POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA

DOCTORAL PROGRAMME IN COMPUTER SCIENCE AND ENGINEERING

BEYOND THE TRADITIONAL ANALYSES AND RESOURCE MANAGEMENT IN REAL-TIME SYSTEMS

Doctoral Dissertation of:

Federico Reghenzani

Supervisor:

Prof. **William Fornaciari**

Tutor:

Prof. **Cristina Silvano**

The Chair of the Doctoral Program:

Prof. **Barbara Pernici**

Ph.D. Cycle XXXII

First version: October 15th, 2020
Revised version: December 21st, 2020
Ph.D. defense date: January 11st, 2021

Acknowledgments

THROUGHOUT these years of my PhD studies, I have received invaluable professional and moral support from colleagues, friends, and many other people I met in this journey, to whom I am sincerely grateful. I heartily think that every single meeting, discussion, chat, laugh, and confrontation has been a priceless occasion to improve myself, even if I met some of these people just one time on the other side of the world. To thank all of them would be very difficult in this short space. For this reason, in the following paragraphs, I limit the acknowledgments to the people who had a major influence on my life, but my gratitude is also extended to the broader circle mentioned above.

I would first like to thank my advisor, Prof. William Fornaciari, whose expertise and competencies guided me in the research path that led to the writing of this thesis. His suggestions and advice helped my growth as a scientist and also as a person. I thank William to involve me in an international research context, give me the possibility to participate at conferences, project meetings, summer schools, and student exchanges around the globe, which incentivized me to widen my research perspectives and to improve my researcher skills that led to the writing of this thesis.

To my colleagues of HEAPLab, with whom I spent most of the time of these years, I express my gratitude and appreciation. The laboratory has been not only a place to share and discuss ideas, visions, and thoughts, but also a group of people to have fun and

relax with. Collaboration is an essential component for the researcher's growth, and an amusing collaboration is something precious but rare to find. I would like to particularly thank Giuseppe and Michele, my historical office mates since I was a master's student.

During the last years, I had the opportunity to visit ONERA (France) and University of Central Florida (United States). I sincerely thank the scholars of these institutions, respectively Dr. Luca Santinelli and Prof. Zhishan Guo, who welcomed and hosted me in their labs for my short PhD visits. People here have been genuinely nice, and I am glad to have kept the collaboration active even today.

I would like to thank the reviewers for their effort in reviewing this thesis and the PhD committee, which verified my research progress every year. Their suggestions were useful to improve my research and, consequently, this final document.

I wish to express my deepest gratitude to my parents, Manuela and Mauro, who supported me during these years and pushed me to study and never give up. I always have my family to count on, also during these difficult pandemic times. Last but definitely not least, a special thanks goes out to my girlfriend, Lara, for the relentless support during these years. She kept me going on, and this path would not have been possible without her input. Approving my difficult choices, tolerating my long abroad periods and long work sessions, giving up some weekends, and many other everyday aspects, are not common traits, and I feel very fortunate to have found them in Lara.

Abstract

ESTIMATING the worst-case execution time in modern architecture is a non-trivial problem. The presence of complex hardware features, necessary to improve the computational power due to the single-core performance barrier, makes this estimation difficult or even impossible if traditional techniques are employed. Probabilistic real-time and, in particular, the measurement-based probabilistic timing analysis have been developed to overcome this issue. Measurement-based approaches are very appealing for the industrial world, because they enable to obtain a statistical estimation of the worst-case execution time by directly observing the execution time of the real system. Unfortunately, the current status of probabilistic real-time is far to be acceptable for certification purposes of safety-critical systems. In this thesis, both theoretical and experimental works, related to the PhD activities, are presented, with a special focus on the uncertainty estimation and the applicability of the probabilistic real-time techniques to real platforms. Both these problems have been studied, providing theoretical tools and experimental evaluations to advance probabilistic approaches towards a safe solution to the worst-case execution time estimation problem. The issues, which still limit the use of such techniques in industrial environment, are discussed and possible future research directions presented. In addition, unconventional exploitations of probabilistic real-time are proposed for mixed-criticality, high performance computing, and energy-constrained systems. During all of these works, some software tools and datasets have been developed, released as open-source and described in this thesis.

Abstract (in Italian)

STIMARE il tempo massimo di esecuzione nelle architetture dei calcolatori moderni è impegnativo. A causa della necessità di andare oltre la potenza di calcolo singolo-core ormai sostanzialmente in stallo, l'introduzione di funzionalità complesse nell'hardware rende il problema difficile o addirittura impossibile da risolvere con gli approcci classici. Il real-time probabilistico, e in particolare le analisi probabilistiche del tempo di esecuzione basate su misure, sono stati recentemente sviluppati per ovviare a questa difficoltà. Avere a disposizione degli approcci basati su misure sarebbe molto interessante per il mondo industriale, perché consentono di ottenere una stima probabilistica del tempo massimo di esecuzione osservando direttamente il tempo di esecuzione sul sistema reale. Sfortunatamente, allo stato attuale, il real-time probabilistico è lontano dall'essere accettabile nel processo di certificazione di un sistema safety-critical. Questa tesi presenta i lavori teorici e sperimentali relativi al dottorato dell'autore, focalizzandosi in particolare sugli aspetti di stima dell'incertezza e applicabilità delle tecniche di real-time probabilistico a piattaforme reali. Entrambi questi problemi sono stati studiati, fornendo sia strumenti teorici che presentando valutazioni sperimentali, allo scopo di migliorare gli approcci probabilistici verso una soluzione atta a calcolare in modo affidabile il tempo massimo di esecuzione. Nel presente elaborato vengono prima discussi i loro limiti, che ancora oggi relegano il loro uso ad ambienti prettamente accademici, e presentate delle possibili linee di ricerca future. Inoltre, sono proposti usi non convenzionali del real-time probabilistico nell'ambito di

sistemi a criticalità mista, high-performance computing e con energia limitata. Durante il progresso di queste ricerche, sono stati sviluppati e pubblicati in open-source diversi software e dataset descritti in questa tesi.

Contents

CHAPTER 1

Introduction

Since the beginning of the 2000s, computing platforms shifted towards more complex architectures because of the increasing computing power demand not achievable with simple and single-core processors. This introductory chapter describes this trend, the ensuing modeling problems, and the impact on both embedded and high-performance computing systems, with a particular focus on time-sensitive systems.

1.1 The Evolution of Hardware Platforms

The increasing computational performance demand of applications in recent years leads to the evolution of computing platforms towards complex processor architectures and sophisticated system components. This trend is mainly caused by the single-core performance barrier. For years, the CPU manufacturers worked to increase the clock fre-

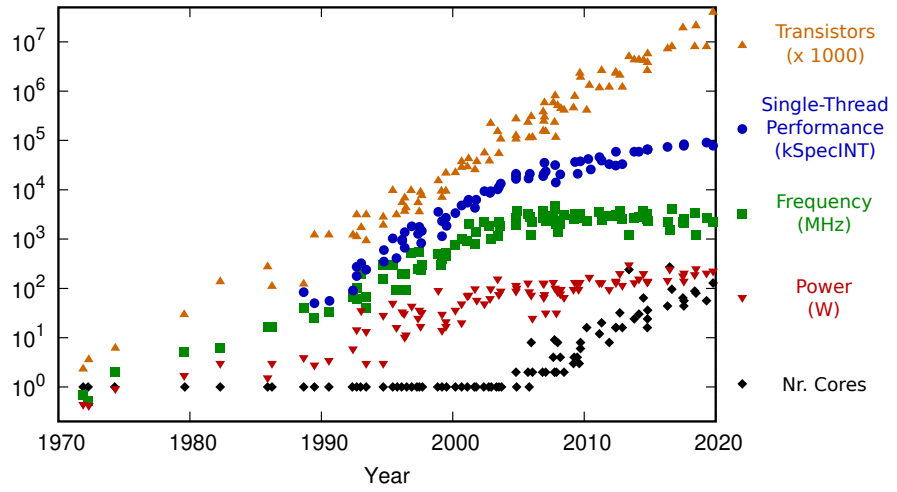


Figure 1.1: *The evolution of microprocessor characteristics in the last 48 years. Data by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, C. Batten, and K. Rupp*[2].

quency and to exploit instruct-level parallelism (ILP), by making the transistors smaller and smaller until reaching the current limit of 7nm[1]. Robert H. Dennard predicted in 1974 that the power density (amount of power per area) of the processors remains constant over the years [70]. However, around 2006-2007, this scaling law appeared to be no more valid, and the linear trend of the power density started to decrease towards a plateau. The same happened for frequency and ILP performance gain. The consequence of these limitations is today's difficulty in improving the single-core performance at the same rate as the previous decades. To overcome this barrier, manufacturers have started introducing several advanced features, primarily multi-core, but also complex pipelines, multi-level caches, memory prefetcher, and many other mechanisms and architectural features.

Figure 1.1 perfectly depicts these trends. The Moore's law – i.e., the number of transistors doubles approximately every 2 years – is still valid in 2020, but it is expected

[1]Smaller transistor sizes are affected by quantum effects. Some manufactures are developing smaller transistors (5nm and 3nm), but they are not commercially available at the time of writing.

[2]Released under 'Creative Commons Attribution 4.0 International Public License'. Repository: `https://github.com/karlrupp/microprocessor-trend-data`

to end in the next few years [189]. However, the single-thread performance increased (sub-)linear in the last 15 years, and they are substantially constant looking at the last few years. Frequency reached its top value during the late 2000s and started to decrease to reduce the power consumption and dissipation. The power, in fact, has also stabilized in the last years. Since increasing single-thread performance and frequency is no longer a viable solution for processors' manufacturers, increasing the number of cores is currently the primary solution to respond to the performance demand, as clearly shown by the trend depicted in the figure.

1.1.1 Computing Continuum

This single-thread performance barrier, in addition to the explosion of computational power-hungry applications, changed the paradigms of the system development for both large-scale and small-scale computing. Nowadays, even small and mobile embedded systems need extensive computational capabilities to perform their tasks, e.g., image recognition or artificial intelligence algorithms. The order of magnitude of the number of software code lines in a modern car is hundreds of a million [51]. However, maintaining the miniaturization and, in particular, achieving the power and energy goals is difficult at small-scale, leading to the *computing continuum* concept: Thanks to the increasing interconnection of embedded devices to the global network, the devices "at the edge" can forward the heavy computational load to centralized clusters of machines. In this way, the lightweight computation is kept on the embedded systems, moving the most power-hungry computation to the cloud or High-Performance Computing (HPC) systems.

In this thesis, we focus on both embedded and HPC systems dealing with time constraints. The challenges to achieving timing predictability in modern architectures are shared in both system categories and also linked by the computing continuum concept. The rest of this chapter introduces the current trends of embedded systems (Section 1.2) and HPC (Section 1.3), followed by a description of time-sensitive systems (Section 1.4).

1.2 Embedded Systems

Numerous definitions of the term *embedded system* exist. An embedded system is commonly defined as a computing system built for a specific target application (opposed

to a general-purpose system), that is usually part of a larger system (e.g., a washing machine, a car, or an airplane). Nowadays, human society is pervaded by embedded systems which are more and more connected to the internet. There is an enormous number of possible examples, like the infotainment system of a car, the orbit control computer of a satellite, the interface of a printer, or the computer controlling a medical tomography machine, just to name a few.

The hardware platforms of embedded systems are diverse: they can be as simple as an 8-bit microcontroller or a multi-core machine with an operating system and software virtualization. Most all of the application of embedded systems share the same common goals: to be small, inexpensive, power/energy efficient, reliable, and with a low development cost.

1.2.1 Critical systems

An important segment of embedded systems is the *critical systems* category. In such systems, providing a correct result is essential due to the nature of their application. We can distinguish two classes:

- *Mission-critical systems*: the failure to correctly perform the intended function for which they have been developed may cause critical financial damage to the organization producing and/or using the system. Examples of this category include financial trading systems or unmanned spacecraft operations.
- *Safety-critical systems* (sometimes, but less commonly, called *Life-critical systems*): the failure to correctly perform the intended function for which they have been developed may result in: 1) death or serious injuries to people, 2) environmental disasters, or 3) severe damage to buildings or equipment. Examples of this category include avionics, nuclear power plant systems, and medical equipment.

