeatedly, contention for requests of a given core can stay repeatedly low or repeatedly high, leading to cumulatively high variability. Further, such systematic patterns may inadvertently switch from low to high contention scenarios (or vice versa) due to several reasons, such as the effects of loop control instructions in a program, which might alter systematic behavior inside the loop, as well as the impact of DRAM refresh operations, just to name some examples.

Authors in [171] perform a hardware analysis of several Intel architectures and formulate several hypotheses on the reasons behind various forms of under and over counting affecting some HEMs (retired instructions, branches, load/stores, floating point, etc). Extending this to modern MPSoCs confronts with the inclusion of large hardware IP blocks with limited description and the increasing number of HEMs monitoring events highly sensitive to such variability. Also, note that knowing the reasons behind such variability would help assessing whether the device allows some configurations under which the variability reduces. However, if the behavior causing the variability is intrinsic to

Table 6.1: *Main terms used in this work.*

Term	Definition
nh, np, nb, nr	number of HEMs, PMCs, sub-experiments, and runs
h^i	HEM with id i
b_j	j^{th} sub-experiment
$r_{j,k}$	run k of a given sub-experiment b_j
$m^i_{j,k}$	measured value for h^i in run $r_{j,k}$
M^i_j	set of measured values for h^i in b_j
$v^i_{j,k}$	range of variation of measured value $m^i_{j,k}$
h^a	anchor HEM
$sr_{j,l}$	Run of b_j with the l^{th} lowest value of h^a
SR_l	Concatenation of $sr_{j,l}$ for all $\{b_j\}_{j=1,\cdots,nb}$
SM_l	HEM readings in $sr_{j,l}$ for all $\{b_j\}_{j=1,\cdots,nb}$

the complexity/functioning of the device, a solution like HRM is still needed – whether or not the root can be explained.

6.3 Problem Formalization

Combining the discussion Section 2.4.1 about the disproportion between HEMs and PMCs with the analysis of high variability between measurements of the previous section, there lies a challenge to merge the information from different HEM readings into a single dataset as if they were measured together. In this section we will formalize the HEM merging problem.

We are interested in collecting the values of a set of relevant HEMs, $\mathcal{H} \ni \{h^1, h^2, \cdots, h^{nh}\}$, whilst a given program executes on the target platform in response to a given input (the main terms used in this chapter are listed in Table 6.1).

In an ideal scenario, all *nh* HEMs are collected at once on a single program execution without incurring the uncontrolled (platform or system level) jitter or variability that may arise across executions. Under such favorable conditions, we obtain a set of measurements (values) for each HEM in $h^i \in \{\mathcal{H}\}$ that cumulatively capture the activity performed by the program. This is referred to as Scenario 1 in Figure 6.3, in which the row shows the single execution, columns the HEMs and the cell their respective values.

In a more realistic scenario, program executions on the target platform are subject to *noise* so that in each execution the measured values for a given HEM h^i can potentially vary. Note that we use the term 'noise' to generically refer to the varying execution conditions across experiments, either due to different initial hardware and system software state (in this respect, our experiments are executed baremetal reducing the variability due to system software). We are not after quantifying such noise, but we just recognize that it is in general uncontrollable, beyond the measures we take in order to reduce it as shown in Section 3.1.2. To capture the impact of noise, several runs of each experiment need to be performed. The noise of the different runs is represented as different levels of grey in Figure 6.3 (Scenario 2). In this scenario, noise can occur but at least all HEMs can be read at once, so all HEMs in each run are exposed to the same noise. This makes possible to reason about their relationships for statistical inference.

In general, however, it is not possible to read all *nh* HEMs at once in a single execution as the number of HEMs that can be tracked simultaneously is determined by the number of available PMCs. Assuming our platform support *np* configurable PMCs[1] typically comparatively small with respect to the number of supported HEMs, with $np \ll nh$. For this reason, HEMs are necessarily collected in groups of at most *np* elements. Hence, to measure all HEMs for a given program we

[1]Without lack of generality, we assume there are no constraints on which specific HEM can be read from each PMC. Some processors exhibit such constraints, due to hierarchy of multiplexors to route HEM readings to a specific PMC. This scenario would just restrict which HEMs can be read in the same run, but would not affect HRM, as some HEMs (e.g. PROCESSOR_CYCLES) can be read along with any other group of HEMs.

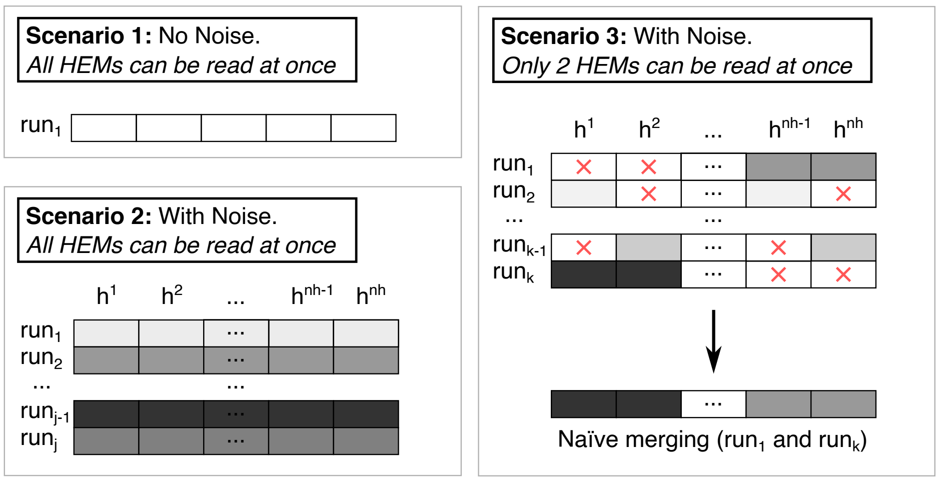


Figure 6.3: *Scenarios in HEM reading.*

must perform a set of at least $nb \geq \lceil nh/np \rceil$ sub-experiment $(b_1, \cdots, b_j, \cdots, b_{nb})$, each capturing the values of at most np distinct HEMs and cumulatively covering all HEMs.

To capture the variability in measured values, several runs of the same sub-experiment b_j are carried out, see Scenario 3 in Figure 6.3, with crosses showing the HEMs not read in a given run. In this case, we assume only 2 HEMs can be read in each run. Also, as shown at the bottom of Scenario 3, naively merging HEMs (from the first run and kth run in this case) results in merging HEMs values obtained under different noise levels, potentially resulting in inconsistent values that cannot be reliably used.

In this chapter, we address the challenge of merging the readings (measurements) for all $h^i \in \mathcal{H}$, each one measured several times in a different sub-experiment (Scenario 3) to obtain noise-consistent measurements for all HEMS (Scenario 2), preserving their relationships with execution time to favor timing analysis. Note that, noise-free HEM values (Scenario 1) are arguably hard to achieve, if at all possible, in MPSoCs. In particular, we aim at obtaining vectors with values for all HEMs under *similar* noise, as if all of them could have been read simultaneously in every single run.

6.4 HRM: a Technique to Merge HEMs

Table 6.2 introduces an example with the main inputs and outputs to be generated by any HEM merging approach. In particular, it shows the measurements made when the number of HEMs is $nh = 12$ and the number of PMCs is $np = 3$, hence being required $nb = 4$ sub-experiments. In the example, $nr = 5$ runs are performed per sub-experiment. On the left, it is reported the sub-experiment and run id. In the top part, the HEM id. We use $r_{j,k}$ to refer to the run k of sub-experiment b_j. We refer to measured value of HEM h^i in b_j and run k as $m^i_{j,k}$. In terms of outputs, a HEM merging mechanism must aim at producing a list of nr all-HEM readings (vectors) where each vector includes all HEMs. Each of the nr measurements of each HEM is placed exactly in one of those nr vectors. This is illustrated at the bottom of Table 6.2, where each run of each sub-experiment is merged with another run from each other sub-experiment so that each run is represented exactly once in the merged result. For instance, in the example, the 1st run of the first sub-experiment $(m^1_{1,1}, m^2_{1,1}, m^3_{1,1})$ is merged with the x'^{th} run of the 4th sub-experiment, and with one run of each other sub-experiment represented as the x^{th} run of the j^{th} sub-experiment.

6.4.1 Approach

Our approach, HRM, builds on non-parametric order statistics, which allows relating random variables based on the order of the sampled values of the variables, regardless of their distributions. In particular, HRM aims at merging the HEM measurements from different sub-experiments in such a way that the noise experienced by the different measurements is as similar as possible. HRM must also allow merging HEMs regardless of the distribution of the data to be merged. Non-parametric

Table 6.2: *Example with nh = 12, np = 3, nb = 4, a nr = 5.*

		h^1	h^2	h^3	$\cdots$	h^i	$\cdots$	h^{10}	h^{11}	h^{12}
	$r_{1,1}$	$m^1_{1,1}$	$m^2_{1,1}$	$m^3_{1,1}$						
	$\vdots$	$\vdots$	$\vdots$	$\vdots$						
b_1	$r_{1,k}$	$m^1_{1,k}$	$m^2_{1,k}$	$m^3_{1,k}$						
	$\vdots$	$\vdots$	$\vdots$	$\vdots$						
	$r_{1,5}$	$m^1_{1,5}$	$m^2_{1,5}$	$m^3_{1,5}$						
					$\ddots$					
	$\vdots$					$\vdots$				
b_j	$r_{j,k}$					$\cdots m^i_{j,k} \cdots$				
	$\vdots$					$\vdots$				
							$\ddots$			
	$r_{4,1}$							$m^{10}_{4,1}$	$m^{11}_{4,1}$	$m^{12}_{4,1}$
	$\vdots$							$\vdots$	$\vdots$	$\vdots$
b_4	$r_{4,k}$							$m^{10}_{4,k}$	$m^{11}_{4,k}$	$m^{12}_{4,k}$
	$\vdots$							$\vdots$	$\vdots$	$\vdots$
	$r_{4,5}$							$m^{10}_{4,5}$	$m^{11}_{4,5}$	$m^{12}_{4,5}$
						$\Downarrow$				
SM_1		$m^1_{1,1}$	$m^2_{1,1}$	$m^3_{1,1}$	$\cdots$	$m^i_{j,x}$	$\cdots$	$m^{10}_{4,x'},$	$m^{11}_{4,x'},$	$m^{12}_{4,x'}$
			$\vdots$			$\vdots$			$\vdots$	
SM_i		$m^1_{1,k}$	$m^2_{1,k}$	$m^3_{1,k}$	$\cdots$	$m^i_{j,y}$	$\cdots$	$m^{10}_{4,y'},$	$m^{11}_{4,x'},$	$m^{12}_{4,y'}$
			$\vdots$			$\vdots$			$\vdots$	
SM_{nr}		$m^1_{1,5}$	$m^2_{1,5}$	$m^3_{1,5}$	$\cdots$	$m^i_{j,z}$	$\cdots$	$m^{10}_{4,z'},$	$m^{11}_{4,x'},$	$m^{12}_{4,z'}$

order statistics, which resort to the order of data regardless of their distribution, allow relating runs across measurements through the use of an 'anchor' HEM, referred to as h^a, measured in all sub-experiments. HRM derives the relation between HEMs in different sub-experiments via their relation to h^a.

This is illustrated in the left side of Figure 6.4 that shows how the individual readings of h^i and h^j (m^i and m^j respectively) from different sub-experiments are related to those of the h^a in each sub-experiment referred to as. HRM provides the following properties. First, it preserves the distribution of each individual HEM. It also preserves the joint distribution between each HEM, h^i, and the anchor HEM, h^a (recall that the joint distribution between the HEMs read in the same sub-experiment is maintained). Finally, HRM estimates the most reliable joint distribution across HEMs in different sub-experiments. Next, we detail the procedure followed by HRM to provide those properties, followed by the mathematical foundation of the approach.

6.4.2 Procedure

The application process of HRM includes four main steps.

STEP ①. HRM starts by selecting the anchor HEM, h^a, that will be read in all sub-experiments. In each sub-experiment $np - 1$ PMCs are used to read different HEM. That is, from all available np PMC, HRM uses one of them in each sub-experiment b_j for the anchor, and the other $np - 1$ PMCs for other HEM. HRM approximates unobserved HEM relationships via their individual (observed) relationship with h^a. Thus, the selection of h^a is critically important as it determines how effective is HRM to merge HEMs for the problem under study. As the problem at hand relates to timing analysis, we chose h^a to be as relevant as possible to timing. In the case of the T2080, execution time is measured via the HEM `PROCESSOR_CYCLES`, and hence h^a = `PROCESSOR_CYCLES`.

STEP ②. After performing nr runs of each sub-experiment, HRM sorts the runs of each sub-experiment by h^a, from lowest to highest. As a result, each element in the sorted list for each

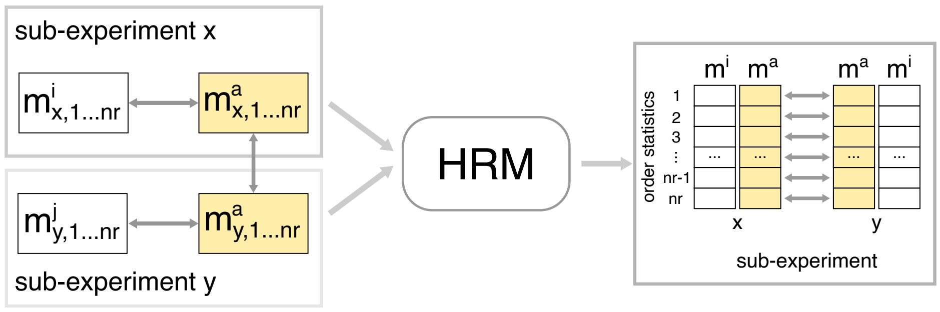


Figure 6.4: *Introduction to the HRM approach.*

sub-experiment will indicate an order statistic, with the k^{th} order statistic of a sample being its k^{th}-lowest value.

Each sub-experiment is characterized by a small fixed set of HEMs, limited by the number of PMCs available, np. Each sub-experiment in $\{b_j\}_{j=1,\cdots,nb}$ is represented by a set of nr runs of dimension np,

$$r_{j,k} : \left(m^a_{j,k}, m^{(np-1)(j-1)+1}_{j,k}, \cdots, m^{(np-1)(j-1)+(np-1)}_{j,k}\right).$$

The selection of nr should be based on prior knowledge of h^a random variable behavior. Without such knowledge, one must resort to $nr \geq 30$, as this is the minimum size to estimate the main properties of a distribution through the central limit theorem. Runs in each sub-experiment and across them, should be designed to ensure that they are independent and identically distributed to enable the probabilistic reasoning on which HRM builds. To achieve this property, we empty the processor state between runs (see Section 3.1.2). We assess it by performing statistical independence and identical distribution tests (see Section 6.5.2).

STEP ③. Once all sub-experiments are sorted based on h^a, we merge the different sub-experiments so that the k^{th} measurement in the list for h^i in a given sub-experiment is merged with the k^{th} measurement of h^j in another sub-experiment. Naturally, HEM measurements in the same run of the same sub-experiment remain at exactly the same position in the sorted list, so they remain together upon merging.

Let $sr_{j,l}$ be the run of sub-experiment b_j with the l^{th} lowest value of h^a. $sr_{j,l}$ is defined as $sr_{j,l} = r_{j,k}$ where $m^a_{j,k}$ is the l-lowest value in the set $\{m^a_{j,k}\}_{k=1,\cdots,nr}$. Finally, the concatenation produces a vector with completed representation of HEMs $SR_l = (sr_{1,l}, \cdots, sr_{j,l}, \cdots, sr_{nb,l})$ for each $l = 1, \cdots, nr$. Since the l-lowest value of $m^a_{j,k}$ is well-defined, we can assume that for each $j = 1, \cdots, nb$, $m^a_{j,k} \leq m^a_{j,k+1}$ for all $k = 1, \cdots, (nr-1)$. Therefore, $m^a_{j,k}$ is the $(k : nr)$-order statistic of the sample of size nr, $\{m^a_{j,k}\}_{k=1,\cdots,nr}$. HRM merges the values read in the same ordered run across all sub-experiments, referred to as SM_l. For each SM_l, HRM produces one reading for each HEM and nb readings for h^a.

STEP ④. After merging, we compute the summarized order statistics for h^a. In particular, we compute the quantiles of the distribution of all values of h^a across all sub-experiments so that we obtain exactly nr quantiles, i.e. one for each row of our merged list of HEM values. The resulting array yields $\hat{m}^a = \text{quantile}(0, \cdots, k/(nr-1), \cdots, 1)$, where $k = 0, \cdots, (nr-1)$. HRM estimates the nr equal spaced quantiles of h^a using the sample of size $(nr \cdot nb)$ obtained from joining all h^a values.

6.4.3 Quantile Estimation

Several methods for quantile estimation can be considered. Let $\{x_{(k)}\}_{k=1,\cdots,n}$ be an ordered sample of size n. In general, a method for quantile estimation corresponds to weighted averages of consecutive order statistics. Given fixed values for a function γ and a constant m, the p-quantile is defined by $q(p) = (1-\gamma(j,m))x_{(j)} + \gamma(j,m)x_{(j+1)}$, where $(j-m)/n \leq p < (j-m+1)/n$, $x_{(j)}$ is the $(j : n)$–order statistic. We consider a continuous representation of quantile estimation with $\gamma(j,m) = p \cdot n + m - j$ and $m = 1 - p$, which is equivalent to do linear interpolation between the points $\{(p_k, x_{(k)})\}$ where p_k attempts to estimate the mode of $F(x_{(k)})$. Then $q(p)$ is a continuous function of p and $p(k) = (k-1)/(n-1)$. We refer the interested reader to [88] for a review of programming quantile estimation.

6.4.4 Correlation Boundary

HRM produces a solution that preserves the observed information and reliably builds unobserved information by preserving joint distributions. That is, HRM preserves the correlation across HEMs. In particular, HRM describes the relationship between the expected value of a target HEM, the anchor h^a, and the values observed for a different HEM h^i. The relationship across different HEMs, namely h^i and h^j, not observed together, is built therefore through h^a. HRM builds such relationship by estimating the covariance matrix across all HEMs. The covariance matrix can be used because each h^i is a random variable with at least 4 *finite moments*. In our case, each HEM has infinite finite moments since all HEM values are bounded, i.e. they count finite events per cycle during a finite interval, since the measurement starts until the measured value is collected. Therefore, each HEM value is a bounded number, thus guaranteeing the existence of infinite finite moments. By being random variables with finite moments, we can describe the relationship across expected values of HEMs through a multivariate normal distribution based on the central limit theorem, which ultimately ensures the existence of the covariance matrix that characterizes the relationship between the expected values of HEMs asymptotically.

In particular, correlation across HEMs is based on Pearson correlation coefficient [66]. It can be obtained via a Least-Squares fit, where a value 1 represents a perfect positive relationship, -1 a perfect negative relationship, and 0 the absence of any apparent relationship across variables. Let X and Y be random variables, and denote by $\text{cor}(X, Y)$ the Pearson correlation, obtained as $\frac{\text{cov}(X,Y)}{\sigma_x \sigma_y}$, where σ and *cov* describe the variances and covariance, respectively.

Lemma. Let (Y, X_1, X_2) be a random vector with multivariate standardized normal distribution. Then, the correlation between X_1 and X_2 is in the interval

$$\rho_1 \rho_2 \pm \sqrt{1-\rho_1^2}\sqrt{1-\rho_2^2},$$

where $\rho_i = \text{cor}(Y, X_i)$ for $i = 1, 2$.
Proof. Let Σ be the covariance matrix the joint distribution between the random variables, Y, X_1 and X_2. Σ can be described as the correlation matrix

$$\begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho \\ \rho_2 & \rho & 1 \end{pmatrix}. \tag{6.1}$$

where ρ cannot be arbitrarily set between $[-1, 1]$, since the matrix must be positive semidefinite. A Hermitian matrix is positive semidefinite if and only if all principal minors are non-negative. Building on Silvester's criterium, only the minors defined by submatrices starting from the upper left corner need being checked. The 2-by-2 submatrix, $\left(\begin{smallmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{smallmatrix}\right)$, is trivial, and the 3-by-3 matrix produces the result to prove.