Designing a critical system requires not only proper hardware and software components but also a complex company organization to trace the development activities to ensure the reliability of the final product. While standards for both software and hardware exist, no one can certify that the final system is exempt of bugs and design errors. *Testing shows the presence, not the absence of bugs* (Dijkstra, [74]). Formal methods, which can instead prove the logical correctness, are challenging when used to verify complex hardware devices [107], and they are usually feasible for only a small part of the software, due to the complexity of the state space exploration. Consequently, creating a

proper and documented development process that is built to reduce the risk of errors is the key of a critical system. This is especially important for safety-critical systems, which usually require the approval by a certification authority. The failure to create and maintain a proper development process and requirement tracing may lead to disastrous financial and human consequences. A recent example is the Boeing 737 MAX. The crashes of two airplanes of this model are attributed to inadequate requirements analyses and certification process of a flight control computer [127]. These tragedies cost 346 human lives and an estimated money loss, at the time of writing, of $20 billion for Boeing[3]. This sad episode perfectly serves as an example of how a safety-critical embedded system, even if it has a minor role in the global function of the machine, may have a tremendous impact on the correctness of the whole system and the organization developing it.

1.2.2 The Commercial Off-The-Shelf trend

Many industries use *Commercial Off-The-Shelf (COTS)* platforms to reduce the design cost of embedded systems, rather than to develop custom hardware and software. This is, usually, not the case for critical systems, where the strict rules for the development process hinder the possibility of using off-the-shelf hardware or software. However, a recent trend to using COTS components in critical systems can be observed: Applications are becoming more and more sophisticated and require a large amount of computational power, and it would be too costly to build custom systems from scratch. Companies like Boeing and Airbus are currently evaluating the possibility of using COTS components in their aerospace products [9, 131]. Specific technical committees – e.g., the IEC Technical Committee 107 for avionics – are developing standards for COTS to ensure the product quality and enable their certification.

COTS components may improve the time-to-market and reduce the design cost, but they present many challenges when used in critical systems, as we will later explain Section 1.4.3. The trend towards COTS software and hardware cannot be ignored, and it represents a further complexity in the analysis of embedded systems.

[3]Reference news source: `https://www.theguardian.com/business/2020/jan/29/boeing-puts-cost-of-737-max-crashes-at-19bn-as-it-slumps-to-annual-loss`.

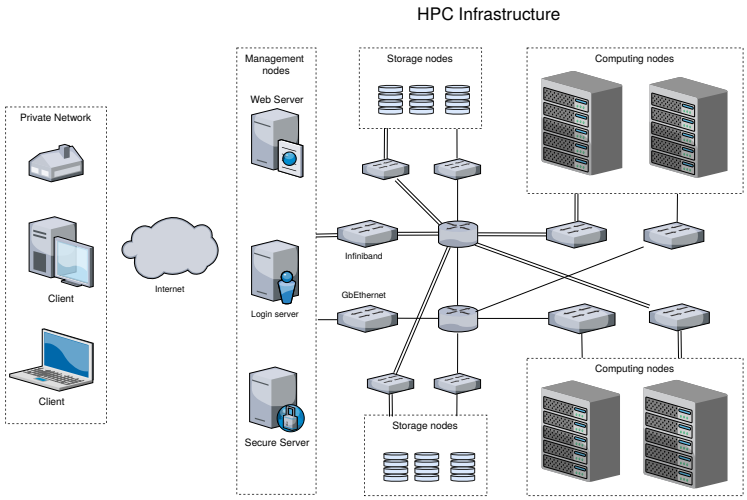


Figure 1.2: *An exemplification of the architecture of an HPC cluster.*

1.3 High Performance Computing

At the opposite side of the computing continuum, it is possible to find the *High–Performance Computing (HPC)*. This term refers to the technology of large clusters of interconnected machines, generally called *supercomputers*. The main goal of HPC clusters is to provide a computing infrastructure capable of fulfilling the increasing performance requirements of modern large-scale applications, in both scientific and industrial domains. The HPC infrastructure usually consists of clusters of thousands of computing, storage, and management nodes interconnected by a high-speed network, as sketched in Figure 1.2.

1.3.1 System and application goals

The software development for HPC systems is completely different from the embedded systems. However, many challenges, especially from the research perspective, are in common. Examples include task scheduling, parallel and distributed computing, and timing requirements. A work in this thesis, presented later in Section 5.3, focuses on the latter example. Regarding power and energy, HPC also has these two requirements. However, the focus is very different: while embedded systems necessitate optimizing

these metrics to keep the thermal and energy budgets contained, for HPC, these requirements are more related to the cost of the infrastructure. Large energy consumption has a direct impact on electricity costs. Large power dissipation requires powerful air-conditioning systems that, in turn, increase both the design and the running costs of electricity. The problem of energy dissipation is so important for HPC that the TOP500 organization[4] – which traditionally ranks the HPC systems by their performance – created a parallel ranking method, called Green500[5], to classify the systems based on performance/watt metric.

1.3.2 Programming models

To achieve high performance levels, computing nodes typically include multiple processors and multiple cores, because of the already described single-core performance barrier. Having both a distributed topology and high-performance multi-core processors, HPC systems allow the developers to scale the applications' performance by leveraging on both *inter-node* and *intra-node* parallelisms. To effectively exploit these parallelisms, proper software frameworks are required [8], including programming models (e.g., MPI, OpenMP), and management frameworks, like job schedulers and resource managers.

For heterogeneous HPC architectures (described in the next Section 1.3.3), an intermediate run-time framework to support programming models across a wide range of different accelerators becomes essential. Frameworks exposing basic services for communication, synchronization, and task spawning are needed, in addition to the high-level programming models built over the intermediate model, and specific primitives for each accelerator. Only with this complex structure, the programmer has all the tools he or she needs to achieve the maximum performance and exploit heterogeneous architectures.

1.3.3 Heterogeneous HPC

The hardware of HPC systems is traditionally uniform and homogeneous across the whole cluster and inside each node. In this way, management costs are reduced, and it makes software development easier because only one version of the application is

[4]Website: `https://www.top500.org/`
[5]Website: `http://www.green500.org/`

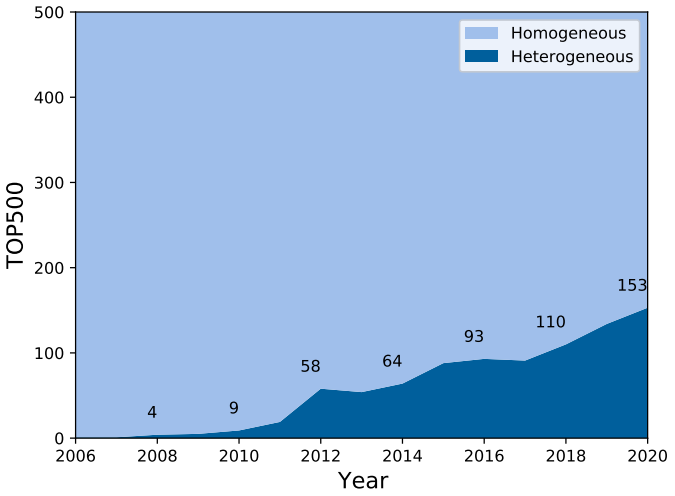


Figure 1.3: *The evolution of the number of heterogeneous HPC systems in the TOP500 list*[6].

needed. Moreover, the exploitation of parallelism is less challenging than having heterogeneous computational resources [72]. Nevertheless, from the 2010s, an increasing number of HPC systems started to include heterogeneous resources, as depicted in Figure 1.3 that shows the number of clusters in the TOP500 list exploiting heterogeneity. The achievement of Exascale capabilities (i.e., 10^{18} FLOPS[7]) requires not only adequate computational performance but also the meeting of power and energy budget constraints for the previously mentioned reasons. Heterogeneous architectures are a natural energy-efficient solution in this scenario [231], and they are promising to improve the aforementioned performance/watt metric.

The traditional approach towards resource management, which is essentially limited to assigning to each application a set of physical nodes at the job scheduler level, is not sufficient in such a scenario. This approach leaves many deeply heterogeneous resources unused, as each application typically employs only a subset of the available resources. To counter this effect and improve resource utilization, capacity computing

[6]Data source: `https://www.top500.org/statistics/list/`
[7]Floating-Point Operations per Second.

approaches allow multiple applications to coexist on the same node, sharing the heterogeneous resources based on each application's priority and Quality of Service (QoS) requirements. To achieve utilization goals and improve the QoS of applications, proper resource management protocols must be implemented. In this thesis, we focus on the time predictability requirement, and we propose resource management strategies based on embedded systems approaches.

1.4 Real-Time Systems

A *real-time system* is defined as a system whose correctness depends not only on the correctness of the results (*logical* or *functional correctness*) but also on the time constraints (*temporal correctness*).

1.4.1 Classification

Real-time applications (and by the extension, the systems on which they run on) can be further classified depending on the strictness of the requirements. Many different classifications exist, but the three categories commonly used are:

- *Hard real-time*: the time constraints of the tasks (usually expressed as deadlines) must be met under any condition. The failure to meet even one single deadline makes the system failed. Most of these applications are also safety-critical.
  - Examples: fly-by-wire computers, pacemaker controller, cars' engine control module, etc.
- *Firm real-time*[8]: the violation of a single time constraint does not cause the failure of the overall system, but a late computation makes the output of the applications useless, and it must be discarded.
  - Examples: financial trading software, weather forecast applications, etc.
- *Soft real-time*: the time constraints are not mandatory, and they are usually loosely defined (for instance, Quality of Service metrics). However, the violation of timing requirements should not happen "too often"[9].

[8] Different definitions of firm real-time exist, we selected the most common one for this thesis.

[9] This requirement has been voluntarily imprecisely defined: the exact definition of the requirement and how much is acceptable to violate it depends on the single model, system, and application.

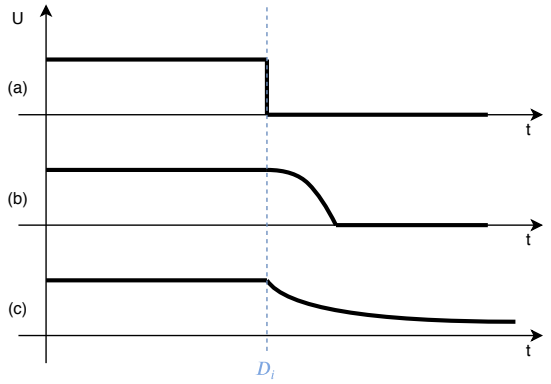


Figure 1.4: *Three examples of utility functions with respect to the deadline D_i: (a) hard constraint, the utility of the output is zero immediately if the task overruns its deadline; (b) soft constraint, the system is considered functioning for a certain period of time, in degraded mode, for a given time frame after the deadline; (c) soft constraint, the system is considered functioning with a decreasing level of utility.*

- Examples: multimedia players, simulation software, etc.

These categories can be generalized in the concept of *time-utility function* proposed in 1985 by Jensen et al. [138], which provides a general function model linking the execution time with the system utility. Three common examples are depicted in Figure 1.4. Case (c) is typical of soft real-time systems, while Case (a) is typical of a hard/firm real-time system, in which a deadline miss makes the system/output useless. Case (b) is, instead, a mixed-case: deadline can be missed, and the task exposes degraded performance, but only for a short and well-defined time interval. Having this general function makes it possible to formally express the fuzzy definition of soft real-time systems. The next chapter and, in particular, Section 2.4 provide the formal definitions for hard real-time systems.

Many embedded systems are also real-time systems (soft, firm, or hard). The safety-critical systems are usually hard real-time because the timeliness of the applications is often a requirement for such systems. For instance, an automotive airbag control system must react within a given time frame, and a soft real-time requirement is not sufficient to guarantee safety. However, real-time systems are not a perfect subset of embedded

systems because they can be applied to general-purpose machines or HPC, as shown in the next paragraphs.

1.4.2 Timing sensitive applications in HPC

The performance of HPC systems is typically measured in terms of *throughput*, i.e., the amount of work completed in a given time unit. The most common measurement unit for the throughput is the FLOPS. New application domains that require an HPC infrastructure to run are emerging. These include, for instance, use cases from automotive, smart city, healthcare, environmental, and infrastructure monitoring [95]. Application requirements in such domains include specific timing constraints, similar to real-time applications running on embedded systems. To show these new requirements, we provide in the next paragraphs some examples of real world applications.

Example – Autonomous driving. The automotive world, with the Autonomous Vehicles (AV) horizon, provides probably the most straightforward and one of the most challenging examples. From the computational perspective, an AV is equipped with a sensing system, which provides a large amount of data on the environment, nearby vehicles, pedestrians, and other objects. Such data must be processed to carry out prompt control actions (e.g., steering or braking). This processing requires the execution of a large number of computationally intensive tasks, which could hardly be executed as a whole by the local computing system of the vehicle. Consequently, the need for offloading part of the workload on a remote HPC infrastructure has been proposed [177]. This challenging approach demands strict requirements on the maximum latency that must be considered in a worst-case sense rather than in average-case sense.

Example – Environmental monitoring. Another large class of use cases is the one including applications for monitoring. For example, in the environmental context, we can mention the rainfall forecasting models or, more in general, the disaster prediction applications [201], where we need to process a large amount of data and make a prediction, by a specific deadline. This is necessary in order to generate an alert signal in a useful time frame and, thus, trigger suitable emergency plans. Similarly, large infrastructures also need efficient health monitoring systems. For example, a bridge monitoring system [254] requires the collection and the real-time processing of data from sensors on a large scale. When a dangerous situation is detected, an alert should

be raised as soon as possible to stop the road traffic. The necessity of having a worst-case requirement is justified by the fact that a late alert may compromise the bridge safety and, then, lead to loss of human life.

Example – European projects. Recent EU H2020 projects, like MANGO [94] and RECIPE [95] are addressing the challenges introduced by use cases with similar requirements. In the former, the three use cases – a real-time medical imaging application, a video transcoder, and a network digital signal processing algorithms – need a large amount of computational power to process a highly variable amount of input data, with guarantees in terms of maximum response time or latency. For example, the medical application receives data from a large set of tomographic sensors, performs the proper transformations to create a human body rendering, and recomposes them to create a streaming video. Since the video is played and observed by the medical personnel in real-time, its reproduction has low-latency requirements. The latter project, RECIPE, includes three use cases: a weather forecast and environmental monitoring system, the real-time analysis of health bio-signals in a big-data context, and a geophysical imaging tool. For the first use case, the application needs to collect data from a weather forecast model and a wide range of sensors, located by rivers and basins, to monitor the water levels. Then, suitable algorithms are required for analyzing data and detecting risky conditions in a short time frame, so that a proper emergency procedure could be implemented. Similarly to the previous bridge monitoring, in this case, worst-case timing requirements are necessary to ensure people's safety.

Summary. Many of the previously described applications can be considered mission- or safety-critical. These categories expose the systems to authority regulations and, in particular, to certification processes. Even if we are not aware of any certification requirement for HPC applications related to timing requirements at the time of writing, we expect that in the future, it may become a legal requirement.

Overall, all of these use cases can benefit from the scalability capabilities of HPC platforms, but, at the same time, they need solutions that could also provide timing constraints guarantees. Current HPC resource management solutions are focused on maximizing the throughput or maintaining the best average Quality-of-Service (QoS), whatsoever it is defined. However, the requirements of timing-sensitive applications are usually expressed in the form of maximum response time. Guaranteeing a maximum response-time is in trade-off with guaranteeing the maximum throughput. For instance,

the scheduler of a general-purpose operating system tends to control the number of process context-switches to reduce the overhead and maximizing the system throughput. Unfortunately, the response time is negatively affected in this case.

To guarantee the real-time requirements, we need a combination of design-time effort and run-time resource management strategies. The former's goal is to characterize the applications accurately and, in particular, their timing profiles. On the other hand, the latter needs to develop the knowledge of the target platform at run-time – and this is crucial for HPC platforms that are too complex to be analyzed and characterized at design-time – in order to tune the resource management policies [205]. In this regard, it is worth noting that the number of papers dealing with response-time driven resource management in HPC is only 15% of the total amount reported in the survey by Singh et al. [241]. However, in such papers, the response-time metrics are considered in an average sense only, and the violation of the deadlines is considered a degradation of the *Service Level Agreement (SLA)*, similar to soft real-time systems of Figure 1.4c. The use of the utility function, and in particular the analysis of the case of Figure 1.4a, is uncommon in HPC researches. Even if the *hard* real-time on parallel architectures has been widely studied in recent decades [67], the applicability of the proposed techniques to HPC applications is limited, due to both the complexity of HPC infrastructure and the necessary use of COTS components to reduce the cost of the clusters. A recent work [75] proposed a real-time scheduler for a COTS x64 architecture, capable of dealing with the *System Management Interrupts (SMIs)*. SMIs are, in fact, one of the most critical sources of timing unpredictability of COTS platforms [146]. However, SMIs are just one of the numerous hardware and software mechanisms that make an HPC cluster unpredictable in time.

1.4.3 The WCET problem

To satisfy the temporal requirements of the real-time systems (especially the *hard* real-time systems), we need to know the maximum value for the execution time. The upper bound value on the execution time of a given task is called *Worst-Case Execution Time (WCET)*. The WCET is necessary to perform the scheduling analysis and verify the timing constraints. The techniques to estimate the WCET will be later presented in Section 2.3. However, in general, computing this value in modern architecture and complex programs is a non-trivial problem. The trend we described in Section 1.1 makes the obtaining of a good timing model of the processor (or, more widely, of the system)

difficult, that, in turn, makes the computation of the WCET hard. The use of COTS components in real-time applications is challenging and adds another layer of complexity in WCET estimation, due to the numerous sources of unpredictability affecting these platforms [65]. In fact, COTS platforms are built with average performance in mind, and not intended to provide a timing model able to compute the WCET easily. As an extreme example, certain DRAM controllers of COTS platforms do not even guarantee an upper-bound for the timing latency in accessing a memory location [258], making the computation of a finite WCET impossible. Finally, it is easy to imagine how difficult it could be to estimate a WCET for an HPC application, running on complex machines and possibly multi-node, with network and other external interferences.

1.5 This Thesis

The common motivations behind the works presented in this thesis arise from the WCET problem described above and the necessity of finding alternative approaches to the traditional static analyses and real-time systems, in both embedded and HPC scenarios.

1.5.1 Contributions

This thesis is the result of many scientific works dealing with non-traditional approaches to real-time systems and, in particular, probabilistic real-time. We can identify the following macro-areas of contributions:

1. Advancing the theoretical framework of probabilistic real-time, especially on how to estimate the uncertainty behavior for safety-critical systems.

2. Analyzing the applicability of probabilistic approaches to real systems, from small embedded micro-controller to large HPC infrastructure, identifying the open challenges and the timing requirements of each scenario.

3. Proposing the use of probabilistic techniques in non-traditional real-time systems, such as mixed-criticality and heterogeneous computing.

4. Exploiting the timing analysis techniques for the estimation of different metrics than the execution time, especially in the context of energy-constrained systems.

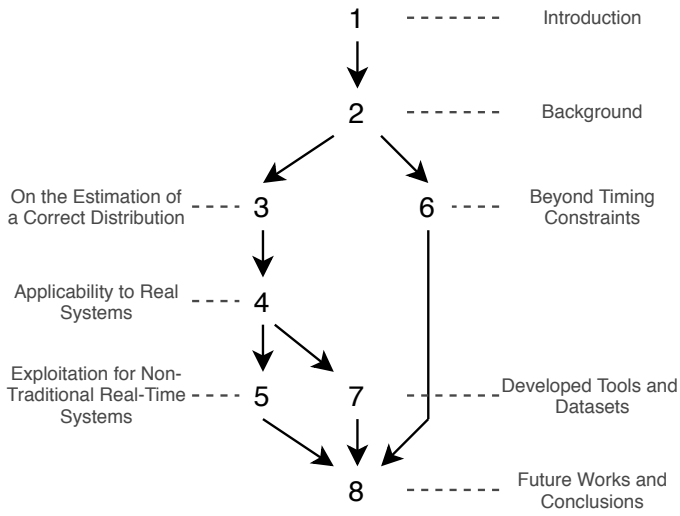


Figure 1.5: *Dependency graph of the chapters of this thesis.*

5. Developing tools and datasets to speed up scientific research and the adoption of these methods in industrial environments.

1.5.2 Structure

Due to the nature of the theoretical and practical works performed, this thesis does not follow the usual structure of *introduction*, *methods*, *results*, and *conclusions*. Rather, this thesis is organized as follows: after the introduction of this chapter, the necessary background is provided in Chapter 2. This background is essential, especially for the statistical aspects, which may be far from the usual knowledge of a computer scientist or engineer. In the same chapter, Section 2.5 presents an overview of the State of the Art, i.e., the previous works related to this thesis. Each of the subsequent chapters includes more specific literature reviews for their own subtopics. After Chapter 2, novel contributions are presented, including the purely theoretical advances (Chapter 3), studies on the applicability of probabilistic techniques to real systems (Chapter 4), exploitations for non-traditional real-time systems (Chapter 5), and for non-timing constraints (Chapter 6). Each chapter and its sections explain the proposed methodology and the results, being, in this way, self-contained works, but connected with the main story-

line of the thesis. Finally, Chapter 7 presents the developed tools and datasets, and the conclusions and possible future works are summarized in Chapter 8. The logical dependencies among the chapters are depicted in Figure 1.5 and among the sections are depicted in Figure 1.6. Appendix A contains the list of scientific articles written during the Ph.D. course of the author, which are the foundations on which this thesis was written. Appendix B contains the formulas of some statistical testing procedures referenced in the thesis.

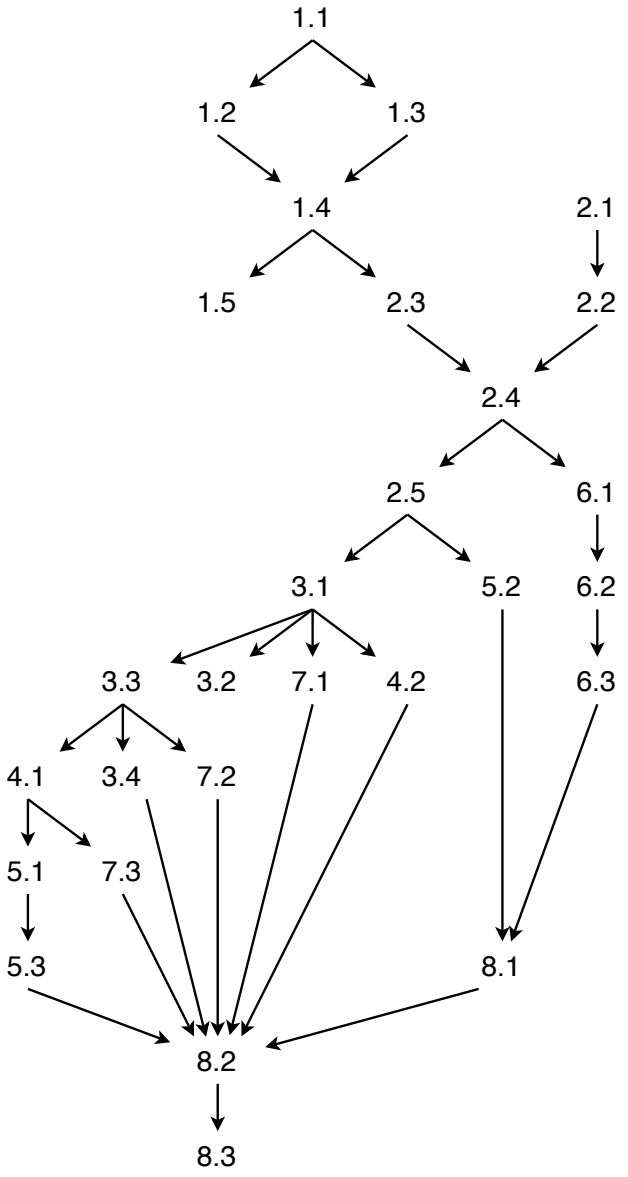


Figure 1.6: *Dependency graph of the sections of this thesis.*

CHAPTER 2

Background

Before entering in the innovative part of this thesis, this chapter provides the necessary background to the reader to understand the subsequent ones. The focus is related to the statistical domain and, in particular, to the Extreme Value Theory and its application to real-time computing. The last section presents an overview of the state-of-the-art works related to this thesis.