Note that, by operating the result of the multivariate standardized normal distribution with the corresponding μ and σ of the random variables of the HEMs whose joint distribution we are studying, we can directly obtain the result for the multivariate non-standardized normal distribution. Moreover, since the random variables studied (the HEMs) have 4 finite moments (in fact they have infinite moments), and based on the central limit theorem, the Lemma guarantees that observed values converge asymptotically to the expected values.

Building on the Lemma, we can prove that HRM guarantees the three properties described in Section 6.4.1.

Theorem Let H be the joint distribution of all HEMs, and assume the set of ordered sub-experiments as shown in the example in Table 6.2. Let be $\{\hat{m}^a_k\}_{k=1,\cdots,nr}$ the set of nr equal spaced quantiles of h^a from the sample $\{m^a_{j,k}\}_{j,k}$ of size $(nb \cdot nr)$. Consider the complete (merged) vector with all HEMs defined as:

Table 6.3: *HEMs with observed relevant variability.*

Name	ID
CYCLES_LSU_SCHE_STALLED	1
CYCLES_LSU_ISSUE_STALLED	2
BLINK_REQUEST	3
L2_MISSES	4
L2_DEMAND_ACCESSES	5
L2_ACCESSES	6
L2_STORE_ALLOCATES	7
L2_DATA_MISSES	8
L2_RELOADS_FROM_CORENET	9
L2_SNOOP_HITS	10
L2_SNOOP_PUSHES	11
STALL_FOR_RLT_CYCLES	12
STALL_FOR_WDB_CYCLES	13
BIU_MASTER_REQUESTS	14
BIU_GLOBAL_REQUESTS	15

$$\Big(\hat{m}_k^a, m_{1,k}^1, \cdots, m_{1,k}^{(np-1)}, \cdots$$
$$m_{j,k}^{(np-1)(j-1)+1}, \cdots, m_{j,k}^{(np-1)(j-1)+(np-1)}, \cdots$$
$$m_{nb,k}^{(np-1)(nb-1)+1}, \cdots, m_{nb,k}^{(np-1)(nb-1)+(np-1)}\Big),$$

for each $k = 1, \cdots, nr$. Then, the empirical joint distribution described by the complete vectors complies the (formalized) properties of HRM:

- *Property 1.* Preserves the marginal distribution of H for all HEMs.
- *Property 2.* Preserves the joint distribution across HEMs in the same sub-experiment.
- *Property 3.* Estimates, with minimum error on correlation, the joint distribution between HEMs in different sub-experiments.

Proof. *Property 1*: The marginal distribution of all HEMs but h^a is preserved, since no modifications are produced in the observed values of those HEMs – they are just sorted. Only h^a is modified, since it is replaced by the order statistics. As its distribution has infinite moments, replacing h^a by its order statistics of a larger sample, leads to a higher amount of information (i.e. $nb \cdot nr$ values instead of nr), improving the sample and hence, preserving the marginal distribution of h^a.

Property 2: The re-ordering procedure preserves measurements of different HEMs in the same sub-experiment together. Therefore, their joint distribution is preserved identical.

Property 3: Regarding the joint distribution of HEMs in different sub-experiments, HRM estimates such joint distribution for each pair of HEMs. Note that, since those pairs of HEMs are never observed in the same sub-experiment, data collected provides no information about their joint distribution. The only relation across sub-experiments is had through h^a, which is observed in all of them, so the joint distribution to be estimated needs to preserve this common relation. Based on the Lemma, such relation is preserved if the estimated correlation for describing the real joint distribution of two HEMs in different sub-experiments is in the interval $\rho_1\rho_2 \pm \sqrt{1-\rho_1^2}\sqrt{1-\rho_2^2}$, where ρ_1 and ρ_2 are the correlation between each of those two HEMs and h^a. Since there is no additional information about the actual correlation between those two HEMs, any value in the interval is equally probable. Thus, the correlation value proposed by HRM is $\rho_1\rho_2$, since this is the value that minimizes the absolute error with respect to the real value.

6.4.5 Matrix Completion Techniques

HRM aims at merging *actual observations* rather than filling missing values with synthetic data. The latter, which may be realized with Matrix Completion methods [83, 136], as discussed before in Section 2.4.2, is not appropriate in our case. This is so because Matrix Completion requires that values in each row and column belong to a different distribution, which is not our case, since each column is a different HEM with its own distribution. As a consequence, the use of Matrix Completion methods for our problem leads to inadequate value distributions where, for instance, the mean and standard distribution of the synthetic data for all HEMs is extremely different from those for actual observations. For instance, in our experiments, the mean for synthetic data is $\approx 20x$ smaller than that of real data, whereas the standard deviation is between $0.36x$ and $5x$ that of real data.

6.5 Experimental Evaluation

6.5.1 Validation Methodology

For the experimental evaluation of this work we resort to the microbenchmarks run in the NXP T2080 Reference Board described in Section 3.1.2 and represented in Figure 3.2. The benchmarks we use to generate validation data are the microbenchmarks described in Section 3.1.2, which are designed to stress different cache levels to conform a representative set of cases. In order to create a realistic scenario for the validation, we use the set of 16 workloads described in Table 3.1, where we run one benchmark per core and the readings are performed from *core0*.

The validation of any HEM merging methodology is complex on real hardware as we do not have the noise-free value for each HEM (h^i), as explained in Section 6.3. This prevents us from directly comparing the estimated value for each HEM with its corresponding noise-free value. Thus, we can only evaluate HRM comparing the correlation for the HEM merged with HRM against the real – measured – correlation.

In order to evaluate estimated and real correlations, first, for each workload, we perform 100 runs for each of the 53 = 261/5 sub-experiments. Hence, we collect readings for all 262 HEMs with 5 HEMs plus h^a read in each sub-experiment[2], except the last group (sub-experiment) that only includes 1 HEM and the h^a.

We validate HRM for 15 HEMs having high relevant variability, see Table 6.3. To that end we on purpose place those HEMs in different groups so that their mutual correlation is not observed in the data used for HRM.

For each of the 120 pairs[3] of HEMs we estimate their correlation $\hat{\rho}^{i,j}$, after merging them with HRM. We also collect 100 runs for a set of experiments in which those 15 HEMs and the anchor are observed in the same group. Thus, for each pair of HEMs (h^i and h^j), as well as h^a, we obtain their actual correlation $\rho^{i,j}$ from those measurements. This allows us comparing their real correlation $\rho^{i,j}$ with the estimated correlation after merging with HRM $\hat{\rho}^{i,j}$. In particular we measure the absolute distance (difference) between $|\rho^{i,j} - \hat{\rho}^{i,j}|$, so that the maximum difference obtained for a pair of HEMs is 2. This happens when the estimated correlation is 1 (or -1) and the real one is -1 (or 1).

6.5.2 Independence and Identical Distribution

HRM builds on these statistical properties for h^a, `PROCESSOR_CYCLES`, to apply order statistics. In practice, this holds since all values of h^a have been collected from the repeated execution of the same workload, with the same inputs, and enforcing the same hardware and software state as much as it can be controlled. We have further evaluated these properties quantitatively. We performed an ANOVA test [63] to assess identical distribution of `PROCESSOR_CYCLES` across sub-experiments. The result of the test is a p-value $p = 0.57$, so the test is not rejected comparing the law on the expected value of `PROCESSOR_CYCLES`, and tells us that the noise is identically distributed across sub-experiments. We assess independence within each sub-experiment with a Ljung-Box test [108] with $lag = 10$. With a significance level $\alpha = 0.05$, independence is not rejected in 96% of the sub-experiments, so

[2]For the sake of convenience, we refer to the HEM read in the same sub-experiment as being in the same (HEM) group.
[3]All possible pairs with the 16 HEMs analyzed (the 15 relevant and h^a).

measurements can be regarded as independent since the expectation is that the test is not rejected by a fraction of the tests matching $1 - \alpha$. Hence, we can use the order statistics for `PROCESSOR_CYCLES` after the merge as part of HRM because the noise is the same across sub-experiments and there is no dependence across values read.

6.5.3 Correlation Between HEMs

In order to have reliable correlation estimates, we use the percentile bootstrap method [49, 60] with the following methodology. We first compute bootstrap samples of size $n = 50$ for all sub-experiments; we compute the correlation between all pairs of HEMs; we then repeat $p = 100$ times those two steps and store the estimates of the correlation; for each pair of HEMs we have 100 estimates and we take the mean of those 100 values, which will be our reference estimation. We can also obtain the confidence interval for those 100 values for completeness.

For reference, we consider two other merging methods, referred to as *unsorted* and *sorted* respectively. The unsorted method simply concatenates results of different sub-experiments in the very same order they are collected, without analyzing any type of relationship between HEMs. The sorted method, instead, sorts the values collected for each HEM from lowest to highest merging in the same vector those in the same relative position for each HEM, so the lowest value for each HEM form a vector, the second lowest for each HEM another vector, and so on and so forth.

We have chosen workloads 1, 2, 5 and 10 due to their high variability and different spectra of measured values and correlations between HEMs in order to provide a representative set of cases. Figure 6.5 shows the correlation distance for the chosen workloads. Each point represents the absolute difference between the estimated correlation for each method and the real correlation obtained measuring those HEMs in the same group. We order all HEM pairs for each method from lowest to highest correlation difference. As shown, the differences between the observed and estimated correlation for HRM is consistently lower than for the other methods, thus reflecting its higher accuracy. While the input data for all methods does not include direct observations of the real correlation and hence, such information is missing in statistical terms, HRM successfully recovers part of this information through their individual correlations with h^a, which is effectively observed.

The (naive) unsorted method is obviously poor and achieves good correlation only in some cases by chance.

The sorted method performs very well for those pairs of HEMs where both HEMs have strong positive correlation, since sorting them precisely joints correlated values. However, in many cases such correlation is either indirect or weak, which makes the sorted method particularly inaccurate leading to the highest discrepancies with respect to real correlations.

In the case of HRM, correlation is precisely estimated for those HEMs with significant correlation with h^a, since their mutual correlation is preserved with a probability matching the product of their individual correlations with h^a. However, if their individual correlation with h^a is weak at least for one of the HEMs, their mutual correlation will be mostly lost, and the estimated correlation will approach 0. However, despite that, a key advantage of HRM is that joint correlation across HEMs is lost if and only if at least one of them is not meaningfully correlated with h^a. Instead, those correlations that matter for timing in our case, are preserved, as opposed to the other methods (unsorted and sorted), which preserve correlation for arbitrary pairs of HEMs, not for those necessarily correlated with timing (i.e. h^a).

For the rest of the 12 workloads we cannot show such detailed results as for workloads 1 and 2. Instead we present a summarized analysis of the three methods for all workloads is shown in Table 6.4, in the form of the mean squared error (MSE). The MSE is the average of the squared errors, it is specifically computed as $\frac{1}{n}\sum_{i,j=1,\cdots,nrh}^{i>j}(\rho^{i,j} - \hat{\rho}^{i,j})^2$, where nrh is the number of relevant HEMs, and n is the number of pairs $\binom{nrh}{2}$. As it can be seen, HRM shows to be the most accurate method sustainedly, and its accuracy is only relatively lower for Workload 4 since correlations with h^a in this workload are relatively weak in general.

Correlation with the anchor. As stated, HRM aims at preserving the relationship between each HEM and the anchor. While such correlation is highly preserved by observing each HEM with h^a,

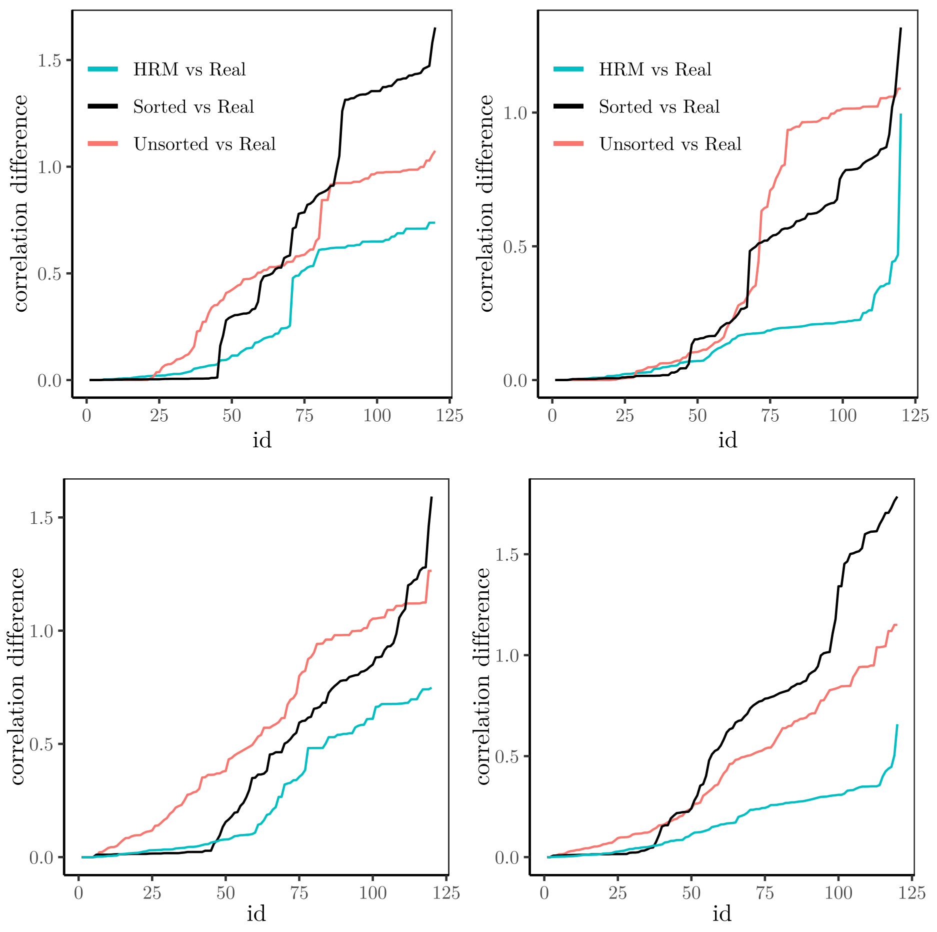


Figure 6.5: *Correlation Difference for each HEMs pair for workload 1 (top-left), workload 2 (top-right), workload 5 (bottom-left), and workload 10 (bottom-right).*

HRM reduces the number of observations of h^a in each merged vector (nb, one for each run of each sub-experiment merged) by applying order statistics. Thus, only 1 HEM out of the $np - 1$ in each merged vector preserves the actual value observed for h^a in its run, whereas the other $np - 2$ have a different h^a value, which may have an effect on the correlation between HEMs and h^a. However, this effect is expected to be tiny. We assess this quantitatively in Figure 6.6 for workloads 1 and 2, where we show the estimated correlation (blue lines), the 95% confidence interval (red lines) and the real correlation (black dots). As expected, correlation is estimated with very high precision. We have observed this very same effect for all workloads and all pairs of HEMs, so we omit those data due to lack of further insights and of space.

6.5.4 Overheads

The HRM algorithm has very low computation requirements. To process the data of the experiments we performed in the T2080, the `R` implementation of HRM required less than 38 milliseconds on a Dell latitude e7490 laptop.

HRM requires nr runs for each of the nb sub-experiments, so a total of $nb \cdot nr$ runs. For instance, to read all 262 HEMs, HRM required 53 sub-experiments to collect 5 different HEMs and h^a in each sub-experiment. Each sub-experiment was executed $nr = 100$ runs, thus above the minimum number of 30. Values for each workload were obtained in around 10 minutes. Note that, given that real-time programs usually last in the order of milliseconds, so few thousands of runs may only take up to few minutes in general.

Table 6.4: *Mean squared error of the merging methods.*

Workload	HRM	Sorted	Unsorted	Workload	HRM	Sorted	Unsorted
1	0.18	0.67	0.39	**9**	0.10	0.45	0.14
2	0.04	0.24	0.38	**10**	0.05	0.69	0.25
3	0.17	0.51	0.44	**11**	0.22	0.28	0.33
4	0.4	0.48	0.46	**12**	0.14	0.43	0.33
5	0.15	0.37	0.45	**13**	0.20	0.34	0.40
6	0.36	0.50	0.38	**14**	0.25	0.35	0.48
7	0.12	0.34	0.12	**15**	0.09	0.55	0.17
8	0.17	0.19	0.30	**16**	0.22	0.21	0.32

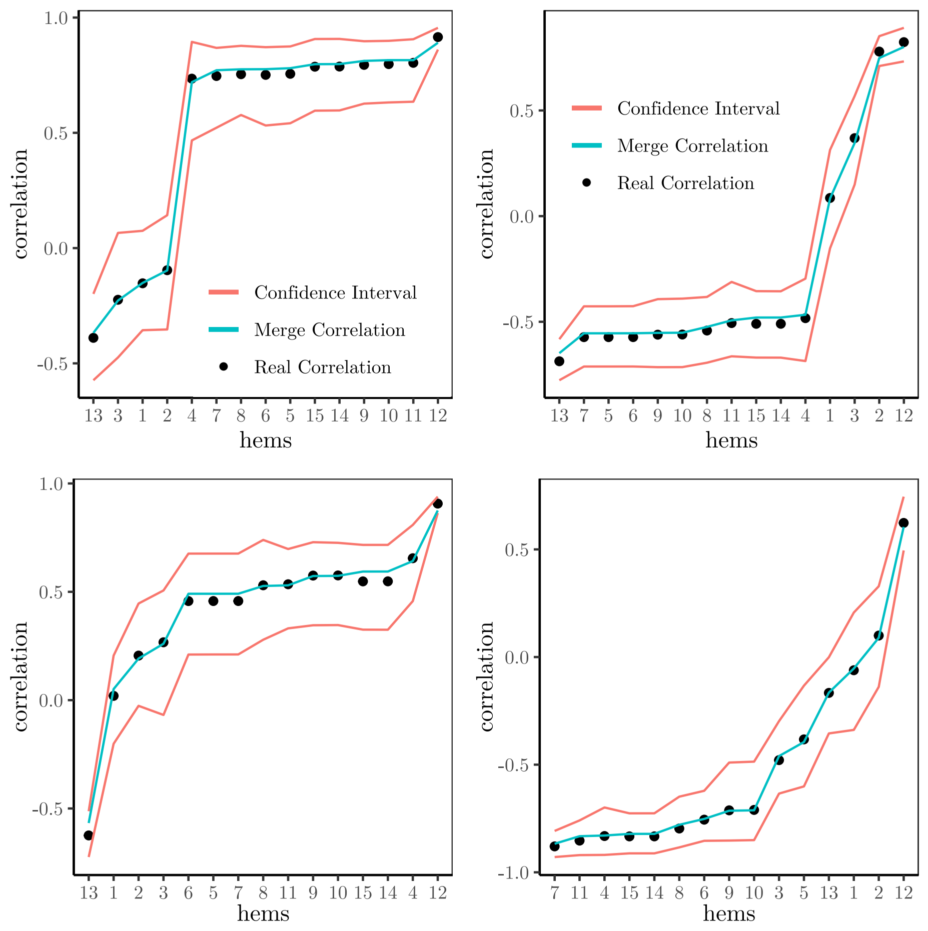


Figure 6.6: *Correlation between relevant HEMs and* PROCESSOR_CYCLES *before and after merging for workload 1 (top-left), workload 2 (top-right), workload 5 (bottom-left), and workload 10 (bottom-right).*

6.6 Summary

The V&V methodologies for MB(P)TA can be aided with the monitoring information at system operation from the hardware. In this work we show how, despite this information being useful, the reduced number of PMCs does not allow to obtain sufficient information from HEMs at once. Furthermore, the high variability of the execution conditions makes merging data very challenging. Our first proposal in this thesis to collect and merge information from HEMs as if they were read at once is called HEM-Reading Merging (HRM). In this method, we match the execution conditions of each run with the *anchor* HEM which is present at every sub-experiment. With the use of order statistics, we can sort the sub-experiments by the *anchor* HEM and preserve the relationships between the *anchor* and the rest of the HEMs after merging.