2.1 Statistical Foundations

In the probabilistic theory, a *random experiment* is a procedure that can be repeated and has well-defined possible outcomes (at least two). The experiment is modeled by a *probability space* $(\Omega, \mathcal{F}, P)$, where Ω is the set of all possible outcomes, $\mathcal{F}$ the set of events, and P a probability measure function. A *random variable* is an entity repre-

Function	Acronym	Symbol	Interpretation	Limits
probabilistic mass func.	pmf	$p_X(x_i)$	$P(X = x_i)$	$[0; 1]$
cumulative distribution func.	cdf	$F_X(x)$	$P(X < x)$	$[0; 1]$
probabilistic distribution func.	pdf	$f_X(x)$	$\frac{d}{dx}F_X(x)$	$[0; \infty]$
complementary cdf	ccdf	$\bar{F}_X(x)$	$P(X \geq x)$	$[0; 1]$
inverse cdf	icdf	$F_X^{-1}(p)$	$x : F_X(x) = p$	$D(X)$

Table 2.1: *Functional characterizations of the probabilistic measure function P. The symbol $D(X)$ represents the domain of the random variable X.*

senting the outcome of the experiment, formally a function having the domain Ω and as co-domain a measurable set. The *realization* of a random variable is its value randomly selected according to the probability measure function P. Finally, the set of events, $\mathcal{F}$, is a subset of Ω representing a set of outcomes. When a random experiment is repeated several times, the outcome is a sequence of random variables $\{X_1, X_2, ..., X_n\}$. If the number of elements of this sequence is not limited, the set $\mathcal{X} = \{X_1, X_2, ...\}$ is called *random process*.

Example 2.1.1 (Rolling a dice)**.** The experiment "rolling a dice" has the probability space depending on the number faces of the dice – e.g., $\Omega = \{1, 2, 3, 4, 5, 6\}$ – the set $\mathcal{F}$ can be any possible subset – e.g., "the resulting face has a number lower than 4" – and the probability function P depends on the fact that the dice is weighted (cheat) or not; in the latter case, $P(X = 5) = \frac{1}{6}$. The random variable X corresponds to the "number we observe on the face of the dice", and its realization can be, for instance, "5". ■

2.1.1 Probability functions

The abstract concept of probability measure function is then developed in the functions shown in Table 2.1. The *probabilistic mass function (pmf)* is used in the case of discrete random variables (such as the dice outcome in Example 2.1.1), while the *probabilistic distribution function (pdf)* is used for continuous variables. Formally,

the *cumulative distribution function (cdf)* is computed as $F_X(x) = \sum_{x_i \leq x} p_X(x_i)$ for discrete variables and $F_X(x) = \int_{-\infty}^{x} f_X(t)dt$ for continuous variables. Then, its complement, the *complementary cumulative distribution function (ccdf)*, is defined as $\bar{F}_X(x) = 1 - F_X(x)$, as well as the *inverse cumulative distribution function (ccdf)* $F_X^{-1}(p) = \{x : F_X(x) = p\}$. All of these functions are said to *describe* a statistical distribution (or, equivalently, probability distribution), in this thesis represented by the calligraphic script font, for instance, $\mathscr{G}$.

Empirical determination of probability functions. Observing a phenomenon via experiments, usually requires to estimate the probabilistic distribution, because the characteristic functions are in many cases unknown. The simplest method to estimate a distribution is to compute the *empirical cumulative distribution function (ecdf)* by measuring the realizations of the random variables several times. Given n observation as independent and identically distributed (i.i.d.) random variables $\{X_1, X_2, ..., X_n\}$, the ecdf is computed as follows:

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{X_i \leq x} \tag{2.1}$$

where $\mathbf{1}_A$ is the *indicator function* defined as:

$$\mathbf{1}_A = \begin{cases} 1 \text{ if } A \text{ is true} \\ 0 \text{ otherwise} \end{cases}$$

By the strong law of large numbers and as stated by the Glivenko-Cantelli theorem [103, 43], the ecdf converges almost surely to the real cdf:

$$P\left(\lim_{n \to \infty} \left| \hat{F}_n(x) - F(x) \right| = 0\right) = 1$$

From the ecdf, the other distribution functions can be computed in their empirical form. An example of ecdf generated from a sample of 100 points on a normal distribution is depicted in Figure 2.1.

2.1.2 Common operations

Truncation. In some cases, it is needed to truncate the distribution at a given value, reducing its domain. This operation is done if some previous knowledge where the

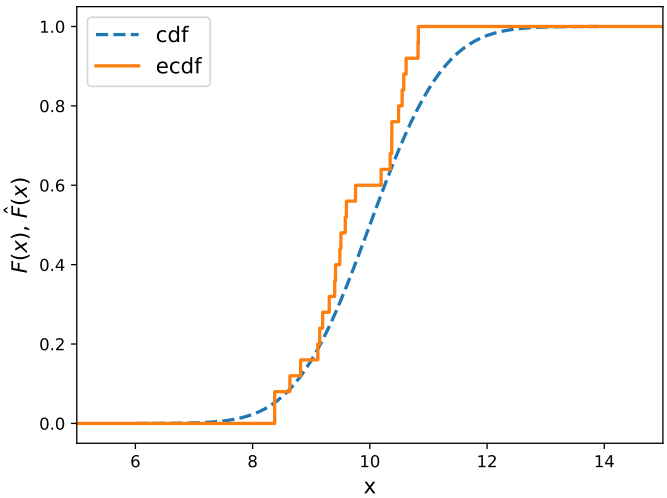


Figure 2.1: *An example of ecdf generated from 25 measurements sampled from the normal distribution $N(10, 1)$, in comparison with the original cdf.*

values lie is available, and it produces a conditional distribution. Let $f(x)$ be the pdf of the distribution to be truncated, and a, b respectively the minimum and the maximum values of the truncating window, then the pdf becomes:

$$\mathcal{T}_a^b[f(x)] = P(x|a < X \leq b) = \frac{f(x)}{F(a) - F(b)}$$

The new function ($\mathcal{T}_a^b[f(x)]$) satisfies all the properties of a pdf. The truncation operators can be equivalently applied to discrete distributions and, consequently, pmfs.

Convolution. The sum of random variables is performed thanks to an operator called *convolution*. This operator is identified by the symbol $\circledast$, and it is applicable to i.i.d. random variables[1]. In particular, we define the convolution of two random variables $X_{sum} = X_1 \circledast X_2$ as the convolution of their pmfs or pdfs $f_{X_{sum}}(x) = f_{X_1}(x) \circledast$

[1] Methods exist to convolute non-i.i.d. random variables [96], but they are not relevant for this thesis.

$f_{X_2}(x)$. The general formula of $X = Y \circledast Z$ for discrete random variables is:

$$p_X(x) = p_Y(x) \circledast p_Z(x) = \sum_{n=-\infty}^{+\infty} p_Y(n)p_Z(x-n)$$

and its counterpart for continuous random variables is:

$$f_X(x) = f_Y(x) \circledast f_Z(x) = \int_{-\infty}^{+\infty} f_Y(t)f_Z(x-t)dt$$

The convolution operator satisfies many algebraic properties, such as commutativity, associativity, and distributivity, but lacks an identity item.

Example 2.1.2 (Truncation and Convolution)**.** Let be Y and Z two random variables with the following pmfs:

$$p_Y(x) = \begin{cases} 0.5 & \text{if } x = 10 \\ 0.5 & \text{if } x = 20 \end{cases} \qquad p_Z(x) = \begin{cases} 0.3 & \text{if } x = 0 \\ 0.3 & \text{if } x = 10 \\ 0.4 & \text{if } x = 15 \end{cases}$$

then, the convolution of $X = Y \circledast Z$ is:

$$p_X(x) = \begin{cases} 0.15 & \text{if } x = 10 \\ 0.30 & \text{if } x = 20 \\ 0.20 & \text{if } x = 25 \\ 0.15 & \text{if } x = 30 \\ 0.20 & \text{if } x = 35 \end{cases}$$

Let us assume we can apply the truncation operator because we know that $X \leq 25$, then the final pdf becomes:

$$\mathcal{T}_{-\infty}^{25}[p_X(x)] = \begin{cases} 0.2308 & \text{if } x = 10 \\ 0.4615 & \text{if } x = 20 \\ 0.3077 & \text{if } x = 25 \end{cases}$$

■

2.2 The Extreme Value Theory

At the beginning of the 20th century, the needs of some astronomers to correctly reject outliers in the observations were the reasons behind the origin of the *Extreme Value Theory (EVT)* [153]. This branch of statistics deals with the probability of extreme (i.e., rare) events to occur. Consequently, it is opposed to the well-known *Central Limit Theorem (CLT)* that, instead, studies the probability distribution around their mean value. The EVT is frequently used for risk management in manifold contexts, such as natural disaster prediction, financial applications, and earth sciences, but also in several fields of engineering [45].

2.2.1 Statistical distribution of the extremes

Given a sequence of random variables $X_1, X_2, ..., X_n$ representing the observation of a phenomenon, the EVT provides the limit distribution at the extremes, i.e., the $\max(X_1, X_2, ..., X_n)$ or $\min(X_1, X_2, ..., X_n)$. In the context of this thesis, the sequence of $X_1, X_2, ..., X_n$ is a sequence of observation of execution times. As detailed later in Section 2.4.1, we are interested in the worst-case value only; thus we can limit our dissertation to the $\max$ value only so that it simplifies the following notation. The $\min$ is symmetrical, but it has no significance for the purposes of real-time computing.

We can formalize the probability of not observing a value for the considered phenomenon larger than a given value x as follows:

$$\begin{aligned} & P(\max(X_1, X_2, ..., X_n) \leq x) \\ & = P(X_1 \leq x,\ X_2 \leq x,\ ...,\ X_n \leq x) \\ & \overset{\text{iid}}{=} P(X_1 \leq x) \cdot P(X_2 \leq x) \cdots P(X_n \leq x) \\ & = [F(x)]^n \end{aligned}$$

where $F(x)$ is the (unknown) cdf of X_i. The symbol $\overset{\text{iid}}{=}$ is a step that requires the sequence of random variables to be independent and identically distributed (i.i.d.), which permits the expansion by multiplication of the single cdf. From the results of the EVT[2], it is possible to demonstrate that [46]:

$$\exists a_n, b_n \text{ s.t. } \lim_{n \to \infty} [F(a_n x + b_n)]^n = G(x) \tag{2.2}$$

[2]The statistical details and proofs are omitted because out of the scope of this document.

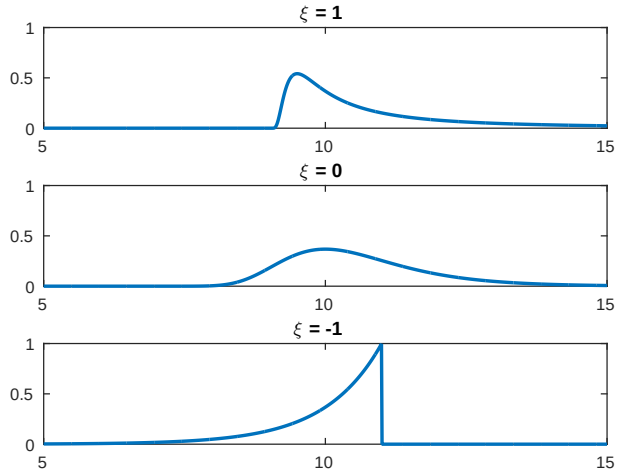


Figure 2.2: *The pdfs of the distribution GEVD*$(10, 1, \xi)$ *with* $\xi = 1$ *in the top subfigure,* $\xi = 0$ *in the top subfigure, and* $\xi = -1$ *in the bottom subfigure.*

where $G(x)$ is the cdf of a so-called *Extreme Value Distribution*. According to the Fisher-Tippett-Gnedenko theorem [93, 104] this distribution can assume only three forms: the Gumbel, the Weibull, or the Fréchet distribution, independently from the original distribution represented by the cdf $F(x)$. This theorem is the main result of the EVT and shows how the original distribution of the random variables does not affect the distribution of extremes.

Generalized EVT. In 1978, McFadden [182] discovered that the three distributions are actually particular cases of a more general distribution: the *Generalized Extreme Value Distribution (GEVD)*. This distribution is characterized by the following cdf:

$$G(x) = \begin{cases} e^{-e^{\frac{x-\mu}{\sigma}}} & \xi = 0 \\ e^{-[1+\xi(\frac{x-\mu}{\sigma})]^{-1/\xi}} & \xi \neq 0 \end{cases} \tag{2.3}$$

The GEVD distribution $\mathscr{G}(\mu, \sigma, \xi)$ is parameterized by the *location* parameter μ, the *scale* parameter σ, and the *shape* parameter ξ. The location parameter is conceptually similar to the mean value of a distribution, while the scale parameter to the standard deviation[3]. The shape parameter ξ has a very important role (especially in probabilistic

[3]They are *similar* for a conceptual standpoint but they have not the same formulation, the GEVD has a

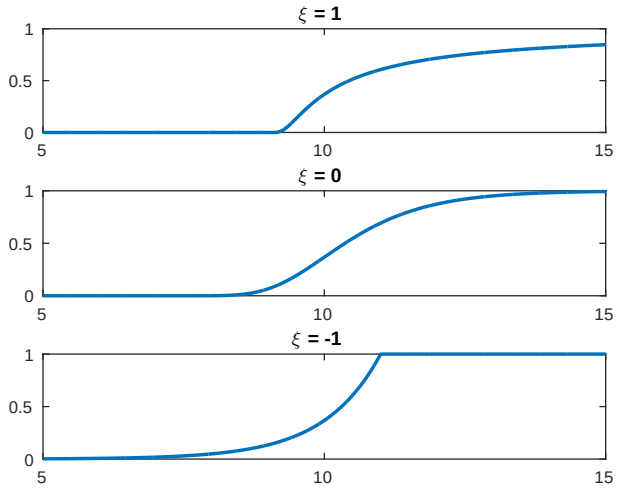


Figure 2.3: *The cdfs of the distribution GEVD*$(10, 1, \xi)$ *with* $\xi = 1$ *in the top subfigure,* $\xi = 0$ *in the top subfigure, and* $\xi = -1$ *in the bottom subfigure.*

real-time computing), since it represents which sub-class of distribution a given GEVD is part of:

- $\xi > 0$: the GEVD is a Fréchet distribution.
- $\xi = 0$: the GEVD is a Gumbel distribution.
- $\xi < 0$: the GEVD is a Weibull distribution.

The pdfs and the cdfs of these three cases are respectively depicted in Figure 2.2 and Figure 2.3. The key difference among the GEVD subclasses is "how fast" the cdf converges to 1. In fact, if $\xi > 0$ the Fréchet distribution converges to 1 slowly compared to the $\xi = 0$ Gumbel case. Instead, in the $\xi < 0$ case, the Weibull subclass, the cdf converges to 1 very quickly, and it has a finite value of x such that $G_{\xi<0}(x) = 1$, conversely to the previous subclass were instead it is infinite: $G_{\xi\geq 0}(+\infty) = 1$. The value of ξ has crucial consequences for probabilistic real-time, as subsequently explained in Section 2.4.1.

mean value of $\mathrm{E}[\mathscr{G}] \neq \mu$ and $\mathrm{VAR}[\mathscr{G}] \neq \sigma^2$. The exact formula is not reported because it is rather complex and irrelevant in this discussion.

Generalized Pareto Distribution. The other distribution which can be used to model extreme events is the *Generalized Pareto Distribution (GPD)* as named by Pickands et al. in 1975 [202]. This distribution, in its most generic form, is characterized by the following cdf:

$$G(x) = \begin{cases} 1 - e^{-\frac{x-\mu}{\sigma}} & \xi = 0 \\ 1 - (1 + \xi\frac{x-\mu}{\sigma})^{-1/\xi} & \xi \neq 0 \end{cases} \tag{2.4}$$

with the following support[4] for x:

- $x - \mu \geq 0$ if $\xi \geq 0$
- $-\frac{1}{\xi} \geq \frac{x-\mu}{\sigma} \geq 0$ if if $\xi < 0$

Similarly to the GEVD, a GPD distribution $\mathscr{G}(\mu, \sigma, \xi)$ is parameterized by the *location* parameter μ, the *scale* parameter σ, and the *shape* parameter ξ. However, in many cases the GPD is restricted to 2 parameters only, by setting $\mathscr{G}(0, \sigma, \xi)$, i.e. $\mu = 0$ [5]. This distinction does not affect the generality of the GPD for the EVT problem. In this thesis, when this differentiation is needed, we refer with GP3D and GP2D for, respectively, the 3-parameters GPD and the 2-parameters GPD.

For specific parameters values (especially the value of the shape parameter ξ), subclasses of GPD can be recognized:

- $\xi = 0$ and $\mu = 0$: the GPD is a Exponential distribution.
- $\xi > 0$ and $\mu = \frac{\sigma}{\xi}$: the GPD is a Pareto distribution.

The GPD and GEVD distributions are asymptotically equivalent and, consequently, both of them can be used to model the extreme behaviors, based on the result of the Pickands-Balkema-de Haan theorem [17, 202]. However, the two distributions have to be estimated by using two different methods explained in the next paragraphs.

2.2.2 The estimation procedure

The estimation procedure comprises two parts: initial filtering is applied to the execution time observations, and then the actual estimation performed by an appropriate estimator algorithm.

[4] Subset of the function domain containing the values for x such that $F(x) \neq 0$.

[5] Rarely, it is used with only with the shape parameter, by setting $\mathscr{G}(0, 0, \xi)$. However, to the best of our knowledge, this form has been never used for probabilistic real-time.

Input filtering. A filtering technique is applied to the input values to focus the estimator on the distribution's tail. Two possible approaches are frequently used in both statistical and real-time fields:

- *Block-Maxima (BM).* The following filter is applied to the sequence of observations $X_1, X_2, ..., X_n$:

$$Y_i = max(X_{B\cdot(i-1)+1}, X_{B\cdot(i-1)+2}, ..., X_{B\cdot i}) \tag{2.5}$$

  where B is a parameter called *block size*. The set of observations is divided into blocks of fixed size B, and for each block, the maximum value is taken. The sequence $Y_1, Y_2, ..., Y_m$ represents the maxima of the blocks, with $m = \lceil \frac{n}{B} \rceil$.

- *Peak-over-Threshold (PoT).* This filter is a simple threshold-based cut:

$$Y = \{X_i > u\} \tag{2.6}$$

  where u is a predefined threshold. The sequence $Y_1, Y_2, ..., Y_m$ represents the measurements that are greater than the threshold u, discarding all the values under this threshold. In this case, contrarily to the BM case, the value of m is unknown a priori of the analysis because it depends on the values themselves.

The selection of BM or PoT impacts the final distribution: if BM is applied, the estimated distribution has to be a GEVD, while if PoT is applied, the resulting distribution has to be a GPD. To estimate such distributions, an appropriate estimator must be used, as described in the following paragraph.

Common estimators. The commonly used estimator for EVT analyses is the *Maximum Likelihood Estimator (MLE)*. This method is well-known because it is used as the estimator in many fields of science and engineering. Given a model, the GEVD or the GPD, MLE estimates its parameters by maximizing the likelihood that the process described by the model is the same as the one which produced the observation data – i.e., the execution time in our case. However, it was observed [222] that, in some cases, the MLE estimator fails to obtain a valid EVT distribution due to the presence of a local minimum of the MLE optimization function. One possible solution to this problem is to initially use the *Probabilistic Weighted Moment (PWM)* estimator and, then, use MLE to refine the solution. In fact, PWM usually leads to a less precise solution compared to MLE, but MLE may not converge or may converge to a local minimum. The reader

is invited to check the paper of Diebolt et al. [73], for a detailed analysis of the two estimators in the EVT context.

2.2.3 The EVT hypotheses

As any mathematical theorem, the EVT theorems also require some hypotheses to be satisfied, in a way that the EVT process produces a correct and non-underestimated distribution. The following three paragraphs briefly describe such hypotheses.

Independent and Identically Distributed (i.i.d.). The input sequence of observations – i.e., the execution time trace in probabilistic real-time scenario – must be *independent and identically distributed (i.i.d.)*. In real-time computing, this hypothesis is mainly dependent on the processor and the system architecture. For example, a multi-core processor including a cache memory would not probably be able to fulfill the independence requirement, due to the time locality principle. Time samples from consecutive executions of the same task, in fact, are affected by the data locality given by the cache, making the execution times not independent. In practice, the i.i.d. requirement can be relaxed in favor of the stationary property and weaker independence properties [155, 230]. If these hypotheses hold, the EVT estimation process can be correctly applied. They can be formalized as follows [228]:

- *Stationarity*: Given a random sequence $X_1, X_2, ..., X_n$ of size n, the process is said to be *strict stationary* iff for any choice of k, l, m with $0 < k+l+m < n$ the following condition is true: $F(X_k, X_{k+1}, ..., X_{k+l}) = F(X_{k+m}, X_{k+m+1}, ..., X_{k+m+l})$, where F is the cdf of the joint distribution. This condition implies the identical distribution of the random variables. In real-time computing, the stationary hypothesis indicates a flat distribution of execution times, with constant variance. For instance, a task that drastically changes the job execution time after some runs violates this property.

- *Short-range independence*. Given a sequence of random variables $X_1, X_2, ..., X_n$ of size n, the sequence is said to be short-range independent if for any $i_1 < i_2 < \cdots < i_p < j_1 < \cdots < j_p \leq n$ s.t. $j_1 - i_p \geq s > 1$, defining F_{IJ} the cdf of $X_{i_1,...,i_p,j_1,...,j_n}$, F_I the cdf of $X_{i_1,...,i_p}$, F_J the cdf of $X_{j_1,...,j_p}$ we have $|F_{IJ} - F_I F_J| \leq \alpha_{n,s}$ where $\alpha_{n,s}$ is a sequence with non-decreasing values with respect to s and $\alpha_{n,s} \to 0$ for $n \to \infty$. The intuition behind this property

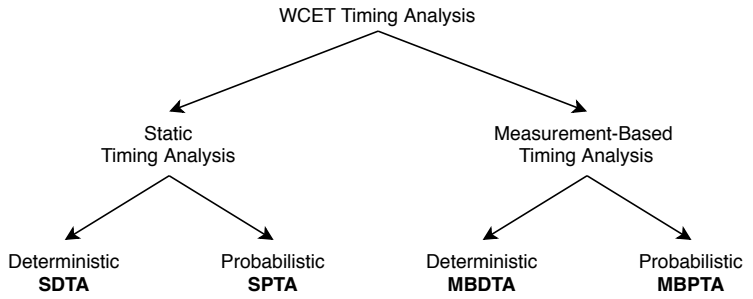
can be noticed looking at the content of the absolute value operator, which is zero if F_I and F_J are perfectly independent; otherwise, the dependency has to be upper-bounded by (a function of) the distance among the random sequences. In real-time computing, an example of a cause of short-independence property violation is the presence of processor cache effects between two job instances.

- *Long-range independence.* According to this property, the time series does not show a significant correlation across large time-spans. We define this property by defining its opposite. A long-range dependent sequence can be defined as: a random sequence $X_1, X_2, ..., X_n$ of size n is said to have long-range dependence if its auto-correlation function $\rho(\tau)$ decays exponentially: $\rho(\tau) \sim \frac{L(\tau)}{\tau^{1-2d}}$ with $0 < d < \frac{1}{2}$ where $L(\tau)$ verifies $\lim_{t\to\infty} \frac{L(at)}{L(t)} = 1$ for some $a > 0$.

It is worth noting that the short-range hypothesis is a sufficient but not necessary condition for EVT applicability [156]. If the dataset presents a short-range dependence, it can still be considered valid if other conditions hold. A statistical paper [156] proposes some diagnostic methods to check these properties, however, they require a non-trivial in-depth analysis of the dataset and the used estimation method. For this reason, we suggest using such techniques only when the short-range independence hypothesis is false, and it is not possible to improve the system to be adherent to this hypothesis.

Maximum Domain of Attraction (MDA). The hypothesis of *Maximum Domain of Attraction (MDA)* (called *matching* by some authors [228]) requires that the original distribution of extremes actually converges to one of the EVT distribution classes. When the observation samples are represented with random variables having continuous distribution functions, the MDA hypothesis is true in the overwhelming majority of the times [223]. This fact is, instead, not necessarily true when discrete distributions are considered. It is hard to provide a generalization of the MDA hypothesis for probabilistic real-time, being a statistical detail that cannot be easily linked to specific hardware or software characteristics. Section 3.4.4 contains some empirical results to give a "practical" meaning of this hypothesis.