Chapter 7

Merging Hardware Event Monitor Data for Complex MPSoCs Using Copulas

7.1 Introduction

In the previous chapter, we proposed a methodology called HRM to reliably join HEMs from multiple runs with different levels of noise, into one single dataset. The strength of HRM resides in its simplicity of application and experimental setting. However, HRM does not come without limitations. In this chapter, we analyze those limitations and propose another methodology that fulfills the same purpose and overcomes HRM's limitations.

HRM Analysis. The approach followed by HRM consists in reading an anchor HEM, h^a in each sub-experiment along with $np - 1$ other HEMs, without repeating any of the other $nh - 1$ HEMs across sub-experiments. See Figure 7.2 for a visual explanation of HRM with two sub-experiments, with the first column representing the anchor. Then, HRM merges all HEMs into complete HEM vectors by preserving their correlation with the anchor h^a perfectly, since all HEMs are read along with h^a in one sub-experiment. That is, for any HEM h^i, the correlation ρ_{ai} is fully preserved. In fact, if two HEMs, h^i and h^j, where $i \neq a$ and $j \neq a$, are read in the same sub-experiment, they will be merged together in the merged HEM vector and hence, their pairwise correlation ρ_{ij} will also be fully preserved. However, no action is taken to preserve the correlation among any other pair of HEMs read in different sub-experiments. In the diagram shown in Figure 7.1, and focusing on a specific HEM (e.g. h^2) this implies that correlations ρ_{21} and ρ_{23} are fully preserved, whereas correlations ρ_{24}, ρ_{25}, ρ_{26}, and ρ_{27} are not. Since those HEMs in different sub-experiments are placed in the same merged HEM vector through their relation with h^a, as discussed before, their pairwise correlation is only preserved to a limited extent dictated by ρ_{ai} and ρ_{aj}. For instance, in our example, correlation ρ_{24} is preserved as much as determined by the product $\rho_{21} \cdot \rho_{41}$. If such product is low, then HRM will preserve ρ_{24} poorly, and increasing the number of runs for each sub-experiment cannot cure such limitation.

Formalization. When two strongly correlated HEMs, h^x and h^y, with correlation ρ_{xy}, are read in different sub-experiments, then, given an anchor HEM h^a, HRM preserves ρ_{xy} correlation only partially $\rho_{xy}^{factor} = \rho_{ax} \cdot \rho_{ay}$. Hence, under HRM the actual correlation preserved is $\rho_{xy}^{hrm} = \rho_{xy}^{factor} \cdot \rho_{xy}$. Unless both ρ_{ax} and ρ_{ay} are very high (e.g. close to 1.0), the degree of joint correlation preserved drops significantly and their actual correlation in the merged HEM vector can be drastically different to the real one. For instance, if $\rho_{xy} \approx 1.0$, but $\rho_{ax} \approx 0.7$ and $\rho_{ay} \approx 0.7$, then $\rho_{xy}^{factor} < 0.5$ and hence, $\rho_{xy}^{hrm} < 0.5$, largely below $\rho_{xy} \approx 1.0$.

Such limitation is intrinsic to the way HRM collects sub-experiments, since, in general the correlation across HEMs in different sub-experiments is not observed (as HRM observes only the relations of

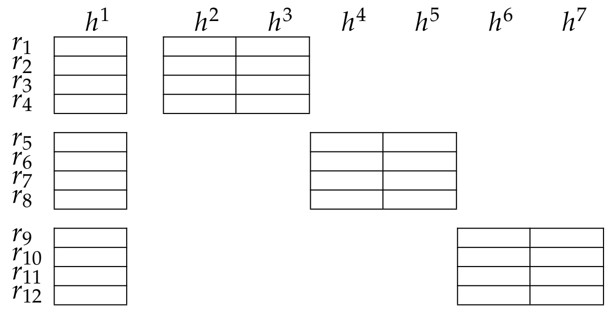


Figure 7.1: *Runs collected with HRM when $np < nh$. In particular, $np = 3$ and $nh = 7$ in the example.*

each HEM with the anchor HEM). As a result, even if HRM collected more runs per sub-experiment or even different sub-experiments, it would still be unable to get rid of its intrinsic limitation.

Summary. With HRM the correlation between each HEM and the anchor is preserved with almost perfect accuracy in the merged HEM vector. The number of sub-experiments needed by HRM, nb_{hrm} is typically very close to the minimum possible number of sub-experiments nb_{min}. However, in general, HRM does not collect any data to retrieve the degree of correlation between HEMs in different sub-experiments. Therefore, the merged HEM vector may completely lose the correlation among strongly correlated HEMs read in different sub-experiments.

In this chapter we propose MUlti-Correlation HEM reading and merging (MUCH) approach. MUCH overcomes the limitations of HRM with a completely different strategy to collect HEM values and using multivariate Gaussian distributions to merge HEM values, drastically improving the accuracy of the merged HEM vectors for all pairs of HEMs at once.

7.2 MUCH: Multi-Correlation HEM Reading and Merging

7.2.1 HRM Analysis

MUCH, builds upon observing all pairwise correlations across HEMs. Then, MUCH merges values from the different HEMs into complete HEM vectors preserving all pairwise correlations *simultaneously*. For instance, given 4 HEMs h^1, h^2, h^3 and h^4, MUCH organizes their values into complete HEM vectors so that preserving some correlations (e.g. all correlations with h^1, $\langle h^1, h^2\rangle$, $\langle h^1, h^3\rangle$ and $\langle h^1, h^4\rangle$) does not impact negatively the other correlations (e.g. $\langle h^2, h^3\rangle$, $\langle h^2, h^4\rangle$ and $\langle h^3, h^4\rangle$). Therefore, MUCH's challenge is putting all HEM values into complete vectors so that all pairwise correlations observed across HEMs *separately* are preserved *simultaneously* after merging.

Interestingly, by observing all pairwise HEM correlations one does not overcome the problem of preserving those correlations. It is still necessary to devise a proper method to merge all the observed readings into a complete dataset while keeping the correlations as close as possible to the ideal scenario. For instance, in the example above, if we merge all HEMs preserving their perfect correlation with h^1, e.g. sorting by h^1 values in all pairs $\langle h^1, h^2\rangle$, $\langle h^1, h^3\rangle$ and $\langle h^1, h^4\rangle$, and merging directly the rows keeping any of the values of h^1 in each row, we would keep perfect correlation between h^1 and each other HEM. However, we would likely degrade significantly the correlation of the pairs $\langle h^2, h^3\rangle$, $\langle h^2, h^4\rangle$ and $\langle h^3, h^4\rangle$. The challenge addressed by MUCH is exactly this one: *how much do I have to "sacrifice" each pairwise correlation so that, when putting all HEMs together, each pairwise correlation is still close to the observed one?*

To observe all pairwise correlations across HEMs, sub-experiments need to be arranged so that all pairs of HEMs are observed together in at least one sub-experiment. This is illustrated in Figure 7.3 for the same example discussed for HRM, with $np = 3$ and $nh = 7$. As shown, for instance, h^2 is observed in the first sub-experiment (r_1-r_4) together with h^1 and h^3, in the fourth sub-experiment (r_{13}-r_{16}) together with h^4 and h^6, and in the fifth sub-experiment (r_{17}-r_{20}) together with h^5 and h^7. This allows *observing* all pairwise correlations, and thus, organizing observed values into merged HEM vectors where $\hat{\rho}_{ij} \sim \rho_{ij}$ for each pair of HEMs h^i and h^j.

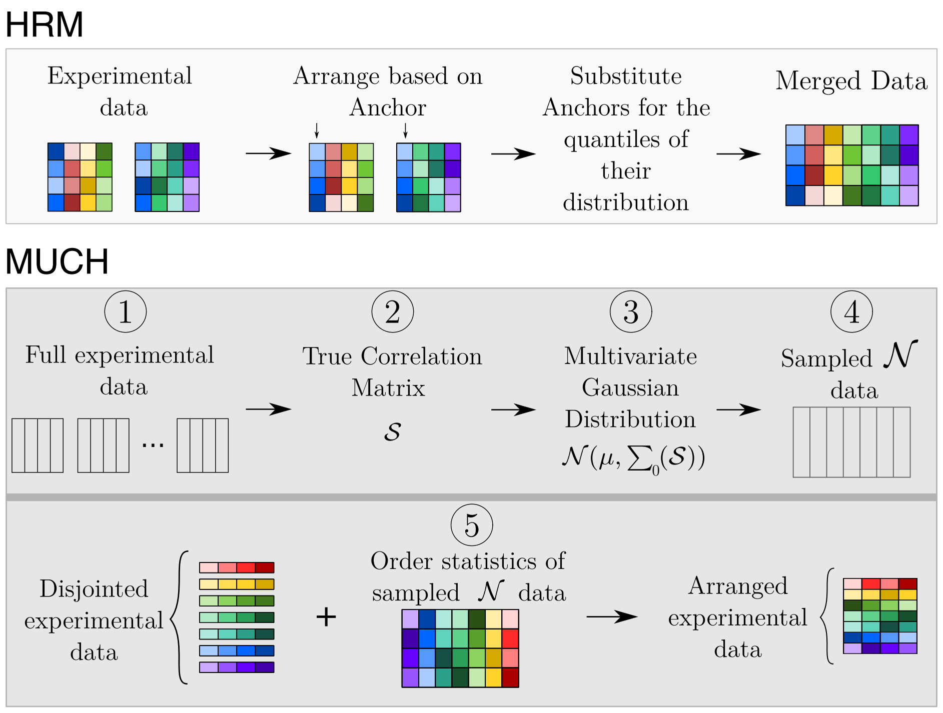

HRM
Experimental data
Arrange based on Anchor
Substitute Anchors for the quantiles of their distribution
Merged Data
MUCH
1
Full experimental data
2
True Correlation Matrix
$\mathcal{S}$
3
Multivariate Gaussian Distribution
$\mathcal{N}(\mu, \sum_0(\mathcal{S}))$
4
Sampled $\mathcal{N}$ data
5
Disjointed experimental data
Order statistics of sampled $\mathcal{N}$ data
Arranged experimental data

Figure 7.2: *Diagrams of HRM and MUCH methodologies. On both diagrams the colors represent the order statistics of each column, i.e. lighter colors represent lower values while darker colors represent higher values.*

Given that we have observed all pairwise HEM correlations, MUCH generates a nh-dimensional model of the HEMs by building on MVGD. Then, by organizing HEM values read according to that model, MUCH generates merged HEM vectors where all pairwise HEM correlations are preserved simultaneously with high accuracy, as detailed in next subsection.

Also MUCH requires a higher number of sub-experiments than HRM to observe all pairwise HEM correlations. Such number increases with the number of relevant HEMs (nh) and whenever the number of PMCs (np) decreases. However, as shown later in the evaluation section, under similar number of runs for HRM and MUCH, i.e. $nb_{hrm} \cdot nr_{hrm} \approx nb_{much} \cdot nr_{much}$, MUCH provides higher accuracy. Note that in that case, typically we have $nb_{hrm} < nb_{much}$ and $nr_{hrm} > nr_{much}$.

7.2.2 Mathematical Approach

By virtue of the Central Limit Theorem, we can model the expected value of each HEM as a Gaussian distribution. This can be extended to multiple variables (e.g. the nh HEMs of interest), which are regarded as a MVGD with nh dimensions modeling the expected value of each HEM.

Definition. Let us consider a set of l i.i.d. Gaussian random variables $Z \sim \mathcal{N}(0, 1)$. The covariance matrix for Z is the identity matrix I_l with expected value 0. Now, let A be a $k \times l$ matrix and μ be a k-vector, both with real finite coefficients. Then, $X = AZ + \mu$ has a MVGD. The expected value of X is μ and the covariance matrix is $\Sigma = AA^T$. The usual way of writing the MVGD is $X \sim \mathcal{N}(\mu, \Sigma)$ [176].

The computation of this model requires the expected value and the variance of each HEM, μ and σ^2, and the covariance matrix Σ, where the latter can be obtained from the correlation matrix S. In particular, the covariance of variables X and Y, σ_{XY}, can be obtained as follows, where ρ_{XY} is their correlation.

$$\sigma_{XY} = \rho_{XY} \cdot \sigma_X \cdot \sigma_Y.$$

The measurement collection procedure allows obtaining some relevant information.

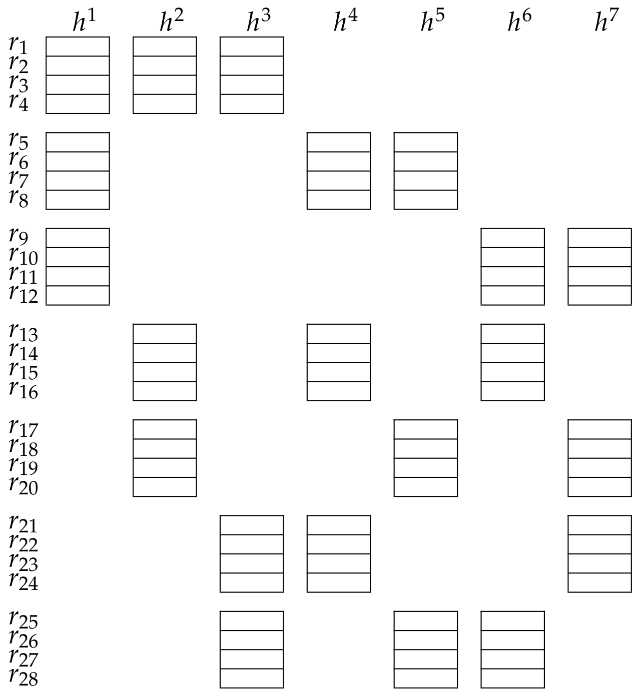

h^1
h^2
h^3
h^4
h^5
h^6
h^7
r_1 r_2 r_3 r_4
r_5 r_6 r_7 r_8
r_9 r_{10} r_{11} r_{12}
r_{13} r_{14} r_{15} r_{16}
r_{17} r_{18} r_{19} r_{20}
r_{21} r_{22} r_{23} r_{24}
r_{25} r_{26} r_{27} r_{28}

Figure 7.3: *Runs collected with MUCH when $np < nh$. In particular, $np = 3$ and $nh = 7$ in the example.*

- For each h^i, the measurements collected in each sub-experiment allow obtaining a grouped sample, that is $\{h^i\}$, which includes all measurements of h^i across all sub-experiments.
- For each h^i, the values of $\hat{\mu}_i$ and $\hat{\sigma}_i$ obtained from $\{h^i\}$.
- For each pair of HEMs h^i and h^j, we obtain $\hat{\rho}_{ij}$, which is the empirical counterpart of ρ_{ij}. $\hat{\rho}_{ij}$ is obtained from the sub-experiment where both h^i and $h^j$are observed together.

Therefore, since we have $\hat{\sigma}_i$, $\hat{\sigma}_j$ and $\hat{\rho}_{ij}$, we can also obtain their empirical covariance $\hat{\sigma}_{ij}$. Then, we can produce the empirical correlation matrix $\hat{S} \sim S$, and the corresponding (empirical) covariance matrix $\hat{\Sigma}$. Once we have constructed this information, we have a model for how each HEM in the experiment relates to all other HEMs in terms of expected values. Since we have $\hat{\mu}$ and $\hat{\Sigma}$ for all HEMs, we can describe the corresponding MVGD as follows:

$$X \sim \mathcal{N}_{nh}\left(\hat{\mu}, \hat{\Sigma}\right).$$

However, now we need to generate actual merged HEM vectors using measured data in accordance with the MVGD model. From a mathematical point of view, the challenge is to reorder the grouped sample $\{h^i\}$ for each HEM h^i, such that for each pair of HEMs h^i and h^j the empirical correlation of the grouped samples is close to ρ_{ij}. Equivalently, the challenge is to reorder the grouped samples such that the new corresponding (empirical) covariance matrix is close to $\hat{\Sigma}$. This corresponds to an optimization problem with an existing solution, since the set of potential orders (all permutations) is large but finite. However, it is not a trivial problem from a computational point of view. We construct the solution with a probabilistic approach using the preliminary results of copula theory [122].

Application of copula theory. For each h^i, the empirical distribution function F^i_{emp} lets us transform the grouped sample into a uniform sample. A uniform sample can be transformed into a Gaussian sample by applying the inverse function of the cumulative distribution function of a standard Gaussian distribution, Φ. Therefore, the grouped sample is transformed one-to-one into a standard Gaussian distribution:

$$\Phi^{-1}\left(F^i_{emp}\left(\{h^i\}\right)\right). \tag{7.1}$$

For instance, if $\{h^i\}$ is $\{1, 2, 5, 8\}$, this would map to a uniform sample $\{0.2, 0.4, 0.6, 0.8\}$[1]. Then, those values would map to a standard Gaussian distribution as $\{-0.842, -0.253, 0.253, 0.842\}$[2].

We apply this process to all HEMs, thus obtaining for each measurement of each HEM its counterpart value for the standard Gaussian distribution. Then, we refer to as $\hat{\Sigma}_0$ to the covariance matrix obtained from the transformed data to differentiate it from the one obtained from the measured data ($\hat{\Sigma}$).

Finally, we generate a sample $samp_{MVGD}$ of n_{MVGD} runs (e.g. $n_{MVGD} = 10\,000$) from the MVGD, which we define using $\hat{\Sigma}_0$:

$$X \sim \mathcal{N}_{nh}(0, \hat{\Sigma}_0), \tag{7.2}$$

which produces a joint sample with marginal standard Gaussian distribution (i.e. sampled values follow such distribution). At this point, $samp_{MVGD}$ provides a matrix with as many columns as HEMs (nh), as many rows as HEM vectors we want to generate (e.g. as many as total measurements per HEM), and preserving the correlations across all pairs of HEMs (ρ_{ij}) simultaneously. However, values in the matrix correspond to a standard Gaussian distribution instead of being HEM values read. Thus, we use the actual $samp_{MVGD}$ to produce the indexes for order statistics. In other words, if for a given HEM h^i, $samp_{MVGD}$ has a particular order of values (e.g. k^{th} lowest first, l^{th} lowest second, m^{th} lowest third, and so on and so forth), we set the actual observed values for h^i in the very same order to generate the merged HEM vectors. The easiest way to do this is setting n_{MVGD} to the actual number of measured values per HEM. For instance, if for a given HEM h^i the $samp_{MVGD}$ has produced the values $\{1.121, -0.870, -0.172, 0.343\}$, and the actual values observed are $9, 10, 12, 17$, they would be sorted as follows: $\{17, 9, 10, 12\}$, thus preserving the same ordering, but this time using the actual values read. By following their $samp_{MVGD}$ ordering for all HEMs to organize the actual values measured, we generate as many merged HEM vectors as actual values have been observed for each HEM. For instance, recalling the example in Figure 7.3, where we have 12 values per HEM, we could set $n_{MVGD} = 12$, and would sort the values for each of the 7 HEMs in the same order as their synthetic values in $samp_{MVGD}$.

In fact, once this process is complete, we could assess $\hat{\rho}'_{ij}$ for all pairs of HEMs in the merged vectors and compare them with the original values $\hat{\rho}_{ij}$ obtained from pairwise HEM measurements. Some (small) discrepancy is expected due to statistical reasons (i.e. sampling processes can always produce inaccuracies). Such discrepancy could be reduced with an iterative process where X in Equation 7.2 is obtained as many times as needed and measured HEM values sorted accordingly to obtain new merged vectors where $\hat{\rho}''_{ij}$ is compared to $\hat{\rho}_{ij}$. This process could be repeated a fixed number of times or until a specific criterion is fulfilled. However, this step is purely optional.

Recalling the example for HRM limitations before, MUCH would successfully preserve all correlations across the three HEMs simultaneously, overcoming the limitation of HRM. In particular, MUCH does not favor any particular random variable (HEM) when merging and, instead, all correlations are preserved as accurately as possible at the same time. Instead, HRM strategy is a supervised one where correlations with a particular variable (anchor HEM) are perfectly preserved at the expense of causing large inaccuracies for other correlations if they are weakly correlated with the anchor.

7.2.3 Procedure

For the sake of completion, we provide the application process of MUCH, which includes five main steps. The procedure can be followed visually on Figure 7.2.

- **STEP** ①. The HEM selection for each sub-experiment does not play a role in MUCH. Each HEM will be measured with every other HEM at least once, to capture the relation between them, i.e. to compute the empirical correlation matrix $\hat{S}$. While generating the minimum number of sub-experiments allowing to capture all pairwise HEM correlations is convenient, it is not strictly mandatory for the application of MUCH, so combinations can be generated with greedy

[1]Given a HEM h^i for which we have n values, the uniform sample probability space is split into $n + 1$ identical parts. Out of the $n + 2$ boundary values, we exclude 0 and 1 since they cannot be used later for the Gaussian distribution as their counterpart values would be $-\infty$ and $+\infty$ respectively.

[2]As for the uniform distribution, those values distribute the probability space into $n + 1$ parts with identical accumulated probability.

Table 7.1: *HEMs with observed relevant variability.*

Name	HRM id
PROCESSOR_CYCLES	h^a
CYCLES_LSU_SCHE_STALLED	1
CYCLES_LSU_ISSUE_STALLED	1
BLINK_REQUEST	1
L2_MISSES	1
L2_DEMAND_ACCESSES	1
L2_ACCESSES	2
L2_STORE_ALLOCATES	2
L2_DATA_MISSES	2
L2_RELOADS_FROM_CORENET	2
L2_SNOOP_HITS	2
L2_SNOOP_PUSHES	3
STALL_FOR_RLT_CYCLES	3
STALL_FOR_WDB_CYCLES	3
BIU_MASTER_REQUESTS	3
BIU_GLOBAL_REQUESTS	3

algorithms if needed. For each sub-experiment at least 30 runs ($nr \geq 30$) are needed to allow the use of the Central Limit Theory [87]. In general, the higher nr, the more accurate $\hat{S}$ will be. As a matter of fact, in this work we set $nr = 50$.

- **STEP** ②. Once the values are gathered, map them to a standard MVGD, and compute the covariance matrix $\hat{\Sigma}_0$.
- **STEP** ③. Compute the MVGD using $\hat{\Sigma}_0$ as shown in Equation 7.2, and generate a sample equal in size (n_{MVGD}) to the number of collected values for each HEM in a sub-experiment or in all sub-experiments. Note that the method could also be applied with larger n_{MVGD} values.
- **STEP** ④. (OPTIONAL) As an optimization step, we can compute the correlation matrix of the generated sample $\hat{S}'$ from the MVGD and compare it to the measured correlation matrix $\hat{S}$, for instance, obtaining the Minimum Square Error (MSE). Then, we can repeat step ③ and keep the $\hat{S}'$ with lowest MSE compared to $\hat{S}$. Such process can be repeated as many times as wanted as a way to further increase accuracy without requiring additional runs on the target platform.
- **STEP** ⑤. Copy the order statistics of the sample of the MVGD into the experimental data. Now, the experimental data is finally merged in accordance with the MVGD.

7.3 Evaluation

This section presents the experimental framework, the validation approach followed to evaluate MUCH and compare it with HRM, and the results of the evaluation.

7.3.1 Validation Approach

In this work the experimental validation is performed on the same platform, the NXP T2080, as with HRM in Chapter 6 with the same microbenchmarks and workloads described in Section 3.1.2.

The reference against which to compare MUCH is the actual correlation of each pair of HEMs when measured together in the platform, so that we can validate whether correlations in the merged HEM vector are accurate with respect to real correlations. For completeness, we compare MUCH against HRM in terms of both, accuracy with respect to the real correlations and number of runs required.

For the sake of comparison, we focus on the same 16 HEMs regarded as relevant in HRM [166], which we list in Table 7.1 for completeness. Note that, while all HEMs are treated homogeneously by MUCH – thus meaning that all pairwise HEM correlations are measured and then processed together in an identical basis – the same does not apply to HRM. In particular, HRM needs a HEM to be the anchor. Then, given that the T2080 MPSoC has 6 PMCs and one of them is used by the anchor, the remaining 15 HEMs need to be distributed across 3 sub-experiments. The sub-experiment where each HEM is read for HRM is shown in Table 7.1 in the *HRM id* column.

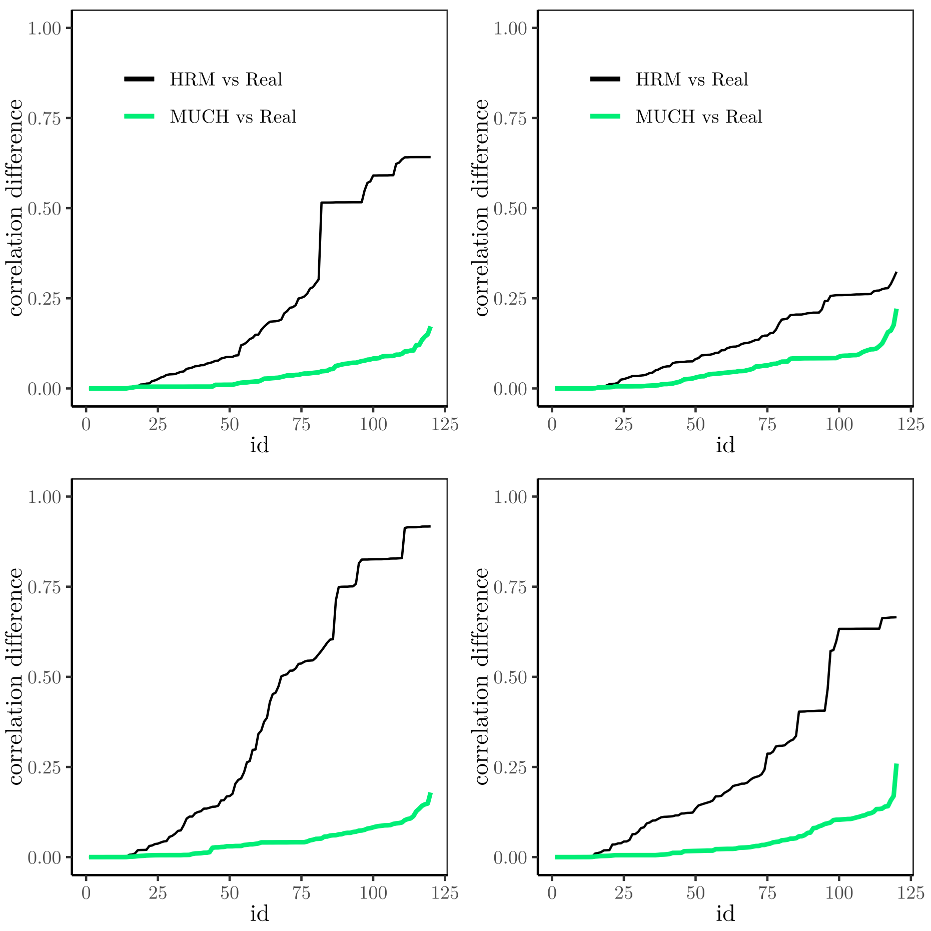


Figure 7.4: *Correlation Difference for each HEMs pair for workload 1 (top-left), workload 2 (top-right), workload 3 (bottom-left), and workload 5 (bottom-right).*

Note that, by using 16 HEMs, there are 120 different pairs of HEMs for which we evaluate the actual correlation obtained for the merged HEM vectors with MUCH and HRM, and compare them against their real empirical correlation, $\hat{\rho}^{i,j}$. In particular, we compute the accuracy for both methods as $\left|\rho^{i,j}_{much} - \hat{\rho}^{i,j}\right|$ and $\left|\rho^{i,j}_{hrm} - \hat{\rho}^{i,j}\right|$.

7.3.2 Results

We evaluate the correlation accuracy obtained, for both MUCH and HRM, against the real empirical correlation. For this first comparison, we set $nr = 50$. The number of sub-experiments is only 3 for HRM ($nb = 3$). For MUCH, while theoretically we could observe 120 pairs of HEMs with $nb = 8$ sub-experiments (15 pairs per sub-experiment with 6 PMCs), we needed $nb = 10$ just following a greedy process to create sub-experiments where we iterate over HEMs from 1 to 16, and for each one we create sub-experiments adding the lowest order HEM not yet observed with any of already selected HEMs in the sub-experiment. Therefore, $n = 150$ for HRM and $n = 1000$ for MUCH.

We show detailed results for the 120 pairs of HEMs in Figure 7.4 for workloads W1, W2, W3, and W5. In particular, W2 corresponds to the case where the improvement of MUCH with respect to HRM is only moderate, W3 to an extreme case with huge improvement, and W1 and W5 to two cases with typical high improvement. The HEM pair values are sorted from lowest difference to highest difference with respect to the real correlation for each technique. As shown, MUCH provides higher accuracy since its differences with respect to the real correlation are much lower than those of HRM. Moreover, the difference in the worst case for MUCH is up to 0.25 for very few pairs of HEMs, whereas HRM reaches values above 0.5 for a non-negligible number of pairs, and even above 0.75 in some cases. Note that the maximum theoretical difference is 2.0, which would occur when the

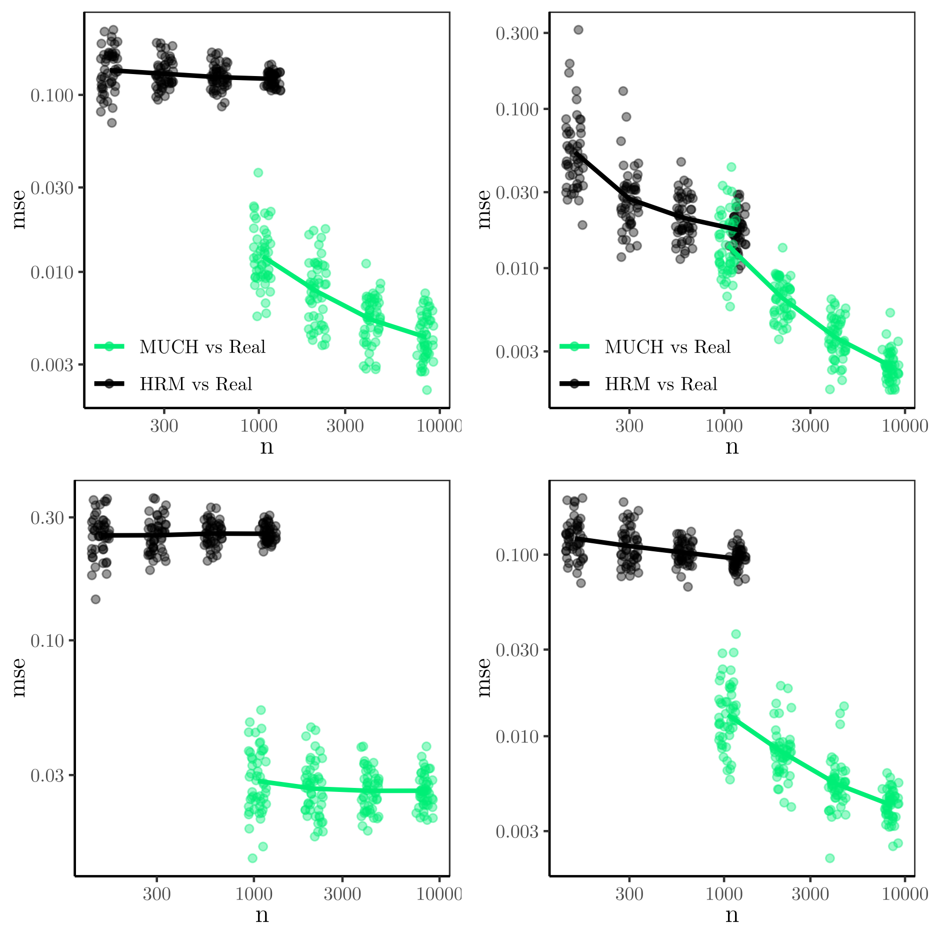


Figure 7.5: *Mean square error for HRM and MUCH as a function of the number of runs. Workload 1 (top-left), workload 2 (top-right), workload 3 (bottom-left), and workload 5 (bottom-right).*

estimated correlation is 1.0 (or -1.0), and the real correlation is -1.0 (or 1.0).

As a second comparison, we study the dependence of each method on the total number of runs $n = nr \cdot nb$, which illustrates the trade-off between cost (in terms of number of runs) and accuracy for both methods. Again, we consider the same 4 workloads, where we vary nr (values 50, 100, 200 and 400), and obtain for each workload, the MSE for their difference with respect to the real correlation across the 120 pairs of HEMs. In particular, to produce a statistically significant comparison, we bootstrapped 50 samples for each of the methods and each nr value. For instance, for $nr = 100$ this implies that we generate a random sample of 100 runs for each sub-experiment and apply the corresponding method on that sample. Then, we repeat the process 50 times, thus obtaining 50 estimates for each method and nr value.

Figure 7.5 shows those results, where dots indicate individual measurements and the line corresponds to the mean across them. Note that both axes are in logarithmic scale. First, we observe that HRM obtains negligible gains from increasing n. Those gains are only noticeable for W2, where pairwise correlations with the anchor HEM are indeed high, allowing HRM to be almost as accurate as MUCH. In any case, HRM apparently plateaus at $n = 1200$ ($nr = 400$).

For MUCH, we observe significant gains in all cases but W3, where increasing n produces limited improvements in accuracy. However, in the other 3 cases we observe improved accuracy as we increase n, and, apparently, such improvement does not plateau even with $nr = 400$, thus offering opportunities to further increase accuracy if the number of runs is increased beyond that number.

When comparing MUCH and HRM, we note that in general, MUCH allows reaching much more accurate merged HEM vectors. It is of prominent importance the case where $n \approx 1000$, because

Table 7.2: *MSE for the 16 workloads under iso-runs (n = 2100).*

Workload	MUCH	HRM
W1	0.003	0.135
W2	0.004	0.024
W3	0.004	0.295
W4	0.006	0.272
W5	0.005	0.110
W6	0.007	0.083
W7	0.006	0.021
W8	0.006	0.068
W9	0.020	0.228
W10	0.007	0.168
W11	0.017	0.207
W12	0.005	0.158
W13	0.008	0.282
W14	0.005	0.106
W15	0.007	0.068
W16	0.017	0.250

it allows performing an iso-cost comparison across both methods, i.e. with the same number of total runs. In this case, where $n_{much} = 1000$ ($nr_{much} = 50$) and $n_{hrm} = 1200$ ($nr_{hrm} = 400$), thus with a slight advantage for HRM, we observe that MUCH is significantly better than HRM in 3 out of 4 workloads, and in the remaining one, where the MSE is already pretty low for both methods, MUCH is slightly better despite its slightly lower number of runs. Finally, note that in both methods, increasing n reduces dispersion of the bootstrap, thus making merged HEM vectors more stable in terms of accuracy with respect to the real correlations.

For completion, we have evaluated all workloads with $n = 2100$, so $nr_{hrm} = 700$ and $nr_{much} = 210$, again with a bootstrap with 50 samples. The mean MSE across the 50 samples is shown in Table 7.2. As expected, MUCH is systematically more accurate than HRM for all workloads, and the difference across both methods is tiny (e.g. W2 and W7) only when HRM is highly accurate, since MUCH is always highly accurate. In fact, the worst MSE for MUCH (0.020 for W9) is indeed better than the best MSE for HRM (0.021 for W7).

So far we have shown that MUCH outperforms HRM under iso-cost (identical number of runs n), and naturally, under identical number of runs per sub-experiment (iso-nr), where MUCH has a higher n value. Moreover, we have shown that in all cases MUCH is highly accurate. However, the number of sub-experiments needed by MUCH is much more dependent on nh and np than that of HRM. For instance, if nh is high or np is very low, MUCH may need many sub-experiments whereas HRM only needs $n_{hrm} = nr \cdot \lceil \frac{nh-1}{np-1} \rceil$. In particular, for MUCH we need to observe $\binom{nh}{2} = \frac{nh \cdot (nh-1)}{2}$ pairs of HEMs, and each sub-experiment allows observing up to $\binom{np}{2} = \frac{np \cdot (np-1)}{2}$ pairs. Assuming that sub-experiments for MUCH can be optimized to generate always unobserved pairs of HEMs only, the number of sub-experiments would be ratio between both values, and hence, the total number of runs would be $n_{much} = nr \cdot \lceil \frac{nh \cdot (nh-1)}{np \cdot (np-1)} \rceil$. Figure 7.6 shows n_{much} and n_{hrm} for the case $np = 6$, as in the T2080, and $nr = 100$, when varying nh. As shown, n_{much} grows much faster than n_{hrm} as we increase nh. Therefore, there may be cases where n_{much} might not be affordable and the only affordable solution is HRM. In those cases, despite the limitations of HRM, such solution has been shown to be systematically better than any other alternative (except MUCH) [166], and thus, it would be the best choice.

7.4 Summary

In the previous chapter, we proposed the HRM approach to merge HEMs from different runs accurately preserving their correlation with respect to one anchor HEM (i.e. processor cycles) building on order statistics. However, HRM does not always preserve the correlation between other pairs of

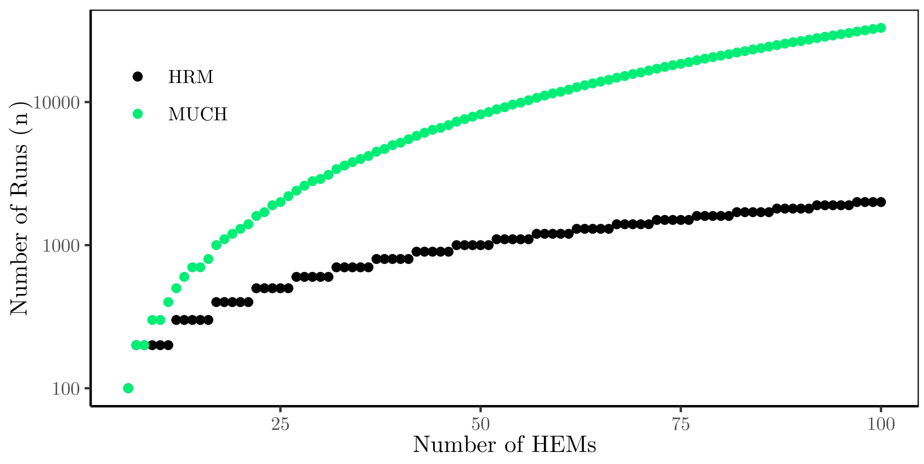


Figure 7.6: *Number of total runs as a function of the number of HEMs to arrange. The parameters on n_{much} and n_{hrm} are $np = 6$, $nr = 100$.*

HEMs that might be lost to a large extent. This chapter copes with HRM limitations by proposing the MUlti-Correlation HEM reading and merging approach (MUCH). MUCH builds on multivariate Gaussian distributions to merge HEMs from different runs while preserving pairwise correlations across each individual pair of HEMs simultaneously. Our results on an NXP T2080 MPSoC used for avionics systems show that MUCH largely outperforms HRM for an identical number of input runs. However, MUCH requires significantly more data to achieve such performance. Therefore, both HRM and MUCH have their use depending on the needs and resources of the engineer.

Part IV

Contention Modelling for CRTES

Chapter 8

Clean Execution Times for MBPTA

8.1 Introduction

Timing validation for CRTES (e.g. automotive systems) occurs in late integration stages when it is hard to control how the instances of software tasks overlap in time. To make things worse, in complex software systems, like those for autonomous driving, tasks schedule has a strong event-driven nature, which further complicates relating those task-overlapping scenarios (TOS), which produce contention, captured during the software timing budgeting and those observed during validation phases.

Multiple approaches exist to account for contention due to concurrency on the access to shared hardware resources, as part of the design and verification process of CRTES. While it is not the purpose of this thesis surveying on this topic, we identify the main families of techniques. Some techniques control the impact of contention by building upon some hardware and/or software support [93, 126, 183]. Other techniques use such support to completely avoid any impact due to contention [8, 17, 25, 129, 130]. Finally, some other approaches, rather than controlling or mitigating contention, aim at upper-bounding it [47, 145].