Representativity. The representativity hypothesis is not commonly discussed in traditional applications of EVT, but it plays a critical role in probabilistic real-time: Observing natural phenomena is very different from observing an execution time that depends on human-made conditions. The representativity hypothesis requires the obser-



Input:	HW and SW Description	HW and SW Description	Execution Time Measurements	Execution Time Measurements
Output:	WCET	pWCET	WCET	pWCET

Figure 2.4: *The classification of timing analysis techniques.*

vations to be a good representation of the real phenomenon, in our case of the execution time, by reducing as much as possible the epistemic uncertainty. The details and explanation of this hypothesis is described later in the dedicated Section 4.2.

2.3 Worst-Case Execution Time Estimation

2.3.1 Taxonomy

Several techniques of the WCET estimation have been developed since the beginning of the use of embedded systems in critical systems. Next paragraphs describe the taxonomy of WCET analyses based on the classification by Abella et al. [1]. The taxonomy is also depicted in Figure 2.4.

Static analyses. Most of the traditional analyses can be classified under the category of *Static Deterministic Timing Analyses (SDTA)*. The SDTA class includes all the analyses that use the design-time parameters of the system (*static*), such as the hardware architecture knowledge, to compute via algorithmic operations a WCET value (*deterministic*). By cross-analyzing the program information (commonly, the control-flow graph) and the architectural characteristics (e.g., the timing of every single instruction),

the WCET can be computed and, usually, proved to be correct or to be an overestimation of the real one.
Instead, the *Static Probabilistic Timing Analyses (SPTA)* category includes the methods that use probabilistic characterizations of the hardware and/or the software to compute the final distribution. This method, which the seminal paper can be considered the Bernat et al. [27] work in 2002, does not infer any probability distribution, but it assumes to have adequate knowledge of the hardware and software phenomena. For example, the control-flow graph can be annotated with statistical distribution instead of execution time, and the final distribution can be computed via the convolution operator described in Section 2.1.2.

Measurement-based approaches. Totally different approaches to compute the WCET are the measurement-based techniques. The WCET is not computed in a classical algorithmic way, but it is derived from the measurements directly performed on the system under analysis. Similar to static analyses, they can be categorized in *Measurement-Based Deterministic Timing Analyses (MBDTA)* and *Measurement-Based Probabilistic Timing Analyses (MBPTA)*. The MBDTA is the most trivial technique we can think to get the WCET: observing the execution time numerous times and take the maximum observed value, in some work called *Worst-Case Observed Time (WCOT)*. Provided that the worst-case condition is observed, the WCOT is equivalent to the WCET. However, ensuring to have observed the worst-case condition is usually very difficult.
For this reason, the probabilistic version, MBPTA, has been developed to obtain a probabilistic bound by applying statistical techniques to the observed values of execution time. MBPTA is the main subject of this thesis, and most of the works are based on the theory previously described in Section 2.2, i.e., the EVT. Section 2.5 presents the State of the Art for MBPTA.

2.3.2 Current standards

Static analyses are still the WCET estimation methods currently in use for safety-critical systems and, in particular, for systems that require certification. Among the various standards, the following documents are worth to be cited [145, 172]:

- *DO-178* [226]: This is the current standard for avionic software used by almost all the certification authorities for aircraft (formally *Software Considerations in Airborne Systems and Equipment Certification*). According to this standard, the

software must be *verified*, and *testing* is not sufficient to achieve the verification goal. Even if not directly stated, this requirement entails that the WCET analysis must be performed with static analyses or with a measurement-based method which can be formally proved to be safe (e.g., providing the exhaustiveness of an MBDTA exploration).

- *IEC-61508* [134]: The *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems* is an international standard published by the *International Electrotechnical Commission*. This standard recommends the use of static tools for any safety-critical software, and it "highly" recommends them for Safety Integrity Levels (SIL)[6] 2, 3, and 4. This standard is the basis for many others, including the automotive ISO-26262 (described in the next bullet item) and the railway standard EN 50126-9.

- *ISO-26262* [135]: The *Road Vehicles - Functional Safety* standard, developed from the IEC-61508, does not explicitly state that the WCET must be calculated statically, but it requires the scheduling analysis to satisfy certain properties. This can be achieved only if the WCET is safely computed, i.e., with static methods.

- *Other standards*: Like the previous, other standards like the railway standard EN 50126-9 or the medical devices standard EN-60601 require the verification or similar concept for the WCET, leading to the requirement of static analyses.

All the standards listed above require the validation of the used tool. For the most critical software, the tool must be formally proven to correctly estimate the WCET or output an upper-bound of it. Measurement-based methods are not excluded a priori, but their safety must be established before using them in real products. One of the aims of this thesis is to identify the challenges of probabilistic and measurement-based methods that make them non-compliant with certification requirements.

2.3.3 The limits of the traditional analyses

We already cited the limits of traditional static analyses, which require an infeasible amount of computational power, or, due to the introduction of a considerable amount

[6]The SIL represents the criticality of a given safety critical feature. SIL1 is the least critical class, while SIL4 is the most critical one. For example, SIL1 includes several automotive safety features, while SIL4 includes some embedded systems in nuclear environments.

of approximations, they produce a very pessimistic WCET. By focusing on just the pessimism caused by the approximations needed in handling the caches, according to the survey of Wilhelm et al. [257], it can vary in the range $15\% - 50\%$. This pessimism refers to experiments conducted in the early 2000s when the processors were not as complicated as today.

To show a modern example, we performed the following experiment: we computed the WCET of a `bsort100` benchmark from the Mälardalen WCET benchmarks suite [118] by using the state-of-the-art static tool Heptane [123] for the architecture ARMv7. The tool has been configured for a Cortex-A8 processor, and we obtained an estimated WCET$= 5\,534\,964$. Then, we run for $10\,000$ times the `bsort100` benchmark in the same conditions on a real Cortex-A8 board, and we obtained an average execution time of $315\,684$ and a worst-case observed time of $466\,655$. Clearly, the latter value is the WCOT and not necessarily the real WCET, but due to the trivial benchmark, it is reasonable to think that it is probably near the real one. Consequently, the static WCET tool estimated a 10x larger value for the WCET. By performing the probabilistic analysis following described, the WCET is estimated to be $882\,597$ (with violation probability of $p = 10^{-5}$). We want to remark that this experiment was not to provide an exhaustive analysis, but a simple example to give the reader a rough idea of the possible order of magnitude of the WCET over-estimation.

2.4 Real-Time Systems

In (hard) real-time systems, it is usual to express the computational workload with the concepts of *task* and *job*. A task is an entity that periodically or aperiodically spawns jobs. The job is the single unit of computation that performs the function the task is developed to. One of the fundamental model class of the real-time systems is the *task model*. The minimal form for a generic task τ_i is $\tau_i = (T_i, C_i, D_i)$, where T_i is the period (in case of periodic task) or inter-arrival period (in case of aperiodic task), C_i is the WCET, and D_i is the relative deadline. The relative deadline D_i refers to the deadline with respect to the activation time: a periodic task is activated at regular interval of size T_i, and each job must complete D_i time unit. A system is called *constrained-deadline* if $\forall \tau_i : D_i \leq T_i$, *implicit-deadline* if $\forall \tau_i : D_i = T_i$, and *arbitrary deadline* if there is no constraint on the deadlines. Many other possible parameters can be set in the model, for example a very common one is ϕ_i, that is the phase, i.e., the time offset

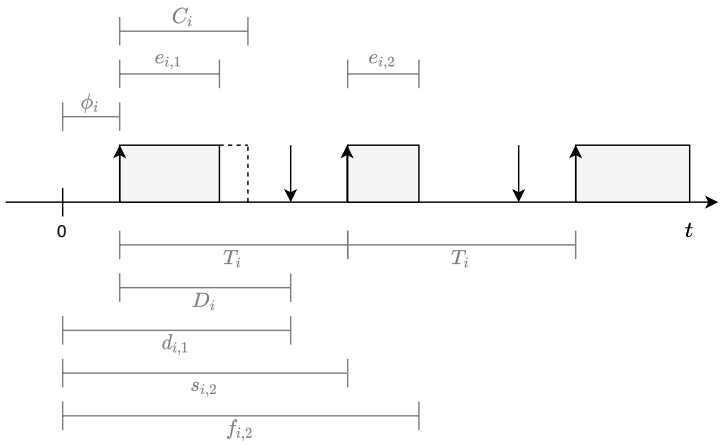


Figure 2.5: *An example of the model of a single task τ_i with some parameters specified. The arrows identify the activation time and the deadline.*

between the origin and the first job. Lowercase letters represent values related to the job (which are usually computed and not directly expressed in the model) and that refer to absolute time. Common examples are $a_{i,j}$, the activation or release time of the j-th job of the i-th task, $s_{i,j}$, the starting time when the job starts to execute, the $f_{i,j}$, the actual finishing time, and $d_{i,j}$, the absolute deadline. The system is correct if, for any hard real-time job, the following relation holds under any condition: $f_{i,j} \leq d_{i,j}$. For periodic jobs, this condition can be rewritten as: $s_{i,j} + e_{i,j} \leq a_{i,j} + D_i$, where $e_{i,j}$ is the actual execution time. Since the actual execution time is rarely know at design time, $e_{i,j}$ is replaced with the WCET C_i. All these variables are depicted for a better comprehension in Figure 2.5. The scheduling decisions of a real-time system composed of periodic tasks repeat over time, and the size of this time interval is called *hyperperiod*. The hyperperiod H can be computed as the Least Common Multiple of the tasks' period: $H = LCM(T_1, T_2, ..., T_n)$.

2.4.1 Probabilistic real-time computing

To overcome the difficulties in estimating a tight WCET in modern processors, *probabilistic real-time computing* has been developed. This thesis focuses on MBPTA approaches only, because SPTA did not attract significant attention from the researchers.

The difference with respect to the traditional model is to estimate a probabilistic distribution instead of a scalar value C_i. The execution times of the jobs of a given task τ_i, i.e. $e_{i,1}, e_{i,2}, ...$, compose a so-called *random process*, composed, in turns, of random variables. For simplicity, we use the traditional upper case syntax of random variables, writing this process as $X_1, X_2, ...$ We consider this random process as unlimited, i.e., we do not put any restriction on the number of times a job could be spawned. In other words, we do not set an upper limit of time for our system to run. These random variables are distributed according to a, usually unknown, probability distribution. We refer to this distribution with the name *Probabilistic Execution Time (pET)*.

Probabilistic-WCET. MBPTA approaches are based on the application of a statistical procedure to a finite sequence of random variables $X_1, X_2, ..., X_n$, sampled from the execution time random process. The output of such procedure is a statistical distribution called *probabilistic-WCET (pWCET)*, and it is usually expressed with its ccdf:

$$p = P(X \geq \bar{C}) = 1 - F_X(\bar{C})$$

The probability p, called *violation probability*, represents the probability of observing an execution time larger than $\bar{C}$. The random variable X represents a generic random variable of the process, by assuming that the random variables are identically distributed. The experimenter can select either C_i or p and accordingly compute the other value: it is possible to estimate the violation probability p given a WCET C_i or, vice versa, estimate the WCET C_i given a target violation probability p. The latter option is computed by using the icdf (see Table 2.1).

Provided that the pWCET distribution is correctly computed and it represents the real distribution of execution times, it is reasonable to expect that it is compliant with the certification of safety-critical systems: a probability of violation, corresponding to the real one, would be just another term in the Fault Tree Analysis of the safety-critical system. However, guaranteeing this safety is non-trivial and represents the major obstacle to probabilistic real-time use in the current industrial system. We will discuss in detail this problematic in Section 4.2.

The difference between the pET and the pWCET is crucial: the pET represents the execution time distribution, while the pWCET represents the distribution of the maxima values only. While it is theoretically possible to use the pET distribution for WCET purposes, it is not practically possible to obtain reasonable confidence, as explained in

the next paragraph.