While each of those techniques has its assumptions and requirements and has been proven appropriate for WCET estimation as part of the verification process, a validation step is still needed during system integration phases, in accordance with safety-related systems development processes. In this context, performing a reliable timing validation process building on measurements with arbitrary overlap across tasks is a difficult challenge.

Tasks overlapping heavily impacts individual tasks' execution time and is an aspect of multicore-based systems for which no satisfactory solution exists yet. Given a reference analysis task, the number of other tasks (and the particular tasks) that overlap with it changes its execution time. Likewise the degree of overlap, i.e. how long tasks overlap, affects also its timing behavior. Task overlap varies in non-obvious manners across individual measurements and also during operation. Therefore, measurements are subject to hard-to-control contention, which brings uncertainty to the validation process. While the user might have means to observe the variability among tasks in observed TOS during tests, it is hard to enforce specific TOS.

In this chapter we propose *CleanET*, that stands for *clean execution times*, an approach to distill both, contention-free measurements as well as measurements subject to specific contention levels, to enable a reliable timing validation process that captures any TOS that might occur during operation. CleanET builds upon *statistical dependence analysis* to derive the dilation factor r affecting execution times due to simultaneous task execution. CleanET resamples the execution times measured during testing to derive tasks execution time under (1) no overlapping scenarios (i.e. in isolation), (2) full single overlapping (i.e. when the task overlaps 100% of the time with exactly one other task), and (3) data with any specific overlap representing the worst overlap of interest (e.g. full overlap with n other tasks). CleanET estimates provide additional means of validation for derived time budgets and alleviates the pressure on end users to produce tests cases with different TOS.

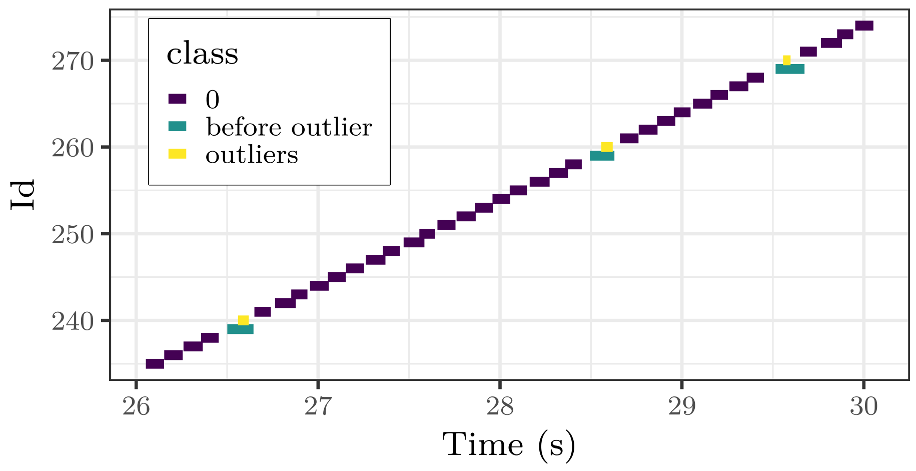


Figure 8.1: *Timeline interval of Prediction showing the behavior of the outliers.*

We apply CleanET on the Autonomous Driving (AD) framework Apollo [11], described in Section 3.1.3, where the different modules (tasks) of the framework overlap partially and in an 'arbitrary' manner, thus exposing the challenge that CleanET targets. Apollo is an intricate set of tightly-coupled software modules, including an Operating System, which jeopardizes the possibility of collecting measurements for each module under controlled TOS. CleanET successfully distills the dilation factor and resamples measurements to any TOS of interest.

8.2 Timing Analysis Validation on Apollo

In this work the methodology will be carried out within the Apollo Autonomous Driving Framework described in Section 3.1.3. As explained, with this framework we are able to obtain a timeline of execution time data on all available modules to monitor from a navigation test.

From all Apollo modules, we focus on those with most intensive computation requirements, which are the ones using the GPU: Prediction and Perception. Those modules create mutual interference (contention), and hence, challenge timing validation. Both modules use the GPU in a similar manner since both build upon Deep Neural Networks and Recurrent Neural Networks. Both call the same matrix multiplication primitives with similar parameters. Therefore, whenever they attempt to use the GPU simultaneously, they are expected to create significant interference on each other. In our analysis, for simplicity, we model how the execution time of the Prediction module is affected by the overlapping with the execution of Perception jobs. The symmetrical analysis can also be performed, but we omit it due to lack of space and since it does not offer further insights.

8.2.1 Prediction Module Timing Behavior

From our analysis of the prediction module's execution times we have realized that the vast majority of executions occur in a non-overlapping manner between each other with few exceptions. In particular, periodically a Prediction job starts with an unusual delay with respect to the finalization time of the previous job; then said job takes longer execution time than expected. Sometimes, before a long job finishes, a new Prediction job is released, but it is cancelled soon after its start since the previous one is still running. Therefore, sporadically (exactly once every 10 jobs at most), we have two jobs with anomalous timing behavior. This is illustrated in Figure 8.1, where we show an excerpt of the chronogram for the Prediction module. As shown, anomalies appear at time instants 26.5, 28.5 and 29.5 seconds, shown in green color (abnormally high execution times) and yellow color (abnormally low execution times after abnormally high ones). We, therefore, classify execution times into three classes: *normal* corresponds to the normal Prediction behavior, *before outlier* class corresponds to overly large execution times, and the *outliers* class corresponds to the outliers that start before the previous job has finished.

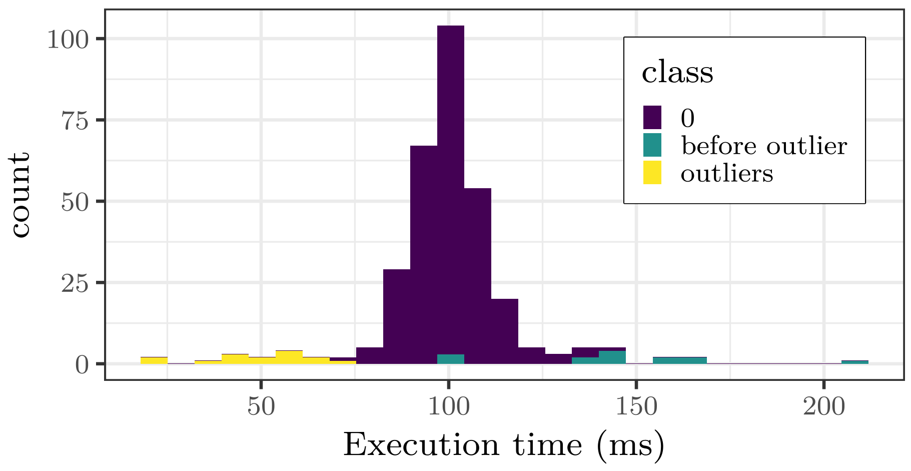


Figure 8.2: *Execution time histogram of Prediction jobs.*

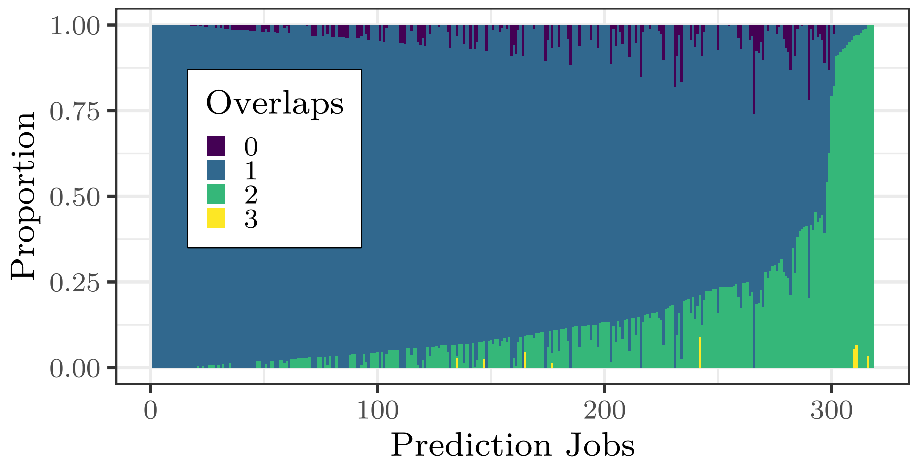


Figure 8.3: *Portion of normalized execution times that overlaps with different number of Perceptions simultaneously.*

Figure 8.2 shows the histograms of each class with the same color code as for previous figure. The particular execution time mean (in milliseconds) for each class is 100, 144, and 50.3 respectively. Hence, outliers have shorter execution times, whereas the execution times before the outliers are particularly high. Finally, *normal* concentrates most jobs in a relatively narrow execution time interval.

Based on this evidence, we conclude that execution times for each class need to be treated separately. Moreover, we build on the common practice in CRTES for automotive and robotics, among others, where some jobs can fail (e.g. due to an overrun) as long as the number of failures in the last M jobs does not exceed a particular threshold, N [187], thus building on top of the typical worst-case response time rather than on the absolute worst-case response time. In our case, by simply studying *normal* jobs, we discard up to 2 jobs every 10, whose timing behavior, if it violates any deadline, would not be a problem as long as 2 out of 10 runs are allowed to fail, thus much in line with automotive and robotics common practice.

8.2.2 Aggregation of Overlaps

Once filtered the Prediction jobs of interest, we study their overlap with Perception jobs. Our analysis reveals that each Prediction job can overlap, during different parts of its execution, with up to 3 Perception jobs. The fraction of time overlapped with 0, 1, 2, or 3 Perception jobs for each of the Prediction jobs is depicted in Figure 8.3. Jobs have been sorted from highest to lowest fraction of overlap with 1 Perception job to ease graphical representation. As shown, most Prediction jobs spend most of their execution time overlapped with exactly 1 Perception job. The fraction of time overlapped with 2 Perception jobs is also significant. Little time is spent without overlapping in general. Finally, only sporadically some Prediction jobs overlap with 3 Perception jobs.

8.3 CleanET

8.3.1 Setting the Objective

Complex MPSoCs like the NVIDIA Xavier have a large internal hardware and software (e.g. CUDA drivers) state. Modelling this state, which strongly influences tasks' execution time, is quite difficult. In such a complex system, the execution time of a task can be regarded as a random variable, J. When several tasks execute simultaneously, they affect each other timing behavior due to contention in the access to shared resources, even if tasks have no data or control dependence among them. The impact of the contention interference depends on (i) the number of tasks with overlapping execution; (ii) the percentage of time they overlap; (iii) the particular part of their code that overlaps; and (iv) the inputs to the program that might affect tasks access pattern to shared resources. In this work, we focus on the first two factors, and hence do not model the particular section of the code of a task that overlaps, nor tasks inputs.

Our objective is to enable the estimation of the execution time of tasks under different TOS. In particular, by modelling the dependence of the execution time on the TOS (i.e. how much overlapping and with how many tasks), we enable the reliable estimation of the dilation factors, $r_1, \cdots, r_m$, impacting execution time due to the task overlapping with $1, \cdots, m$ tasks respectively. Hence, r_i is the factor by which the execution time with no overlap of the task under analysis must be multiplied when overlapping with i tasks. For instance, if $r_i = 1.5$, the non-overlapping execution time of the task under analysis is 10ms, and it overlaps with i tasks during 4ms, then the execution time is $6 + 4 \cdot 1.5 = 12$ms.

Once dilation factors r_i are obtained, and given that we know the start and end time for all jobs – and so their particular overlappings, we can deduce execution time measurements under any TOS (i.e. those regarded relevant but that could not be tested) since we can obtain the pristine (non-overlapped) execution time measurements removing the impact of contention in the measurements collected, and apply dilation factors to model any TOS of interest for validation purposes.

The derived dilation factors can be used to validate the timing behavior of the task under analysis by comparing them against the corresponding bounds from the verification phase, e.g. under worst overlapping conditions. They can also be used to contrast the output of timing analysis techniques that predict WCET estimates under different TOS. For instance, since execution times under a given (homogeneous) overlap correspond to random variables independent and with identical distribution, then we could use MBPTA techniques [3, 44, 76, 141] for that purpose.

It is worth noting that, unlike previous works studying dependent execution times [19, 114, 141], which focus on stationary processes and hence, on dependencies across measurements, our work considers external dependencies that are unlikely to be periodic and that cannot be studied only based on the actual execution time measurements of the jobs of the task under analysis, but accounting for the execution times (more precisely, for the start and end times) of the jobs of the other tasks running simultaneously in the system. Stationary processes could be normally studied on top of the measurements obtained dilating them with our approach to match worst-case overlaps during operation. Thus, our work is orthogonal and complementary to previous works since they consider different types of dependencies (i.e. we target dependencies on the TOS).

8.3.2 Modelling Approach

CleanET assumes a baseline (basal) state without any impact of dilation that corresponds to the execution in isolation with no overlap with any other task. In practice the basal state may never occur in the system as it may be not possible to run the task under analysis without overlapping its execution with others. We denote the hardware/software state information in this basal state as $\mathcal{I}_0$.

CleanET builds around the mathematical expectation E. Let $\mathcal{I}$ be the available information on the state of the system for any arbitrary state, and $\mathrm{E}(T|\mathcal{I})$ the expected execution time T assuming that the system is in a given (observed) state $\mathcal{I}$. The dilation coefficient, r is defined as follows, thus relating the execution time of the basal state with that of any other state:

$$\mathrm{E}(T|\mathcal{I}) = r\mathrm{E}(T|\mathcal{I}_0). \tag{8.1}$$

Whenever the system is not in the basal state, then the expectation of the execution time, $\mathrm{E}(T|\mathcal{I})$, is dilated by the coefficient r, where in general $r > 1$ (i.e. execution overlapping normally leads to increased execution times). In fact, Equation 8.1 holds for any state where the amount of overlapping in $\mathcal{I}$ is higher than in $\mathcal{I}_0$, thus meaning that any TOS necessarily results in an increase of tasks execution time.

The basal state of the system (i.e. the execution in isolation) cannot be observed in general. Otherwise, by observing such state and any other state, we could trivially derive r. Instead, we build upon the measurements collected with arbitrary TOS from a real system where no practical control can be exercised to enforce specific TOS, as it is the case, for instance, for most automotive systems. Given a finite sample of execution times and variables characterizing the state of the system (i.e. how jobs overlap), we want to estimate a basal state of the system as a random variable, as well as the dilation coefficient r. For the sake of this discussion, we consider that jobs either overlap or do not overlap, neglecting whether they overlap with one or more jobs. Later we extend the discussion to arbitrary numbers of overlapping jobs.

8.3.3 Mathematical Development of CleanET

The simplest way for describing the state of a system along the execution period T is by summarizing the information of its state. This can be done with only one property through a binary measure for each instant of time, t of a fixed duration (e.g. 1 processor clock cycle in the extreme), see Equation 8.2 where t is the particular time instant when we assess the property. In this work the particular property is whether the execution of the particular job of the task under analysis overlaps with the execution of any other job.

$$W_t = \begin{cases} 1 & \text{if the property holds,} \\ 0 & \text{otherwise.} \end{cases} \tag{8.2}$$

We can summarize this process through the total time during which the property holds, i.e. $W_t = 1$, denoted by V. The remaining time, $T - V$, is denoted by U. This approach approximates the basal state of the system by $\mathcal{I}_0 = (V = 0)$ and Equation 8.1, so that:

$$\mathrm{E}(T|V = v) = r(v)\mathrm{E}(T|V = 0). \tag{8.3}$$

In Equation 8.3, $\mathrm{E}(T|V = 0)$ stands for the expectation ($E(\cdot)$) of the execution time (T) when there is no overlap at all ($V = 0$), $r(v)$ is the dilation factor given a specific overlap v, and therefore, $\mathrm{E}(T|V = v)$ is the expectation of the execution time given the specific overlap v.

Likewise, let Q be the proportion of the execution time, T, where the property holds (V/T) and $1 - Q$ where it does not (U/T). This approach approximates the basal state of the system by $E(T|Q = 0)$ and Equation 8.1, so that:

$$\mathrm{E}(T|Q = q) = r(q)\mathrm{E}(T|Q = 0). \tag{8.4}$$

Both Equation 8.3 and 8.4, allow using regression methods for parameter estimation [181]. In particular, we want to obtain the dilation coefficient r, which we obtain as $r(1)$, so when $q = 1$ (full overlap).

8.3.4 Obtaining *r*

Let $X = (T|Q = 0) = (T|V = 0)$ be the execution time of the task when the state of the system is basal. Note that $X = (T|V = 0)$ is the actual random variable, which may realize into any specific execution time, instead of $E(T|V = 0)$, the expectation, that can only be the Expectation, thus a single value (i.e. the mean of an infinitely large sample). Let $Y = (T|V) = (T|Q)$ be the observed execution time, thus with V time of overlapping (or Q fraction of overlapping). If an observation of Y is given (the actual execution time measured), then V is fixed to v, and Q fixed to q, for some v and q. Note that v and q are directly obtained from observation of the system since we have the start and end time for each job of each task, and hence, we can determine whether the specific job of the task under analysis overlaps its execution with any other job at any time instant.

We can decompose the observed execution time, Y, as follows:

$$Y = U + V = (1 - q)Y + qY, \tag{8.5}$$

$(1 - q)Y$ is the part of the execution time of the job in basal state (no overlapping). Hence, we can define $(1-p)$ such that $(1-q)Y = (1-p)X$, thus relating the execution time with overlapping (Y) with the execution time in isolation (X). From this equivalence, we can describe the dilation coefficient r as follows:

$$Y = U + V = (1 - q)Y + qY = (1 - p)X + rpX, \tag{8.6}$$

where the rightmost part of the equation decomposes the measured execution time into $(1 - p)X$, so the fraction of non-overlapping execution $(1 - p)$ multiplied by the basal state execution time (X), and rpX, so the fraction of overlapping execution (p) multiplied by the dilation factor (r) and the basal state execution time.

Equivalently, we can apply the distributive property:

$$Y = ((1 - p) + rp)X. \tag{8.7}$$

Moreover, since $U + \frac{V}{r} = X$, then $U = X - \frac{V}{r}$, and we can obtain a linear expression:

$$\begin{aligned} Y &= U + V = X - \frac{V}{r} + V, \\ Y &= \frac{r-1}{r}V + X. \end{aligned} \tag{8.8}$$

Then, we can apply linear regression to obtain r and X in Equation 8.8 since Y and V are known.

Note that the basal state corresponds to $V = 0$, so $Y = X$. Also note that if all execution time is overlapped with other jobs (so $V = Y$), we would have the following:

$$Y = \frac{r-1}{r}Y + X. \tag{8.9}$$

We could transform it into the following expression:

$$Y\left(1 - \frac{r-1}{r}\right) = X. \tag{8.10}$$

Thus, having that $\frac{Y}{r} = X$, and $Y = rX$, which matches Equation 8.7 when $p = 1$ (overlapping fraction).

Based on the assumptions of the model, X and Q are independent, which is equivalent to X and P being independent.

Proof. Here we use the theorem which states that functions of independent random variables are independent. Let A and B be subsets of the real numbers $\mathbb{R}$. Let $g^{-1}[A]$ and $h^{-1}[B]$ denote the pre-image of A and B under g and h respectively. Applying the definition of independent random variables:

$$\begin{aligned}\Pr(g(X) \in A, h(Y) \in B)) &= \Pr(X \in g^{-1}[A], Y \in h^{-1}[B]), && (8.11)\\ &= \Pr(X \in g^{-1}[A])\Pr(Y \in h^{-1}[B]), && (8.12)\\ &= \Pr(g(X) \in A)\Pr(h(Y) \in B). && (8.13)\end{aligned}$$

□

Then, it follows that, in general, given X and P being non-observed independent random variables, and U and V observed random variables such that

$$U = (1 - P)X \quad \text{and} \quad V = rPX,$$

for some coefficient $r > 0$, then a point estimation for r, P and X is given by

$$\begin{aligned}\hat{X} &= U + V/\hat{r}, && (8.14)\\ \hat{P} &= V/(\hat{r}U + V), && (8.15)\\ \hat{r} &= \min_r \left\{\mathrm{cov}(\hat{X}, \hat{P})\right\}. && (8.16)\end{aligned}$$

However, point estimations are not satisfactory from a safety perspective due to the uncertainty caused by the potential error in the estimation of execution times. Therefore, we propose a pessimistic method that overestimates the coefficient r, which we detail next.

8.3.5 Intrinsic Pessimism

Equation 8.8, states the linear relationship between Y and V. On the other hand, the linear regression of Y and V provides us with (α, β, Z) such that $\alpha, \beta \in \mathbb{R}$, and Z is a random variable that maximizes the independence between Y and V such that $Y = \alpha + \beta V + Z$. Given the definition of β in simple linear regression [181]:

Proof. Y, V and Z need: $\mathrm{cov}(V, Z) = 0 \implies Y = \alpha + \beta V + Z$. Using the expression $Y = \alpha + \beta V + Z$:

$$\mathrm{cov}(V, Y - \alpha - \beta V) = 0. \tag{8.17}$$

Which yields:

$$\begin{aligned}\mathrm{cov}(V, Y) &= \mathrm{cov}(V, \alpha) + \mathrm{cov}(V, \beta V), && (8.18)\\ \mathrm{cov}(V, Y) &= \beta \mathrm{var}(V), && (8.19)\\ \beta &= \frac{\mathrm{cov}(Y, V)}{\mathrm{var}(V)}. && (8.20)\end{aligned}$$

Which is the estimator of β in the linear regression. □

Now, using Equation 8.8, we can yield the following expression:

$$\beta = \frac{\mathrm{cov}(Y, V)}{\mathrm{var}(V)} = \frac{\mathrm{cov}(\frac{r-1}{r}V + X, V)}{\mathrm{var}(V)} = \frac{r-1}{r} + \frac{\mathrm{cov}(X, V)}{\mathrm{var}(V)}. \tag{8.21}$$

Therefore, the equivalence $\beta = \frac{r-1}{r}$ would hold if X and V were independent:

$$\mathrm{cov}(X, V) = \mathrm{cov}(X, XPr) \tag{8.22}$$

Let us check the dependency between V and P:

Proof. Start by laying out the expression for $\mathrm{cov}(X, XP)$:

$$2\mathrm{cov}(X, PX) = \mathrm{var}(X) + \mathrm{var}(PX) - \mathrm{var}(X - PX). \tag{8.23}$$

Determine $\mathrm{var}(PX)$ and $\mathrm{var}(X - PX)$.

$$\mathrm{var}(PX) = \mathrm{var}(P)\mathrm{var}(X) + \mathrm{var}(P)E(X)^2 + \mathrm{var}(X)E(P)^2, \tag{8.24}$$

$$\mathrm{var}(X - PX) = \mathrm{var}(1 - P)\mathrm{var}(X) + \mathrm{var}(1 - P)E(X)^2 + \mathrm{var}(X)E(1 - P)^2. \tag{8.25}$$

By subtracting those two equations we obtain the right-hand side of Equation 8.23:

$$\mathrm{var}(X)\left[E(P)^2 - E(1 - P)^2 + 1\right] = 2\mathrm{cov}(X, PX), \tag{8.26}$$

$$\mathrm{var}(X)\left[E(P)^2 - (1 - E(P))^2\right] = 2\mathrm{cov}(X, PX), \tag{8.27}$$

$$\mathrm{var}(X)E(P) = \mathrm{cov}(X, PX) > 0. \tag{8.28}$$

Which holds in our case because $P \in [0, 1]$ and $X \in \mathbb{R}^+$. □

Given that X and P are not independent, the relationship between r and β yields:

$$\beta = \frac{r-1}{r} + \frac{\mathrm{cov}(X, V)}{\mathrm{var}(V)} > \frac{r-1}{r}, \tag{8.29}$$

since $\mathrm{cov}(X, V) > 0$. Note that the covariance between X and V is generally unknown. Thus, by ignoring it, we overestimate the factor $\frac{r-1}{r}$, which, given that $r > 1$ as indicated before, implies that we overestimate r.

The consequence of overestimating r is that, by considering scenarios where the overlapping is higher than measured, the overestimated r leads to higher predicted execution times than those in the real system. In our case, this implies that, if those predicted measurements respect timing budgets, then real system measurements would necessarily also respect the budgets. Also, if the system can overrun the timing budget, predicted measurements obtained with CleanET will indicate even higher overruns, so faults will be reliably detected. However, in some cases, time budgets may be respected and CleanET report that they are violated. Hence, while this may cost some tightness by imposing the allocation of larger time budgets than needed, our approach does not challenge the safety of the system tested.

8.3.6 CleanET for Multiple Overlappings

In the previous section we focused the case where one specific job either does not overlap or partially overlap with another job. The model can be easily extended to account for several jobs overlapping at the same time (from 1 to m). This is done by defining Equation 8.2 for each number of overlaps, so having $W_t^1, \cdots, W_t^m$, where the property in each case is whether there are *exactly* i overlaps. For instance, for $i = 2$, $W_t^2 = 1$ if and only if the job overlaps with exactly 2 other jobs in time t. For any other number of overlapping jobs (e.g. 0, 1, 3, etc), $W_t^2 = 0$. Hence, V_i stands for the total time where property W_t^i holds.

We can, therefore, extend our original definition of Y ($Y = U + V = (1 - q)Y + qY$) decomposing V and q across the different types of overlapping (from 1 to m overlapping jobs) as follows:

$$Y = U + \sum_{i=1}^{m} V_i = \left(1 - \sum_{i=1}^{m} q_i\right)Y + \sum_{i=1}^{m} q_i Y. \tag{8.30}$$

This allows us to reformulate Equation 8.7 as follows:

$$Y = \left(\left(1 - \sum_{i=1}^{m} p_i\right) + \sum_{i=1}^{m} r_i p_i\right)X, \tag{8.31}$$

which can be transformed into the following equation where linear regression can be directly applied:

$$Y = \sum_{i=1}^{m} \left(\frac{r_i - 1}{r_i} V_i \right) + X, \tag{8.32}$$

where Y and all V_i are known, and X and all r_i dilation factors are obtained through linear regression.

8.4 Evaluation

For the evaluation of CleanET, we first estimate r_i dilation factors for the different task overlaps. Then, we validate how dilation factor estimation matches expectations. Finally, we show how those dilation factors can be used to estimate execution time bounds for different overlapping scenarios, thus allowing to validate whether time budgets are violated.

8.4.1 Dilation Factors (r_i)

Single Dilation Factor

We have applied CleanET to obtain the dilation factors, r_i, for the different numbers of overlapped tasks. Results are shown in Table 8.1 considering a single dilation factor (whether overlap exists or not), as per Section 8.3.4. As shown, execution time with no overlap at all is around 16.32ms, with a standard deviation of 1.64ms. However, the impact of overlap is huge since $r = 6.85$ with a standard deviation of 0.78. Hence, on average, we could expect the execution time will full overlap to be around 112ms (16.32ms · 6.85). For the sake of completeness, the table also provides the number of execution time observations used (289), which are those for *class 0* before, and the degree of correlation measured between non-overlapped and overlapped execution time measurements, which is, as expected, very high (0.9).

Table 8.1: *CleanET applied to filtered data with Overlap as co-variate.*

Overlap (r)	6.85 ± 0.78
Constant ($p = 0$)	16.32 ± 1.64
Observations	289
Adjusted R-squared (correlation):	0.90

All Dilation Factors

As a second step, we have applied CleanET to obtain individual dilation factors for 1, 2 and 3 jobs overlapping with the jobs of the task under analysis, as per Section 8.3.6. Results are shown below in Table 8.2. We observe that results for 1 or 2 jobs overlapping are very similar and ranges (e.g. $\mu \pm \sigma$) overlap almost completely. Statistically, we cannot prove that they are distinguishable and, therefore, we conclude that overlapping with 1 or 2 jobs must be considered together, thus defining that the property in Equation 8.2 holds if the task overlaps with exactly 1 or 2 jobs. In the case of 3 jobs overlapping, we observe that (1) the value of r_3 is far lower than that for r_1 and r_2, which would mean that overlapping with 3 jobs leads to a lower dilation factor (so a lower execution time increase) than overlapping with 1 or 2 jobs, which is against intuition. However, the real problem with 3 overlaps relates to the fact that, out of the 289 *class 0* measurements, only 9 have 3 jobs overlapping with the task under analysis for some time, which is, in practice, too scarce data to raise any reliable prediction. In fact, the (very high) standard deviation for this case already indicates this behavior indirectly. The number of measurements including data for each overlap case is provided in Table 8.3 for completeness. Finally, note that, while the case with no overlap ("Constant" in the tables) has no meaningful change with respect to the case where we fit only r, it is not absolutely identical. This relates to the fact that all parameters in Equation 8.32 are fit together, thus dealing to minor variations when varying the parameters to fit.

Table 8.2: *Linear model applied on filtered data with Overlap time of one, two and three processes at a time.*

Overlap 1 (r_1)	6.97 ± 0.81
Overlap 2 (r_2)	6.55 ± 0.81
Overlap 3 (r_3)	3.23 ± 2.42
Constant ($p = 0$)	16.32 ± 1.65
Observations	289
Adjusted R-squared (correlation):	0.90

Table 8.3: *Number of* class 0 *measurements with part of its execution with 0, 1, 2, 3 jobs overlapping.*

Number of overlapping jobs	Measurements	Percentage (w.r.t. 289)
Overlap 0	137	47.4%
Overlap 1	288	99.7%
Overlap 2	218	75.4%
Overlap 3	9	3.1%

Statistically Significant Dilation Factors

After concluding that 1 and 2 overlapping measurements are not statistically distinguishable, we apply CleanET again considering only two dilation factors: r_{1-2} for 1 or 2 overlaps, and r_3 for 3 overlaps, as shown in Table 8.4 below. We note that r_{1-2} is not distinguishable in practice with r in Table 8.1 when considering just one dilation factor since the amount of data for 3 overlaps is too little to cause meaningful differences. We also note that, by considering 1 and 2 overlaps together instead of separated, the case for 3 overlaps varies drastically, which reflects the weakness of the fit for 3 overlaps due to the too little data available for this case.

Table 8.4: *Linear model applied on filtered data with Overlap time of one and two processes added, and three processes.*

Overlap 1-2 (r_{1-2})	6.88 ± 0.78
Overlap 3 (r_3)	2.75 ± 1.68
Constant ($p = 0$)	16.31 ± 1.64
Observations	289
Adjusted R-squared (correlation):	0.90

Overall, our results show that scenarios with 0, 1 or 2 overlaps can be reliably resampled from the data available. If the case with 3 overlaps is regarded as relevant for operation conditions, then additional input data would be required including many more measurements corresponding to that scenario, so that CleanET could deliver a reliable dilation factor for this case.

8.4.2 Validation of the Method

In general, these solutions cannot be validated due to the lack of reference data, which in this case would correspond to measurements with the complete execution time with a constant number of jobs overlapping (e.g. 100% of the execution time overlapping with exactly 1 job). However, in our case, we have measurements with exactly 1 job overlapping during their complete execution. In fact, since 1 and 2 overlappings have been shown to be indistinguishable, and 3 overlappings occur seldom (too occasionally to be statistically significant), we consider as reference measurements those in which some overlapping (with either 1, 2 or 3 jobs) occurs during the whole execution. Hence, we performed the following:

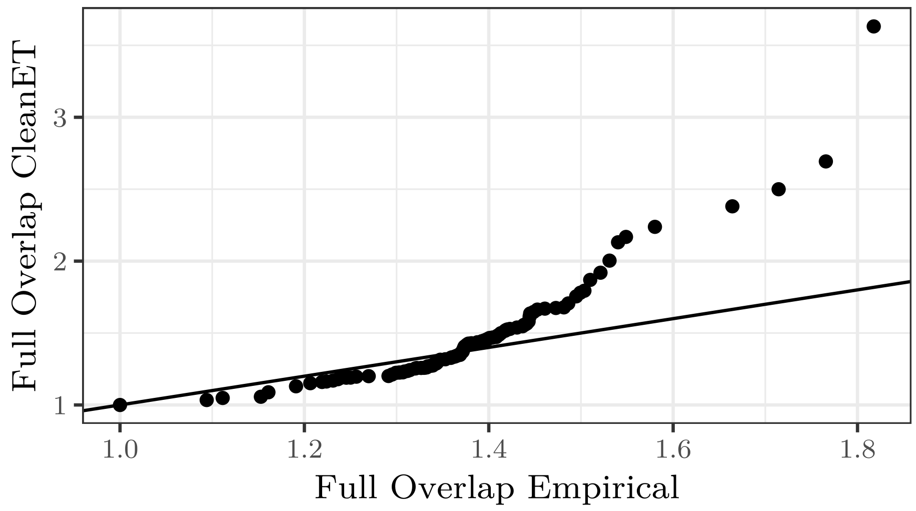


Figure 8.4: *QQ-plot of full overlap resampled data with CleanET with respect to empirical data with full overlap.*

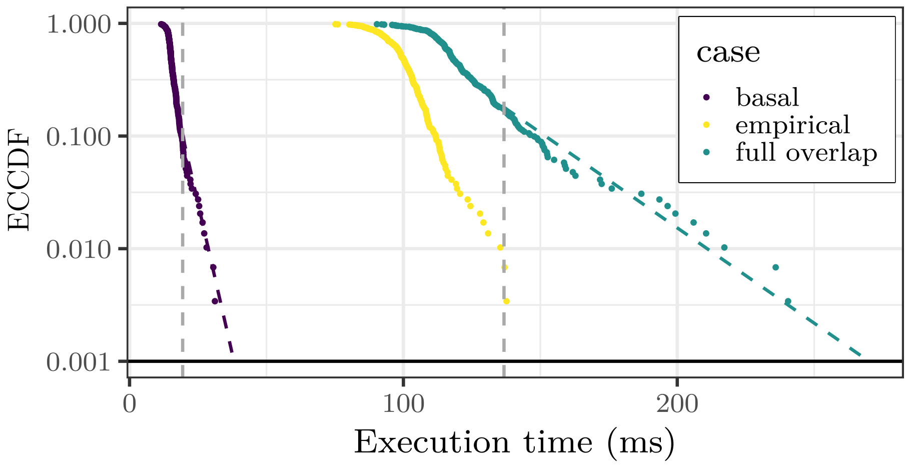


Figure 8.5: *Execution time bounds for no overlapping ($p = 0$) and full overlapping ($p = 1$).*

1. Split the set of measurements into 2 categories: one with those measurements with 100% overlap (OVL1 group, 152 values), and those with non-full overlap (OVLmix, the remaining 137 values).
2. Apply CleanET on OVLmix data.
3. Compare the results obtained with CleanET resampling data in OVLmix to the case with full overlap, with the actual data with those overlapping characteristics (OVL1 data) with a QQ-plot (see Figure 8.4).

In the figure, we would like to have a linear relation between the empirical data and the data resampled with CleanET (straight line). However, the fact that r is overestimated leads to non-linearity. Results show that CleanET, using OVLmix data, produces higher values than those observed empirically (OVL1), thus corroborating our expectation of having overestimated execution times for high overlaps due to the overestimation of r. This supports empirically our expectations with a reference data set different to that used by CleanET by partitioning the data into OVL1 and OVLmix groups.

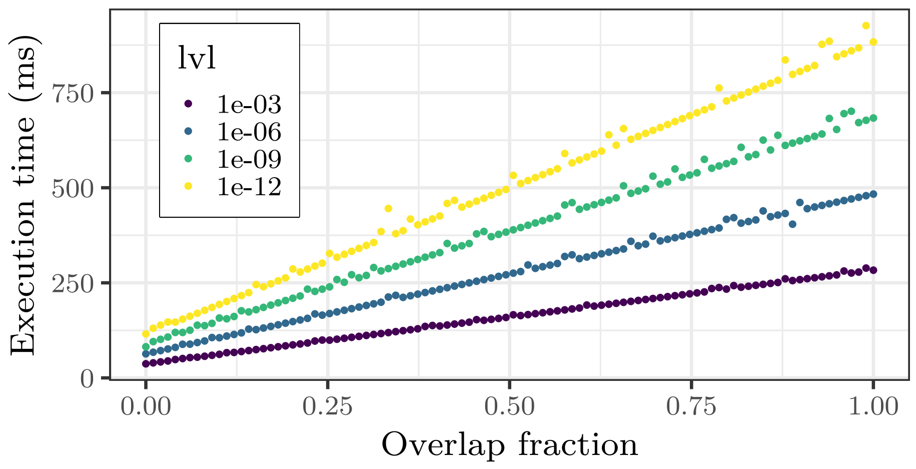


Figure 8.6: *Impact of the dilation factor (r) on the execution time bounds as we vary the degree of overlapping (p).*

8.4.3 Using CleanET Results

We have used CleanET to generate, from the actual measurements obtained from the system with arbitrary overlaps, measurements corresponding to overlapping scenarios of interest. In particular, since the purpose of timing validation is assessing that no overrun occurs, we have considered the case with full overlap. For the sake of illustration, we also show results for the case of no-overlap.

Figure 8.5 plots the empirical complementary cumulative density function (ECCDF) for the three scenarios: actual data measured (empirical), data obtained with CleanET with no overlap (basal), and data obtained with CleanET with full overlap. Together with the empirical data, we fit an exponential tail to the highest values of the extreme cases (basal and full overlap), which has been recommended as a suitable approach to predict high execution times [3, 44]. For that purpose, we build on peaks-over-threshold methods selecting the threshold as indicated in [90].

Measurements resampled with CleanET allow considering cases that could not be explicitly tested in the system under test. As shown, as expected, the full overlap case leads to execution times higher than those of the empirical (measured) case, whereas the basal case, instead, leads to the lowest execution times one would expect in the real system. Note that, as discussed in Section 8.3.5, r values are overestimated. Hence, the full overlap case is an overestimation of what would be in practice the behavior of the real system with full overlap, whose timing behavior would be somewhere in the region between the empirical measurements for arbitrary overlaps and the estimated full-overlap measurements obtained with CleanET. Note also that overestimating r makes that, whenever we consider lower overlaps than in practice (e.g. basal case), execution times obtained are in practice *lowerbounds* of the true basal case. In any case, the basal case is irrelevant to validate whether time budgets allocated suffice.

For the sake of completeness, we provide in Figure 8.6 how execution times expected vary as we vary the degree of overlap (p). We use peak-over-threshold again, and we collect the values obtained at different exceedance levels (lvl), from 10^{-3} to 10^{-12}. As expected, a linear increase of the degree of overlap, leads to nearly-linear increase of the execution times expected, with small disturbances due to the uncertainty associated to threshold selection in the fitting process of a distribution (an exponential distribution in our case). For instance, based on the results obtained, if we consider an exceedance level of 10^{-6}, we would conclude that execution times could be slightly below 500ms (assuming full overlap being the worst-case during operation). Thus, validation tests would be passed if such value is lower than the deadline for the task under analysis.

Overall, CleanET allows exploring any arbitrary degree of overlap of interest, thus providing end users with means to validate their system against scenarios that cannot be triggered during system testing, thus relieving end users from having to create additional test cases with the hope of those test cases to produce the execution scenarios that need being considered.

8.5 Summary

The complexity of current MPSoCs is mainly due to the intricate interactions between their multiple hardware blocks in the execution of complex software like the one governing Autonomous Driving. The competition for resources in these hardware blocks impacts greatly the execution time of each task. In this work we propose cleanET, a mathematical model that expresses how tasks that overlap in time produce a time dilation on each other. CleanET not only can model the dilation factor of a given set of time data, it can also use that factor to estimate a worst-case scenario where the contention is at its peak. We used as case-study data from a real autonomous driving framework, Apollo, to assess the dilation factor of overlapping tasks. CleanET allows generating, apart from the worst-case full overlap scenario, a no overlap scenario so that we can observe a model for the execution times in isolation with only the data during system operation.