The need of the EVT. In Section 2.1.1, we described the ecdf method to estimate a distribution directly from the time samples. The ecdf can be used to estimate the pWCET from the time measurement. However, due to the finiteness of the number of measurements that is possible to gather in a real-world scenario, the ecdf is, clearly, subject to estimation errors. This estimation errors can be upper-bounded by the Dvoretzky-Kiefer-Wolfowitz inequality [78]:

$$P\left(\max|\hat{F}_n(x) - F(x)| > \varepsilon\right) \leq \Psi \tag{2.7}$$

where $F(x)$ is the real cdf, $\hat{F}_n(x)$ is the ecdf computed according to Equation (2.1), Ψ is called *confidence*, and $\Psi = 2e^{-2n\varepsilon^2}$. Unfortunately, to obtain large confidence for a small estimation error bound, a large number of samples are required. For example, to obtain $\varepsilon \leq 10^{-5}$ with a confidence of $\Psi = 1 - 10^{-20}$, by inverting Equation (2.7) the number of measurements required is $n \geq 4 \cdot 10^9$: a large number that is non-practical in several use cases. This problem justifies the need of EVT for such estimation because, even if it has more hypotheses to be satisfied compared to ecdf, it allows us to estimate a very low probability with high confidence and a low number of samples. In the following chapters of this thesis, many works present various methods to estimate and improve such confidence.

The impact of ξ parameter. Besides the mere statistical interest on the distribution type, the parameter ξ of the extreme value distributions described in Section 2.2 has a significant effect on the pWCET:

- When $\xi < 0$ (i.e. a Weibull distribution in the GEVD case), the pWCET is a *truncated-tail* distribution, i.e. there exists a maximum iccdf $\bar{F}^{-1}(p)$ value for $p \to 0$. According to the distribution, such value represents the maximum admissible value, allowing us to select this value as the WCET.

- When $\xi = 0$ (i.e. a Gumbel distribution in the GEVD case), the pWCET is a *light-tail* distribution, i.e. the iccdf $\bar{F}^{-1}(p) \to \infty$ for $p \to 0$, however the $\bar{F}(x)$ goes "extremely fast" to zero, making the WCET, not theoretically, but practically bounded. In fact, a linear increase of the WCET corresponds to a decrement of the probability p faster than an exponential function.

- When $\xi > 0$ (i.e. a Fréchet distribution in the GEVD case), the pWCET is a *heavy-tail* distribution, i.e. the iccdf $\bar{F}^{-1}(p) \to \infty$ for $p \to 0$, but the ccdf $\bar{F}(x)$ goes to zero slower than the exponential distribution: arbitrarily large WCET has a non-negligible probability to be observed. Moreover, if $\xi > 1$, the mean of the distribution is even infinite: $E[X] = \infty, X \sim GEV(\mu, \sigma, \xi > 1)$, and this is clearly a problem for a tight WCET estimation.

Some effort was dedicated in previous works to identify the distribution class that best fits the worst-case execution time values. The results are controversial: some researchers [2, 60, 62, 121] claimed we should consider only the distribution with $\xi \leq 0$, because unbounded WCET has no sense. However, other researchers [167, 228] do not exclude the $\xi > 0$ case too. Often, the Gumbel distribution ($\xi \leq 0$) is the only one taken into account because it is an upper-bound of the Weibull. However, this property was improperly used in some works, hindering the pWCET reliability, as described by Section 3.2.

Most of the state-of-the-art works based their selection on empirical observations, which lack strong mathematical justifications, that are – indeed – difficult to achieve. For this reason, voluntarily ignore this discussion: this thesis does not consider any GEVD/GPD class restriction, remaining in this way as general as possible.

2.5 Literature Review - A General Overview

The research on time predictability of the embedded systems has been very active since the beginning of the use of embedded systems for safety-critical applications. It is still a very hot topic for research, from the WCET analyses to the scheduling of tasks. This section provides an overview of the literature having a role in this thesis and, in particular, on probabilistic approaches to estimate the WCET. Additional specific literature reviews are present in the following chapters when dealing with specific research topics; for instance, the state of the art of energy-constrained systems is proposed in Chapter 6 because related only to the works presented in that chapter.

2.5.1 Traditional WCET estimation

Determining an upper-bound to the execution time becomes a hot research topic since the 1990s, consequently to the increasing complexity of the hardware platforms. A survey in 2008 by Wilhelm et al. [257] is widely used as the reference for WCET

analyses. Even if, at the time of writing of this thesis, the article is 12 years old, it is still valid, and most all the techniques presented in that manuscript are currently used in both academia and the industrial world.

The traditional static methods to estimate the WCET usually require four inputs: 1) the executable program in binary form, 2) the user annotations (usually in the source code), 3) the processor model, and 4) the semantic of the instructions. From the executable, a *Control-Flow Graph (CFG)* is built and analyzed thanks to the user annotations to determine variable values, loop bounds, and any other information needed to perform the architectural analysis. Then, the the architectural analysis output is used to determine a global bound on the execution time. Since this thesis is not focused on static methods, we do not describe all the literature regarding the single phases of static analyses and tools used, which could be found in the survey mentioned above [257].

2.5.2 Probabilistic WCET approaches

The early probabilistic approaches to WCET estimation date back to the first years of the 2000s. Edgar and Burns [79] proposed in 2001 the first work exploiting EVT to compute the pWCET. The paper explained how to estimate the Gumbel distribution parameters, the confidence of the obtained WCET, and how to apply the scheduling analyses having a pWCET instead of a scalar WCET value. The next year, in 2002, another article by Bernat et al. [27] was published, introducing the concept of probabilistic hard real-time systems. The authors presented a mixed measurement-based and static method, based on the convolution of execution time random variables of the single blocks of the CFG. These two papers can be considered the seminal articles for probabilistic real-time computing research activities. It is worth citing that previous works on stochastic scheduling of real-time tasks exist [100, 247, 5], although not directly related to the pWCET concept. About a decade later, the pWCET techniques gained significant attention for the research community, which developed many theoretical tools and experimental evidence. The interest is still very high nowadays, due to the numerous open challenges summarized by Jiménez Gil et al. [139] in 2017. Two recent comprehensive surveys, [48] and [66], provided a general overview of probabilistic real-time WCET analyses research of the last years while, regarding the probabilistic task scheduling, another recent survey is available [69].

EVT hypotheses. The i.i.d. hypothesis described in Section 2.2.3 received attention since the first works on pWCET [79]. Griffin and Burns [106] discussed the possibility of satisfying i.i.d. in real systems, concluding that appropriate methods must be used to ensure this property. A new sampling method was proposed to overcome the issue of i.i.d. [174], but its effect possibly hinders the satisfaction of other hypotheses (i.e., representativity). The main efforts on the satisfaction of the i.i.d. hypothesis have been focused on creating a cache-randomized architecture [209]. This was one of the main goals of the European FP7 project PROARTIS [47]. Cache-randomized architecture tries to make the execution time independent across different executions by randomizing the hit/miss behavior of the caches. The problem has been faced with both hardware [148, 150, 242, 128] and software solutions [149, 151]. Other randomized hardware components were proposed, for instance, buses for multi-cores [136]. Later, in 2017, Santinelli et al. [228] showed that the i.i.d. hypothesis could be relaxed in favor of three hypotheses, which together represent a less strict requirement than the full i.i.d. hypothesis. The MDA hypothesis is the one less studied in the context of probabilistic real-time. The low interest is due to the difficulty to map this property to a real platform or software property. It was formally described by Santinelli et al. [228] and included in the experimental evaluation of other works [112, 90, 77, 110]. Regarding the representativity hypothesis, some researchers focused on how to analyze multi-path programs [61] to obtain the pWCET and how epistemic variability relates to the MBPTA analysis [68]. The presence of multiples "modes" in applications is a source of epistemic uncertainty and, consequently, hinders the pWCET estimation, as shown by Guet et al. [112]. Experimental studies [167] indicated that all the GEV models are plausible and that the hardware randomization is not sufficient to achieve representativity, as pointed out, instead, by other researchers [186]. In parallel to representativity, the reproducibility was proposed [181] as another hypothesis, i.e., the ability of the pWCET method to generate the same distribution according to different input time traces belonging to the same system.

Analysis process and uncertainties. Several approaches have been proposed by many authors to estimate the pWCET distribution via MBPTA. Among the works, it is worth mentioning the Santinelli et al. [228] paper, which recapped the whole estimation process with a special focus on the statistical tests to be executed. Other articles described how to execute the MBPTA process to estimate a safe distribution focusing on the ap-

plicability to real systems [230, 167]. Abella et al. [4] proposed in 2017 a new method called MBPTA-CV to estimate the extreme distribution. Regarding the quantification of uncertainties on the pWCET distribution, not many previous works are available. Silva et al. [238] proposed a method to estimate with a confidence region the parameters of the extreme distribution. Other works (e.g., [54]) suggested a simplification of the GEV model by considering just the Gumbel distribution.

2.5.3 Timing requirements in HPC

The performance of HPC systems is typically measured in terms of *throughput* – i.e., the amount of work completed in a given time unit – as introduced in Section 1.4.2. The most common measurement unit for the throughput is the FLOPS (Floating-Point Operations per Second). Currently, the top supercomputers in the world offer performance in the *petaFLOPS* (10^{15}) order of magnitude. On the other hand, the research is striving towards *exascale* compliant solutions [233], taking into account the power consumption envelope and trying to go beyond the scalability limits of current infrastructures.

The throughput is not the only metric considered in HPC, especially from the users' standpoint: they are usually more interested that their jobs complete successfully and in a short time. Two surveys, [133] and [241], provided an overview of the current state-of-the-art strategies to deal with application requirements and resource availability, in both HPC and Cloud computing. The common goal of such techniques is to optimize or to fulfill a *Quality-of-Service (QoS)* metric defined by the user. The actual QoS definitions may vary, but they usually include system throughput, average execution time, infrastructure cost, reliability, and availability metrics. However, all the QoS or performance metrics are mostly considered in an average sense, i.e., the violation of user requirements is just a *Service Level Agreement* degradation [8].

CHAPTER 3

On the Estimation of a Correct Distribution

In probabilistic real-time, the reliability of the estimated pWCET distribution is the fundamental metric. The estimation process, described in the first section of this chapter, is subject to errors that must be quantified. This chapter provides an overview of the correct estimation process for the pWCET and how we can derive reliability from statistical testing procedures.

3.1 The EVT Estimation Process

To estimate the pWCET by using the EVT, the two filtering methods BM and PoT exist, as we described in the previous Section 2.2.2. Once the input measurements have been

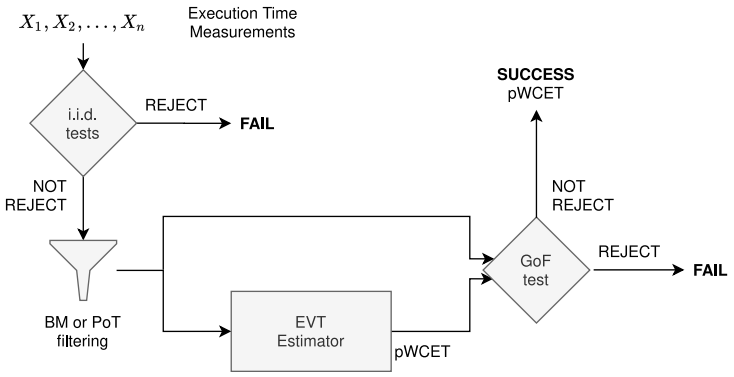


Figure 3.1: *The simplified scheme of the EVT process.*

filtered, distribution estimators (e.g., MLE or PWM) provide the pWCET parameters' values. However, the overall process leading to the pWCET distribution generation is more sophisticated than just running an estimator. The main components of this process are depicted in Figure 3.1:

1. The sequence of execution time measurements $X_1, X_2, ..., X_n$ is tested to verify the compliance of the i.i.d. hypothesis or all its sub-hypotheses.

2. If the sequence is considered valid, the process proceeds, and the BM or PoT technique is applied to the time measurements.

3. The resulting sequence is then fed to an EVT estimator, which estimates a GEVD or GPD distribution (according to the choice BM/PoT).

4. A Goodness-of-Fit test is then applied to check if the input data of the estimator corresponds to the pWCET distribution estimated.

5. If the test does not reject the hypothesis, the distribution is considered valid and the process successfully stops.

This chapter focuses on the analysis of the reliability aspects of the EVT process applied in probabilistic real-time. In particular, Section 3.2 shows a common error in selecting the appropriate GEVD/GPD model, while Section 3.3 and Section 3.4 explores the uncertainties on statistical testing and estimation procedures.