Part V

Conclusions and Future Work

Chapter 9

Conclusions and Future Work

9.1 Conclusions

The growth in the complexity of CRTES continues to challenge the real-time community to provide trustworthy timing analysis solutions. The execution of increasingly-complex mixed-critically software, like machine learning, on complex multicore hardware makes timing analysis, and providing evidence of correctness, difficult tasks. In order to address the complexity rise in CRTES, several measurement-based techniques have been proposed. In particular, measurement probabilistic timing analysis techniques based on Extreme Value Theory have been shown to deliver provably trustworthy WCET estimates on execution scenarios that have not been observed. In order for MBPTA techniques to be applied, the execution time occurrences under validation should be representative of all possible execution conditions and scenarios that may arise during system operation, hardware- and software-wise. The particularity of MBPTA resides in providing WCET bounds via measurements in the form of a probability distribution.

This thesis pushes the state-of-the-art in the MBPTA domain: We have proposed new tools to i) provide tighter and trustworthy pWCET estimates, ii) merge disjoint hardware event monitor data, and iii) assess contention from competing tasks. More specifically, we achieved the following:

- **Sky-high Quantile Estimation**: Light tails are the theoretical best fit for pWCET estimation. However, the lack of reliable fitting methods (for light tails as part of EVT) that deliver reliable pWCET estimates for arbitrarily low exceedance probabilities imposes the use of exponential tails, which although proven reliable, are increasingly pessimistic for decreasing exceedance probabilities. We proposed two different solutions for this problem.
  - We proved that risk analysis, where EVT is used, and survivability analysis tackle the same fundamental problem by predicting the occurrence of rare events. Then, building on distributions from survivability analysis, we proposed the use of Weibull tails (*tailW*), which are proven to be as reliable as reference log-concave distributions, but enabling the tight modelling of arbitrarily low exceedance probabilities.
  - We presented for the first time a method based on Markov's Inequality for pWCET estimation that represents a solid alternative approach to EVT. In particular, we showed that MIK (Markov Inequality to the power-of-*k*) has no model uncertainty and proposed a method to handle sampling uncertainty (RESTK) that consistently provides more trustworthy, tighter and stable results than EVT in different scenarios, including a railway case study. These promising results suggest that RESTK can be effectively used as a standalone method for pWCET estimation, or even as an alternative approach to validate EVT results in those cases where EVT is already consolidated. In this line, the fact that MIK (RESTK) and EVT build on completely different mathematical foundations provides stronger evidence on the trustworthiness of the obtained pWCET estimates.
- **HEM Data Merging**: Measurement-based timing analysis methods increasingly build on HEMs to measure and estimate the timing behavior of time-critical applications running on

MPSoCs. Unfortunately, in complex MPSoCs, HEM values are subject to some unavoidable noise. Besides, they can only be read in small subsets, as many as the number of PMCs the hardware has. As a result, the system designer can only collect partial snapshots (i.e. only a subset of HEMs) in every run, each potentially subject to different noise with different impact on HEMs. In this thesis we have proposed two solutions for this problem:

  – HRM, a flexible method to merge HEM values across runs that allows preserving precisely their correlation with timing and their joint correlation, to a good extent. HRM achieves its goals by building on (1) non-parametric statistics, which do not pose any constraint on the distributions observed for different HEMs in the T2080, and (2) the use of an anchor HEM to relate measurements from different HEMs.

  – MUCH, a method that works by observing pairwise HEM correlations and building on multivariate Gaussian distributions to generate merged HEM vectors where discrepancies between produced pairwise HEM correlations and real ones is very low. Our results on an NXP T2080 MPSoC used for avionics' CRTES support the effectiveness of MUCH and show that it outperforms HRM systematically, thus being HRM the best choice only in those cases where too many HEMs are needed and there are very few PMCs.

- **Contention Modelling:** Validating execution time bounds for CRTES becomes increasingly difficult due to the growing complexity of those systems driven by a higher system automation. This is particularly true in automotive systems with the advent of autonomous driving. This thesis proposes CleanET, a novel solution to obtain measurements representative of any relevant scenario from the set of measurements obtained during analysis. In particular, CleanET builds upon estimating the impact of execution time interference across tasks to allow resampling data into any degree of interference (i.e. execution overlap) of interest. CleanET is a purely measurement-based solution, thus in line with the requirements of automotive end users, that facilitates testing execution scenarios that cannot be practically produced by testing engineers. Our results on a commercial AD framework, Apollo, corroborate the advantages of CleanET for timing validation.

9.2 Future Work

In this thesis, we pioneered multiple methodologies that can be explored further. In particular, this thesis pushes the state-of-the-art forward in three main areas. Some possible future works on each of those areas are the following:

- **Sky-high Quantile Estimation:** This thesis introduces two new tail estimation methodologies that improve the state-of-the-art on the WCET problem. In both, *tailW* and Markov's Inequality works, we always work with i.i.d. data. Even if the data is not fully independent, as long as the dependence is weak and it is not present at the most extreme values, our methodologies will hold. In our work, the dependence we found is weak, and there was no dependence on the extremes. However, it can be interesting to explore these new methodologies specifically in the case of dependence on the extremes to provide with more robust guarantees in scenarios where such dependence on the extremes manifests. Note that, in none of the platforms and benchmarks we used this extreme dependence appeared. But, of course, we do not discard it appearing in other unexplored scenarios.

- **HEM Data Merging:** We proposed two solutions to the HEM merging problem. As explained, keeping the relationships between HEMs read in different runs is not an easy task. HRM solves this problem by means of statistics of order. Conversely, MUCH tackles this problem by using multivariate Gaussian copulas. Both methods deliver good results for HEM merging. However, both methods model the central part of the distribution rather than the extremes. As firsts steps into this new approach, they are satisfactory. However, increasing the accuracy for the extreme data can be particularly useful in the CRTES domain. In that regard, MUCH has the potential to be improved for an extreme modelling approach given that we can change the copula model for one that focuses on keeping the relationships in the tail. Additionally, there are some platforms with more constraints on HEM data gathering. In some cases, a particular group of HEMs is restricted to only be read through a specific PMC. Therefore those HEMs cannot be read simultaneously. If one were to implement HRM in this scenario those HEMs cannot function as an anchor properly. Although, HRM could still be used in those scenarios if one considers the execution time of a program (i.e. processor cycles) as the anchor given it is a free HEM and can be read through any PMC. Implementing MUCH in a scenario where two HEMs cannot be read at once is an interesting challenge that would generalize its applicability.

- **Contention Modelling:** In this thesis we proposed a model for the contention produced by tasks accessing shared resources at the same time. We proposed CleanET, a model devised from basic mathematical modelling principles. Once the model for the contention was established, we performed a linear fit to estimate its parameters. With these parameters we can estimate a WCET for the full overlap of the tasks. As shown, the model is conservative in nature and may yield pessimistic results for the full overlap scenario. Also, with a linear fit model it is modelling the central part of the distribution. Similarly to HRM and MUCH, CleanET is the first step into a more general model. The next step would be considering a model which takes into account the dependencies on the overlaps which produce the most dilation. As mentioned, the other aspect to work on is a tighter full overlap scenario to model the WCET.

A natural step for all the solutions developed in this thesis is increasing the maturity to enable industrial adoption. To ease this step, part of the work of the research group aims at the integration of these solutions into industrial use cases on commercial platforms, and into open source and commercial toolsets. Those activities are being conducted in the context of European projects just starting (e.g., Horizon Europe SAFEXPLAIN, coordinated by BSC), and some project proposals under submission or to be submitted soon. Also, a first effort towards easing the adoption of these solutions is the release as open source of the *distTails* package for sky-high quantile estimation.

Bibliography

[1] J. Abella, C. Hernandez, E. Quiñones, F. J. Cazorla, P. R. Conmy, M. Azkarate-askasua, J. Perez, E. Mezzetti, and T. Vardanega. WCET analysis methods: Pitfalls and challenges on their trustworthiness. In *International Symposium on Industrial Embedded Systems (SIES)*, pages 1–10, 2015.

[2] J. Abella, E. Mezzetti, and F. J. Cazorla. On assessing the viability of probabilistic scheduling with dependent tasks. In *Symposium On Applied Computing (SAC)*, 2019.

[3] J. Abella, M. Padilla, J. del Castillo, and F. J. Cazorla. Measurement-based worst-case execution time estimation using the coefficient of variation. *Transactions on Design Automation of Electronic Systems*, 22(4):72:1–72:29, 2017.

[4] J. Abella, E. Quiñones, F. Wartel, T. Vardanega, and F. J. Cazorla. Heart of Gold: Making the improbable happen to extend coverage in probabilistic timing analysis. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 255–265. IEEE, 2014.

[5] I. Agirre. *Development and certification of mixed-criticality embedded systems based on probabilistic timing analysis*. PhD thesis, UPC, Departament d'Arquitectura de Computadors, 2018.

[6] I. Agirre, F. J. Cazorla, J. Abella, C. Hernández, E. Mezzetti, M. Azkarate-askatsua, and T. Vardanega. Fitting software execution-time exceedance into a residual random fault in ISO-26262. *Transactions on Reliability*, 67(3):1314–1327, 2018.

[7] A.R. Alameldeen and D.A. Wood. Variability in architectural simulations of multi-threaded workloads. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 7–18, 2003.

[8] A. Alhammad, S. Wasly, and R. Pellizzoni. Memory efficient global scheduling of real-time tasks. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 285–296, 2015.

[9] V. Antinyan. Revealing the complexity of automotive software. In *Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 1525–1528, New York, NY, USA, 2020. ACM.

[10] C. Antoniadis, D. Garyfallou, N. Evmorfopoulos, and G. Stamoulis. Evt-based worst case delay estimation under process variation. In *Design, Automation Test in Europe (DATE)*, pages 1333–1338, 2018.

[11] Apollo. An open autonomous driving platform. `http://apollo.auto/`, 2018.

[12] L. F. Arcaro, K. P. Silva, R. S. de Oliveira, and L. Almeida. Reliability test based on a binomial experiment for probabilistic worst-case execution times. In *Real-Time Systems Symposium (RTSS)*, pages 51–62, 2020.

[13] Autoware. An open autonomous driving platform. `https://github.com/CPFL/Autoware/`, 2016.

[14] F. Balabdaoui, K. Rufibach, and J. A. Wellner. Limit distribution theory for maximum likelihood estimation of a log-concave density. *Annals of Statistics*, 37(3):1299–1331, 2009.

[15] A. A. Balkema and L. de Haan. Residual Life Time at Great Age. *The Annals of Probability*, 2(5):792 – 804, 1974.

[16] F. Bartolucci and L. Scrucca. Point estimation methods with applications to item response theory models. In Penelope Peterson, Eva Baker, and Barry McGaw, editors, *International Encyclopedia of Education (Third Edition)*, pages 366–373. Elsevier, Oxford, third edition edition, 2010.

[17] M. Becker, D. Dasari, B. Nicolic, B. Akesson, V. Nelis, and T. Nolte. Contention-free execution of automotive applications on a clustered many-core platform. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 14–24, 2016.

[18] P. Benedicte. *Smart hardware designs for probabilistically-analyzable processor architectures*. PhD thesis, UPC, Departament d'Arquitectura de Computadors, 2022.

[19] K. Berezovskyi, F. Guet, L. Santinelli, K. Bletsas, and E. Tovar. Measurement-based probabilistic timing analysis for graphics processor units. In *International Conference on Architecture of Computing Systems (ARCS)*, pages 223–236, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[20] K. Berezovskyi, L. Santinelli, K. Bletsas, and E. Tovar. WCET Measurement-Based and Extreme Value Theory Characterisation of CUDA Kernels. In *International Conference on Real-Time Networks and Systems (RTNS)*, page 279–288, New York, NY, USA, 2014. ACM.

[21] G. Bernat, A. Burns, and M. Newby. Probabilistic timing analysis: An approach using copulas. *Journal of Embedded Computing*, 1(2):179–194, 2005.

[22] G. Bernat, A. Colin, and S. M. Petters. WCET analysis of probabilistic hard real-time system. In *Real-Time Systems Symposium (RTSS)*, 2002.

[23] G. Bernat, A. Colin, and S.M. Petters. WCET analysis of probabilistic hard real-time systems. In *Real-Time Systems Symposium (RTSS)*, pages 279–288, 2002.

[24] A. Betts, N. Merriam, and G. Bernat. Hybrid measurement-based wcet analysis at the source level using object-level traces. In *International Workshop on Worst-Case Execution Time Analysis (WCET)*, pages 54–63, 2010.

[25] A. Biondi and M. Di Natale. Achieving predictable multicore execution of automotive applications using the let paradigm. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 240–250, 2018.

[26] J. Blanchet, F. He, and K. Murthy. On distributionally robust extreme value analysis. *Extremes*, 23:317–347, 2020.

[27] Catalina Bolancé, Carlos Alberto Acuña, and Salvador Torra. Non-normal market losses and spatial dependence using uncertainty indices. *Mathematics*, 10(8), 2022.

[28] G. E. P. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970.

[29] F. Caeiro and M. Gomes. Semi-parametric tail inference through probability-weighted moments. *Journal of Statistical Planning and Inference*, 141(2):937–950, 2011.

[30] F. Caeiro and M. Gomes. Threshold selection in extreme value analysis. *Extremes*, 2014.

[31] J. Cai, P. Wan, and G. Özel Kadilar. Parametric and non-parametric estimation of extreme earthquake event: the joint tail inference for mainshocks and aftershocks. *Extremes*, 24, 2021.

[32] J. Cardona, C. Hernandez, J. Abella, and F. J. Cazorla. Maximum-contention control unit (mccu): Resource access count and contention time enforcement. In *Design, Automation Test in Europe (DATE)*, pages 710–715, 2019.

[33] M. Casas, H. Servat, R. M. Badia, and J. Labarta. Extracting the optimal sampling frequency of applications using spectral analysis. *Concurrency and Computation: Practice and Experience*, 24(3):237–259, 2012.

[34] E. Castillo, A. Hadi, N. Balakrishnan, and J. Sarabia. *Extreme Value and Related Models With Applications in Engineering and Science*. Whiley, 2005.

[35] F. J. Cazorla, L. Kosmidis, E. Mezzetti, C. Hernandez, J. Abella, and T. Vardanega. Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. *Computing Surveys*, 52(1), 2019.

[36] F. J. Cazorla, E. Quiñones, T. Vardanega, L. Cucu-Grosjean, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim. Proartis: Probabilistically analyzable real-time systems. *Transactions on Embedded Computing Systems*, 12(2s), 2013.

[37] K. Chen and J. Chen. Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *International Symposium on Industrial Embedded Systems (SIES)*, pages 1–8, 2017.

[38] K. Chen, N. Ueter, G. von der Brüggen, and J. Chen. Efficient computation of deadline-miss probability and potential pitfalls. In *Design, Automation Test in Europe (DATE)*, pages 896–901, 2019.

[39] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[40] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

[41] M. Colnaric, D. Verber, and W. A. Halang. *Distributed Embedded Control Systems: Improving Dependability with Coherent Design*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[42] C. Cox. *Delta Method*. John Wiley & Sons, Ltd, 2005.

[43] D.R. Cox and D. Oakes. *Analysis of Survival Data*. Chapman and Hall, 1984.

[44] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla. Measurement-based probabilistic timing analysis for multi-path programs. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 91–101, 2012.

[45] J. Danielsson, L. de Haan, L. Peng, and C.G. de Vries. Using a bootstrap method to choose the sample fraction in tail index estimation. *Journal of Multivariate Analysis*, 76, 2001.

[46] D. Dasari, B. Andersson, V. Nelis, S. M. Petters, A. Easwaran, and J. Lee. Response time analysis of cots-based multicores considering the contention on the shared memory bus. In *International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1068–1075, 2011.

[47] D. Dasari and V. Nelis. An Analysis of the Impact of Bus Contention on the WCET in Multicores. In *International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, pages 1450–1457, 2012.

[48] R. Davis and L. Cucu-Grosjean. A survey of probabilistic schedulability analysis techniques for real-time systems. *Leibniz Transactions on Embedded Systems*, 6(1):04–1–04:53, 2019.

[49] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application*. Cambridge University Press, 1997.

[50] A. C. Davison and R. L. Smith. Models for exceedances over high thresholds. *Journal of the Royal Statistical Society. Series B (Methodological)*, 52(3):393–442, 1990.

[51] Ddc-i. Patent Details for Managing Cache. `https://www.ddci.com/manage_cache_patent/`, 2020.

[52] J. del Castillo, J. Daoudi, and R. Lockhart. Methods to distinguish between polynomial and exponential tails. *Scandinavian Journal of Statistics*, 41(2):382–393, 2014.

[53] J. del Castillo, D. Moriña, and I. Serra. *ercv: Fitting Tails by the Empirical Residual Coefficient of Variation*, 2017. R package version 1.0.0.

[54] Deloitte. *Semiconductors – the Next Wave Opportunities and winning strategies for semiconductor companies*, 2019.

[55] E. Díaz, E. Mezzetti, L. Kosmidis, J. Abella, and F. J. Cazorla. Modelling multicore contention on the aurixtm tc27x. In *Design Automation Conference (DAC)*, 2018.

[56] J. L. Diaz, D. F. Garcia, K. Kim, C. Lee, L. Lo Bello, J. M. Lopez, S. Min, and O. Mirabella. Stochastic analysis of periodic real-time systems. In *Real-Time Systems Symposium (RTSS)*, pages 289–300, 2002.

[57] B. Dreyer, C. Hochberger, A. Lange, S. Wegener, and A. Weiss. Continuous non-intrusive hybrid wcet estimation using waypoint graphs. In *International Workshop on Worst-Case Execution Time Analysis (WCET)*, 2016.

[58] L. Duembgen, A. Huesler, and K. Rufibach. Active set and EM algorithms for log-concave densities based on complete and censored data. Technical report, IMSV, Univ. of Bern, 2010.

[59] S. Edgar and A. Burns. Statistical analysis of wcet for scheduling. In *Real-Time Systems Symposium (RTSS)*, pages 215–224, 2001.

[60] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[61] R. Ernst. Codesign of embedded systems: status and trends. *Design Test of Computers*, 15(2):45–54, 1998.

[62] Federal Aviation Administration, Certification Authorities Software Team (CAST). *CAST-32A Multi-core Processors*, 2016.

[63] R. A. Fisher. Xv.—the correlation between relatives on the supposition of mendelian inheritance. *Transactions of the Royal Society of Edinburgh*, 52(2):399–433, 1919.

[64] R. A. Fisher. Moments and product moments of sampling distributions. *Proceedings of the London Mathematical Society*, s2-30(1):199–238, 1930.

[65] R. A. Fisher and L. H. C. Tippett. Limiting forms of the frequency distribution of the largest or smallest member of a sample. *Mathematical Proceedings of the Cambridge Philosophical Society*, 24(2):180–190, 1928.

[66] D. Freedman, R. Pisani, and R. Purves. *Statistics: Fourth International Student Edition*. W.W. Norton & Company, 2007.

[67] Freescale semicondutor. ARM® architecture reference manual. armv8, for armv8-a architecture profile. ARM DDI 0487F.b (ID040120).

[68] Freescale semicondutor. e6500 Core Reference Manual. `https://www.nxp.com/docs/en/reference-manual/E6500RM.pdf`, 2014. E6500RM. Rev 0. 06/2014.

[69] Freescale semicondutor. QorIQ T2080 Reference Manual, 2016. Also supports T2081. Document Number: T2080RM. Rev. 3, 11/2016.

[70] M. Fusi, F. Mazzocchetti, A. Farres, L. Kosmidis, R. Canal, F. J. Cazorla, and J. Abella. On the use of probabilistic worst-case execution time estimation for parallel applications in high performance systems. *Mathematics*, 8(3), 2020.

[71] S. Jiménez Gil, I. Bate, G. Lima, L. Santinelli, A. Gogonel, and L. Cucu-Grosjean. Open challenges for probabilistic measurement-based worst-case execution time. *Embedded Systems Letters*, 9(3):69–72, 2017.

[72] S. Girbal, M. Moretó, A. Grasset, J. Abella, E. Quiñones, F. J. Cazorla, and S. Yehia. On the convergence of mainstream and mission-critical markets. In *Design Automation Conference (DAC)*, New York, NY, USA, 2013. ACM.

[73] T. Grass, A. Rico, M. Casas, M. Moreto, and A. Ramirez. Evaluating execution time predictability of task-based programs on multi-core processors. In *Parallel Processing Workshops*, pages 218–229. Springer International Publishing, 2014.

[74] D. Griffin and A. Burns. Realism in Statistical Analysis of Worst Case Execution Times. In Björn Lisper, editor, *International Workshop on Worst-Case Execution Time Analysis (WCET)*, volume 15, pages 44–53, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. The printed version of the WCET'10 proceedings are published by OCG (www.ocg.at) - ISBN 978-3-85403-268-7.

[75] D. Griffin, B. Lesage, I. Bate, F. Soboczenski, and R. I. Davis. Forecast-based interference: Modelling multicore interference from observable factors. In *International Conference on Real-Time Networks and Systems (RTNS)*, page 198–207, New York, NY, USA, 2017. ACM.

[76] F. Guet, L. Santinelli, and J. Morio. On the Reliability of the Probabilistic Worst-Case Execution Time Estimates. In *Embedded Real-time Software and Systems (ERTS^2) Conference*, 2016.

[77] F. Guet, L. Santinelli, and J. Morio. Probabilistic analysis of cache memories and cache memories impacts on multi-core embedded systems. In *Symposium on Industrial Embedded Systems (SIES)*, pages 1–10, 2016.

[78] F. Guet, L. Santinelli, and J. Morio. On the representativity of execution time measurements: Studying dependence and multi-mode tasks. In *International Workshop on Worst-Case Execution Time Analysis (WCET)*, pages 3:1–3:13, 2017.

[79] Y. Guo, T. Sayed, L. Zheng, and M. Essa. An extreme value theory based approach for calibration of microsimulation models for safety analysis. *Simulation Modelling Practice and Theory*, 106:102172, 2021.

[80] M. Gyllenhammar, R. Johansson, F. Warg, D. Chen, H. Heyn, M. Sanfridson, J. Söderberg, A. Thorsén, and S. Ursing. Towards an operational design domain that supports the safety argumentation of an automated driving system. In *European Congress on Embedded Real Time Systems (ERTS)*, 2020.

[81] P. R. Halmos. The Theory of Unbiased Estimation. *The Annals of Mathematical Statistics*, 17(1):34 – 43, 1946.

[82] J. Hansen, S. Hissam, and G. A. Moreno. Statistical-Based WCET Estimation and Validation. In Niklas Holsti, editor, *International Workshop on Worst-Case Execution Time Analysis (WCET)*, volume 10, pages 1–11, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. also published in print by Austrian Computer Society (OCG) with ISBN 978-3-85403-252-6.

[83] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *J. Mach. Learn. Res.*, 16(1):3367–3402, 2015.

[84] M. L. Hazelton. Assessing log-concavity of multivariate densities. *Statistics and Probability Letters*, 81(1):121–125, 2011.

[85] C. Hernández, J. Abella, F. J. Cazorla, A. Bardizbanyan, J. Andersson, F. Cros, and F. Wartel. Design and Implementation of a Time Predictable Processor: Evaluation With a Space Case Study. In Marko Bertogna, editor, *Euromicro Conference on Real-Time Systems (ECRTS)*, volume 76, pages 16:1–16:23, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[86] B. M. Hill. A Simple General Approach to Inference About the Tail of a Distribution. *The Annals of Statistics*, 3(5):1163 – 1174, 1975.

[87] R. V. Hogg and E. A. Tanis. *Probability and statistical inference*. Prentice Hall, 1997.

[88] R. Hyndman and Y. Fan. Sample quantiles in statistical packages. *The American Statistician*, 50(4):361–365, 1996.

[89] International Organization for Standardization. *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.

[90] J. Daoudi J. del Castillo and R. Lockhart. Methods to distinguish between polynomial and exponential tails. *Scandinavian Journal of Statistics*, 41(2):382–393, 2014.

[91] A. F. Jenkinson. The frequency distribution of the annual maximum (or minimum) values of meteorological elements. *Quarterly Journal of the Royal Meteorological Society*, 81(348):158–171, 1955.

[92] N. L. Johnson. *Continuous univariate distributions. Vol. 2.* Wiley and Sons, New York, 2nd ed. / norman l. johnson, samuel kotz, n. balakrishnan. edition, 1994.

[93] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar. Bounding memory interference delay in cots-based multi-core systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 145–154.

[94] T. Kloda, M. Solieri, R. Mancuso, N. Capodieci, P. Valente, and M. Bertogna. Deterministic memory hierarchy and virtualization for modern multi-core embedded systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–14, 2019.

[95] M. Kolmogorov. Sulla determinazione empírica di uma legge di distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 1933.

[96] H. Kopetz. The complexity challenge in embedded system design. In *International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 3–12, 2008.

[97] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. A cache design for probabilistically analysable real-time systems. In Enrico Macii, editor, *Design, Automation and Test in Europe (DATE)*, pages 513–518. EDA Consortium San Jose, CA, USA / ACM DL, 2013.

[98] L. Kosmidis, E. Quiñones, J. Abella, G. Farrall, F. Wartel, and F. J. Cazorla. Containing timing-related certification cost in automotive systems deploying complex hardware. In *Design Automation Conference (DAC)*, page 1–6, New York, NY, USA, 2014. ACM.

[99] L. Kosmidis, E. Quiñones, J. Abella, T. Vardanega, C. Hernandez, A. Gianarro, I. Broster, and F. J. Cazorla. Fitting processor architectures for measurement-based probabilistic timing analysis. *Microprocessors and Microsystems*, 47:287–302, 2016.

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

[101] B. Light. Concentration inequalities using higher moments information. *arXiv*, 2020.

[102] H. W. Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402, 1967.

[103] R. V. Lim. Computationally efficient multiplexing of events on hardware counters. In *Linux Symposium*, 2014.

[104] G. Lima and I. Bate. Valid Application of EVT in Timing Analysis by Randomising Execution Time Measurements. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 187–198, 2017.

[105] G. Lima, D. Dias, and E. Barros. Extreme value theory for estimating task execution time bounds: A careful look. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 200–211, 2016.

[106] J. S. Liu and S. M. Shahrier. On predictability of caches for real-time applications. In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOT)*, pages 52–56, 1994.

[107] L. Liu, Z. Cui, M. Xing, Y. Bao, M. Chen, and C. Wu. A software memory partition approach for eliminating bank-level interference in multicore systems. In *Pact*, pages 367–376. ACM, 2012.

[108] G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, 1978.

[109] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean. A new way about using statistical analysis of worst-case execution times. *SIGBED Rev.*, 8(3):11–14, 2011.

[110] R. Mancuso, R. Dudko, E. Betti, M. Cesati, M. Caccamo, and R. Pellizzoni. Real-time cache management framework for multi-core architectures. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 45–54. IEEE, 2013.

[111] A. Markov. On certain applications of algebraic continued fractions. *Ph.D. thesis, St. Petersburg*, 1884.

[112] A.W. Marshall and I. Olkin. *Life distributions. Structure of Nonparametric, Semiparametric, and Parametric Families*. Springer, 2007.

[113] Maspatechnologies. Maspatechnologies Helps Airbus Pass Multicore Certification on NXP T2080. `https://maspatechnologies.com/wp-content/uploads/2022/08/A3R_MulticoreCertification_ADSMPT.pdf`, 2022.

[114] A. Melani, E. Noulard, and L. Santinelli. Learning from probabilities: Dependences within real-time systems. In *Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8, 2013.

[115] E. Mezzetti, L. Kosmidis, J. Abella, and F. J. Cazorla. High-integrity performance monitoring units in automotive chips for reliable timing v v. *Micro*, 38(1):56–65, 2018.

[116] T. Mikosch. Regular variation, subexponentiality and their applications in probability theory. *International Journal of Production Economics*, 1999.

[117] S. Milutinovic. *On the limits of probabilistic timing analysis*. PhD thesis, UPC, Departament d'Arquitectura de Computadors, 2019.

[118] S. Milutinovic, E. Mezzetti, J. Abella, T. Vardanega, and F. J. Cazorla. On uses of Extreme Value Theory fit for industrial-quality WCET analysis. In *International Symposium on Industrial Embedded Systems (SIES)*, pages 1–6. IEEE, 2017.

[119] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. Sweeney. Producing wrong data without doing anything obviously wrong! volume 44, pages 265–276, 2009.

[120] T. Naoi, Y. Kagawa, K. Nagino, S. Niwa, and K. Hayashi. Extreme value analysis of the velocity of axonal transport by kinesin and dynein. *Biophysical Journal*, 121(3, Supplement 1):400a, 2022.

[121] R. Neill, A. Drebes, and A. Pop. Fuse: Accurate multiplexing of hardware performance counters across executions. *Transactions on Architecture and Code Optimization*, 14(4), 2017.

[122] R. B. Nelsen. *An Introduction to Copulas*. Springer Publishing Company, Incorporated, 2010.

[123] J. Neyman, E. Pearson, and K. Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

[124] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337, 1933.

[125] J. Nowotsch, M. Paulitsch, D. Bühler, H. Theiling, S. Wegener, and M. Schmidt. Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 109–118, 2014.

[126] J. Nowotsch, M. Paulitsch, D. Bühler, H. Theiling, S. Wegener, and M. Schmidt. Multi-core interference-sensitive WCET analysis leveraging runtime resource capacity enforcement. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 109–118, 2014.

[127] S. Osborne and J. H. Anderson. Simultaneous multithreading and hard real time: Can it be safe? In Marcus Völp, editor, *Euromicro Conference on Real-Time Systems (ECRTS)*, volume 165, pages 14:1–14:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[128] X. Pan and F. Mueller. Controller-aware memory coloring for multicore real-time systems. In *Symposium on Applied Computing (SAC)*, pages 584–592. ACM, 2018.

[129] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, and R. Kegley. A predictable execution model for COTS-based embedded systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 269–279, 2011.

[130] R. Pellizzoni, B. D. Bui, M. Caccamo, and L. Sha. Coscheduling of CPU and I/O Transactions in COTS-Based Embedded Systems. In *Real-Time Systems Symposium (RTSS)*, pages 221–231, 2008.

[131] R. Pellizzoni, A. Schranzhofer, J. Chen, M. Caccamo, and L. Thiele. Worst case delay analysis for memory interference in multicore systems. In *Design, Automation Test in Europe (DATE)*, pages 741–746, 2010.

[132] J. Pickands. Statistical inference using extreme order statistics. *The Annals of Statistics*, 3(1):119–131, 1975.

[133] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[134] D. Radack, H. G. Tiedeman Jr, and P. J. Parkinson. Civil Certification of Multi-core Processing Systems in Commercial Avionics, 2018.

[135] A. Ravindar and Y. N. Srikant. Estimation of probabilistic bounds on phase CPI and relevance in WCET analysis. In *International Conference on Embedded Software*, page 165–174, New York, NY, USA, 2012. ACM.

[136] B. Recht. A simpler approach to matrix completion. *J. Mach. Learn. Res.*, 12(null):3413–3430, 2011.

[137] F. Reghenzani, G. Massari, and W. Fornaciari. Probabilistic-WCET reliability: Statistical testing of EVT hypotheses. *Microprocess. Microsystems*, 77:103–135, 2020.

[138] F. Reghenzani, L. Santinelli, and W. Fornaciari. Dealing with uncertainty in pWCET estimations. *Transactions on Embedded Computing Systems*, 19(5):33:1–33:23, 2020.

[139] RTCA and EUROCAE. *DO-178C / ED-12C, Software Considerations in Airborne Systems and Equipment Certification*, 2011.

[140] C. El Salloum, M. Elshuber, O. Höftberger, H. Isakovic, and A. Wasicek. The across mpsoc – a new generation of multi-core processors designed for safety-critical embedded systems. In *Euromicro Conference on Digital System Design*, pages 105–113, 2012.

[141] L. Santinelli, F. Guet, and J. Morio. Revising measurement-based probabilistic timing analysis. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 199–208, 2017.

[142] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart. On the Sustainability of the Extreme Value Theory for WCET Estimation. In Heiko Falk, editor, *International Workshop on Worst-Case Execution Time Analysis (WCET)*, volume 39, pages 21–30, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[143] A. Satyanarayana. Intelligent sampling for big data using bootstrap sampling and Chebyshev inequality. In *Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6, 2014.

[144] K. Schmidt, D. Marx, J. Harnisch, A. Mayer, U. Dannebaum, and H. Christlbauer. Non-intrusive tracing at first instruction. In *SAE Technical Paper*. SAE International, 2015.

[145] A. Schranzhofer, R. Pellizzoni, J. J. Chen, L. Thiele, and M. Caccamo. Worst-case response time analysis of resource access models in multi-core systems. In *Design Automation Conference (DAC)*, pages 332–337, 2010.

[146] J. Serrà, A. Corral, M. Boguñá, M. Haro, and J. Ll. Arcos. Measuring the evolution of contemporary western popular music. *Scientific reports*, 2(1):1–6, 2012.

[147] H. Servat, G. Llort, J. Giménez, K. Huck, and J. Labarta. Folding: detailed analysis with coarse sampling. In *Tools for High Performance Computing 2011*, pages 105–118. Springer, 2012.

[148] D. Shapiro. Introducing xavier, the nvidia ai supercomputer for the future of autonomous transportation. *NVIDIA blog*, 2016.

[149] K. P. Silva, L. F. Arcaro, and R. Silva De Oliveira. On Using GEV or Gumbel Models When Applying EVT for Probabilistic WCET Estimation. In *Real-Time Systems Symposium (RTSS)*, pages 220–230, 2017.

[150] N. Smirnov. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. *Moscow University Mathematics Bulletin*, 1939.

[151] D. Sornette. *Critical Phenomena in Natural Sciences: Chaos, Fractals, Selforganization and Disorder: Concepts and Tools*. Springer, 2006.

[152] S. M. Steinberg and C. E. Davis. Distribution-free confidence intervals for quantiles in small samples. *Communications in Statistics - Theory and Methods*, 14(4):979–990, 1985.

[153] Z. Stephenson, J. Abella, and T. Vardanega. Supporting industrial use of probabilistic timing analysis with explicit argumentation. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 734–740, 2013.

[154] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

[155] N. Suzuki, H. Kim, D. de Niz, B. Andersson, L. Wrage, M. H. Klein, and R. Rajkumar. Coordinated bank and cache coloring for temporal protection of memory accesses. In *CSE*, pages 685–692. IEEE Computer Society, 2013.

[156] P. Tchebichef. Des valeurs moyennes. *Journal de mathématiques pures et appliquées*, 12(2):177–184, 1867.

[157] D. Trilla. *Non-functional considerations of time-randomized processor architectures*. PhD thesis, UPC, Departament d'Arquitectura de Computadors, 2020.

[158] Udacity. An Open Source Self-Driving Car. `https://github.com/udacity/self-driving-car/`, 2017.

[159] `http://www.gaisler.com/index.php/products/processors/leon3`. *Leon3 Processor*.

[160] V. Utkin. Calculating the reliability of machine parts on the basis of the Chebyshev inequality. *Russian Engineering Research*, 32, 2012.

[161] P. Kumar Valsan, H. Yun, and F. Farshchi. Taming non-blocking caches to improve isolation in multicore real-time systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–12, 2016.

[162] S. H. VanderLeest and C. Evripidou. An approach to verification of interference concerns for multicore systems (CAST-32A). In *SAE Technical Paper*. SAE International, 2020.

[163] G. V.G. Baranoski, J. G. Rokne, and G. Xu. Applying the exponential Chebyshev inequality to the nondeterministic computation of form factors. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 69(4):447–467, 2001.

[164] S. Vilardell and À. Pineda. *distTails: A Collection of Full Defined Distribution Tails*, 2019. R package version 0.1.2.

[165] S. Vilardell, I. Serra, J. Abella, J. del Castillo, and F. J. Cazorla. Software timing analysis for complex hardware with survivability and risk analysis. In *International Conference on Computer Design (ICCD)*, pages 227–236, 2019.

[166] S. Vilardell, I. Serra, R. Santalla, E. Mezzetti, J. Abella, and F. J. Cazorla. Hrm: merging hardware event monitors for improved timing analysis of complex mpsocs. *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

[167] G. von der Brüggen, N. Piatkowski, K. Chen, J. Chen, and K. Morik. Efficiently approximating the probability of deadline misses in real-time systems. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2018.

[168] B. von Mises. Fundamentalsätze der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 4:1–97, 1919.

[169] H. Wang and K. J. Jonas. The likelihood of severe covid-19 outcomes among plhiv with various comorbidities: a comparative frequentist and bayesian meta-analysis approach. *Journal of the International AIDS Society*, 24(11):e25841, 2021.

[170] F. Wartel, L. Kosmidis, A. Gogonel, A. Baldovin, Z. R. Stephenson, B. Triquet, E. Quiñones, C. Lo, E. Mezzetti, I. Broster, J. Abella, L. Cucu-Grosjean, T. Vardanega, and F. J. Cazorla. Timing analysis of an avionics case study on complex hardware/software platforms. In Wolfgang Nebel and David Atienza, editors, *Design, Automation and Test in Europe (DATE)*, pages 397–402. ACM, 2015.

[171] V. M. Weaver, D. Terpstra, and S. Moore. Non-determinism and overcount on modern hardware performance counter implementations. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 215–224, 2013.

[172] 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. *Transactions on Embedded Computing Systems*, 7(3):36:1–36:53, 2008.

[173] 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. *Transactions on Embedded Computing Systems*, 7(3), 2008.

[174] D.J. Wilkins. The bathtub curve and product failure behavior, part one: The bathtub curve, infant mortality and burn-in. *Reliability Hotwire*, (21), 2002.

[175] S. S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, 1938.

[176] S. N. Wood. *Core Statistics*. Cambridge University Press, 2015.

[177] J. Worms and S. Touati. Parametric and Non-Parametric Statistics for Program Performance Analysis and Comparison. *[Research Report] RR-8875, INRIA Sophia Antipolis - I3S; Université NiceSophia Antipolis; Université Versailles Saint Quentin en Yvelines; Laboratoire de mathématiques deVersailles*, 2017.

[178] J. Worms and S. Touati. Modelling program's performance with gaussian mixtures for parametric statistics. *Transactions on Multi-Scale Computing Systems (TMSCS)*, 4(3):383–395, 2018.

[179] Xilinx. Zynq ultrascale+ device technical reference manual. `https://docs.xilinx.com/v/u/en-US/ug1085-zynq-ultrascale-trm`, 2020.

[180] H. Xu, Q. Wang, S. Song, L. K. John, and X. Liu. Can we trust profiling results? understanding and fixing the inaccuracy in modern profilers. In *International Conference on Supercomputing*, page 284–295, New York, NY, USA, 2019. ACM.

[181] X. Yan and X. G. Su. *Linear Regression Analysis: Theory and Computing*, chapter 2. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2009.

[182] H. Yun, R. Mancuso, Z. P. Wu, and R. Pellizzoni. PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 155–166. IEEE, 2014.

[183] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha. Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 55–64, 2013.

[184] D. Zaparanuks, M. Jovic, and M. Hauswirth. Accuracy of performance counter measurements. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 23–32, 2009.

[185] P. G. Zaykov and J. Kubalčik. Worst-case measurement-based statistical tool. In *Aerospace Conference*, pages 1–10, 2019.

[186] M. Ziccardi, E. Mezzetti, T. Vardanega, J. Abella, and F. J. Cazorla. Epc: Extended path coverage for measurement-based probabilistic timing analysis. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 338–349, 2015.

[187] D. Ziegenbein and A. Hamann. Timing-aware control software design for automotive systems. In *Design Automation Conference (DAC)*, pages 56:1–56:6, New York, NY, USA, 2015. ACM.