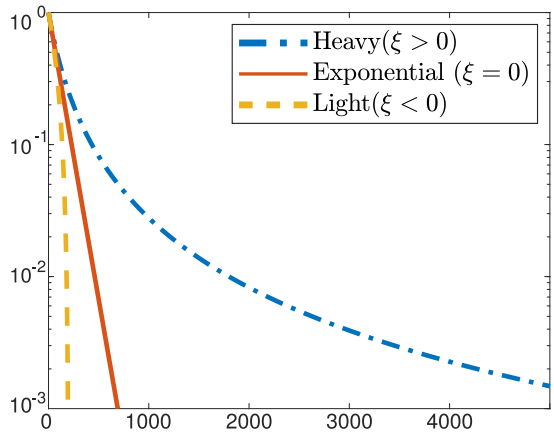


Figure 3.2: *The asymptotic behaviors of the GP3D*$(0, 100, \xi)$ *cdf* $F(x)$. *(Original source: [215])*

3.2 A Common Misconception in Literature

In probabilistic real-time research, some authors argued that the exponential-tail distributions – i.e., GEVD/GPD with $\xi = 0$ – are good candidates for fitting the pWCET distribution. These claims are motivated by empirical demonstrations because it is almost impossible to provide formal proofs. Conversely, other experiments [228, 167] showed that it is worth considering the $\xi < 0$ and $\xi > 0$ cases. These authors concluded that the exponential distribution might not be representative of all the scenarios.

Figure 3.2 shows the tails of the extreme distributions, depicting the the ccdf for different values of the shape parameter ξ. This information has been recently used [4, 14, 7] to state that exponential-tail distribution upper-bounds the light-tail distribution. Formally:

$$1 - F_{\mathscr{G}_1}(x) > 1 - F_{\mathscr{G}_2}(x)$$

where $\mathscr{G}_1 \sim \text{GP3D}(\mu, \sigma, 0)$ and $\mathscr{G}_2 \sim \text{GP3D}(\mu, \sigma, \xi < 0)$. This relation is the same for the GEVD. In the probabilistic real-time context, this means that using a distribution with exponential-tail ($\xi = 0$) to upper-bound a light-tail ($\xi < 0$) should not lead to pWCET underestimation. This upper-bound is valid because, for a certain WCET x, the probability to incur in a longer execution time value ($1 - F_{\mathscr{G}_1}(x)$) is always higher than the one computed with light-tail ($1 - F_{\mathscr{G}_2}(x)$). Vice versa, for a given probability

p, the WCET estimated by exponential tail ($F^{-1}_{\mathscr{G}_1}(1-p)$) is always higher than the one computed by light-tail ($F^{-1}_{\mathscr{G}_2}(1-p)$).

These results are valid only if the other parameter(s) of the distribution remains unchanged. For example, let us assume to fit the complete distribution GP3D(μ, σ, ξ) (with $\xi < 0$) and then enforce $\xi = 0$ obtaining GP3D$(\mu, \sigma, 0)$. In this case, the second distribution upper-bounds the first one, and the pWCET is not under-estimated. However, enforcing $\xi = 0$ before performing the distribution fitting may lead to different values of μ and σ, with respect to the real distribution of the data. This possibility would invalidate the previous result, carrying out potentially unsafe pWCET estimations. The estimation procedure, in fact, provides the μ and σ values that best fit the input data. In general, these are different from the ones that would have been computed without enforcing $\xi = 0$. Moreover, if the estimator is unbiased with respect to the extreme population's mean, the resulting distribution is always unsafe, as proven below.

3.2.1 Formal proofs

Parameters shift effects on the first moment. To prove the above statements, we initially use a two-parameters Generalized Pareto Distribution GP2D(σ, ξ) (this to simplify the calculus). The extension to GP3D and GEVD is then discussed at the end of this subsection.

Let $Y_1, ..., Y_m$ be the m maximum measurements of execution time distributed under $\mathscr{G} \sim$ GP2D(σ, ξ), as result of the Peak-over-Threshold algorithm, with $Y_i > u$. By assuming the estimator as unbiased, the mean value of Y_i matches the expected value of the GP2D distribution:

$$E[Y] = \frac{\sigma}{1-\xi}$$

It follows that, for the exponential distribution case ($\xi = 0$), the expected value is:

$$E[Y] = \sigma_{\xi=0}$$

Upper-bounding with the exponential-tail distribution means forcing $\xi = 0$ for the same set of data, while maintaining the same expected value $E[Y]$. The consequent effect is to obtain:

$$\sigma_{\xi=0} = \frac{\sigma_{\xi<0}}{1-\xi} \tag{3.1}$$

In case the data are distributed with $\xi < 0$, the simplification $\xi = 0$ leads to estimate $\sigma_{\xi=0} > \sigma_{\xi<0}$, instead of the real σ value. However, the following section provides the

proof that this scale parameter skew may lead to unsafe pWCET estimations.

Proof of failure of exponential-tail upper-bounding. Given the definition of cdf $F(x) = P(X \leq x)$, upper-bounding a distribution in the context of MBPTA means that the relation $F'(x) < F(x)$ holds for any x.

Proof. Let $\bar{F} = 1 - F(x) = P(X > x)$ be the complementary cdf. A safe upper-bound for pWCET has to guarantee the conservative relation $P'(X > x) > P(X > x) \quad \forall x$. From this, it is possible to obtain $1 - F'(x) > 1 - F(x)$ and, in turn, $F'(x) < F(x)$. □

In our scenario, $F'(x)$ corresponds to the upper-bound with $\xi = 0$, while $F(x)$ is the cdf of the real distribution, with ξ assumed to be set to an unknown negative value $\xi < 0$. By expanding the cdfs we obtain:

$$1 - e^{-\frac{x}{\sigma_{\xi=0}}} < 1 - \left(1 + \xi \frac{x}{\sigma_{\xi<0}}\right)^{-1/\xi} \tag{3.2}$$

This inequality must hold for any x, but since we deal with a positive variable (execution time), this holds only for $x > 0$. As a consequence of Equation (3.1), it is possible to state that this inequality is not true in general.

Proof. Removing the constant term and multiplying by -1:

$$e^{-\frac{x}{\sigma_{\xi=0}}} > \left(1 + \xi \frac{x}{\sigma_{\xi<0}}\right)^{-1/\xi} \tag{3.3}$$

Let us now replace $\sigma_{\xi<0}$ according to Equation (3.1):

$$e^{-\frac{x}{\sigma_{\xi=0}}} > \left(1 + \frac{\xi}{1-\xi} \frac{x}{\sigma_{\xi=0}}\right)^{-1/\xi} \tag{3.4}$$

The equation corresponding to this inequality has a trivial zero for $\bar{x}_1 = 0$, but it has another solution $\bar{x}_2$ for $x > 0$:

$$\bar{x}_2 = \frac{\sigma_{\xi=0}}{\xi} \left(-\mathrm{W}\left[(\xi - 1)\left(e^{\frac{1}{\xi}-1}\right)^{-\xi}\right] + \xi - 1\right) \tag{3.5}$$

where $\mathrm{W}[\cdot]$ is the *Lambert W function*. Since $\xi < 0$, the argument of $\mathrm{W}[\cdot]$ is negative, as well as $\mathrm{W}[\cdot]$. While, being $|\,\mathrm{W}[\cdot]\,| < 1$, $\bar{x}$ assumes a positive value. This means that there is a second zero ($\bar{x}_2 > 0$) and consequently at least a value ($\bar{x}_2$) that violates the inequality of Equation (3.2). □

As detailed later in Section 3.2.2, there is a continuous interval – i.e., infinite points – that violates the inequality. Providing analytical proof for it is not straightforward because of the complexity introduced by the *Lambert W function*. Luckily, this is not necessary to demonstrate that the exponential-tail upper-bounding is unsafe. The counterexample obtained by numerical evaluation is, in fact, sufficient to prove this.

Applicability to GEVD and GPD 3-parameters. The exponential-tail bounding problem also exists in GP3D and GEVD since both share the same tail behavior presented in Figure 3.2. The previous proofs can be easily ported to GP3D and GEVD distributions. Following the same approach of the GP2D version, the mean value of GP3D(μ, σ, ξ) is $E[Y] = \mu + \frac{\sigma}{1-\xi}$. If the mean value does not change once the estimation runs with $\xi = 0$, the results are the same as the provided proof. If $\mu_{\xi=0} < \mu_{\xi<0}$, then the error is higher and the estimation becomes unsafe. If $\mu_{\xi=0} > \mu_{\xi<0}$, then nothing can be said without an accurate analysis of the specific case.

Similarly to the GPD case, the GEVD condition for safe upper-bounding is:

$$e^{1-e^{-\frac{x}{\sigma_{\xi=0}}}} < e^{1-\left(1+\xi\frac{x}{\sigma_{\xi<0}}\right)^{-1/\xi}}$$

and since $e^{f(x)} < e^{g(x)} \leftrightarrow f(x) < g(x)$, it results that:

$$1 - e^{-\frac{x}{\sigma_{\xi=0}}} < 1 - \left(1 + \xi\frac{x}{\sigma_{\xi<0}}\right)^{-1/\xi}$$

that is exactly the same as Equation (3.2). For this reason, the previous analysis can also be applied to GEVD.

3.2.2 Numerical evaluation

To clarify the previous equations and provide a counterexample to the exponential upper-bounding claim, we consider a GP2D distribution with $\sigma = 100, \xi = 0$ as reference, and we compare it to other two GP2D distributions with $\xi = -0.4$ and $\xi = -0.8$. The scale parameter σ is computed according to Equation (3.1). The respective ccdfs are depicted in Figure 3.3a. The exponential GPD ($\xi = 0$) upper-bounds both distributions only starting from the value $\bar{x}_2 \approx 179$. The absolute value of $\bar{x}_2$ is not negligible: in this case, we consider $\sigma = 100$, that is the mean value of the extremes, and the exponential tail upper-bound becomes safe only after nearly the double of it. Figure 3.3b

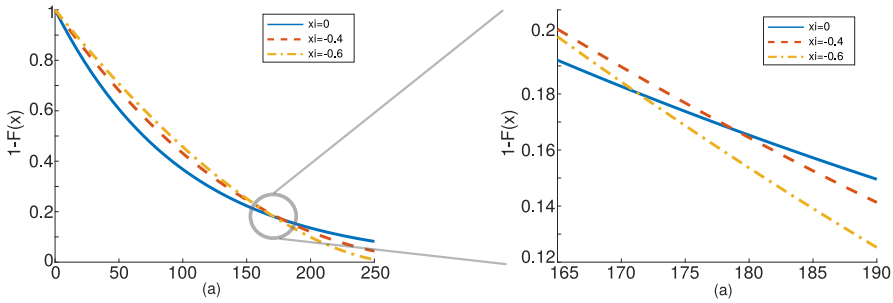


Figure 3.3: *The ccdf varying ξ of two-parameter GPD. The reference distribution GP2D($100, 0$) is represented by the solid blue line. The plot (b) is the zoom of the plot (a) at the ccdf intersection. (Original source: [215])*

zooms in the intersection point of the previous Figure 3.3a. As expected, increasing the value of ξ towards 0 produces a smaller error in the difference of cdf between $\xi < 0$ and $\xi = 0$. However, it also shifts the intersection point – i.e., the point where the upper-bound is safe – towards $+\infty$.

To better investigate the last result, we compute the maximum error between a GP2D($\sigma_{\xi<0}, \xi < 0$) and the reference distribution GP2D($100, 0$), by varying the value of ξ from -1 to 0. The result is depicted in Figure 3.4a. The blue (solid) line represents the value $\bar{x}_2$, after which the upper-bound is safe, while the red (dashed) line shows the maximum error compared to the reference distribution. The $\bar{x}_2$ value increases with a peculiar slope, for which we provide Figure 3.4b to show the trend for small values of ξ, while the maximum error has a quasi-linear trend. The key point here is to observe that there is no upper-bound for $\bar{x}_2$. Even if there exists a point $\bar{x}_2$ from which the upper-bounding is safe, it is not possible to know it without knowing the real value of ξ. This leads to an uncertainty on the pWCET estimation that can not be accepted in hard real-time systems. On the other hand, when ξ is close to 0 and $\bar{x}_2$ increases towards infinite, the error decreases, but we still need to know ξ in order to quantitatively estimate both.

3.2.3 Lesson learned

Some recent works proposed to upper-bound the pWCET extreme value distribution when it has a light-tail ($\xi < 0$) with its exponential tail version ($\xi = 0$). While it

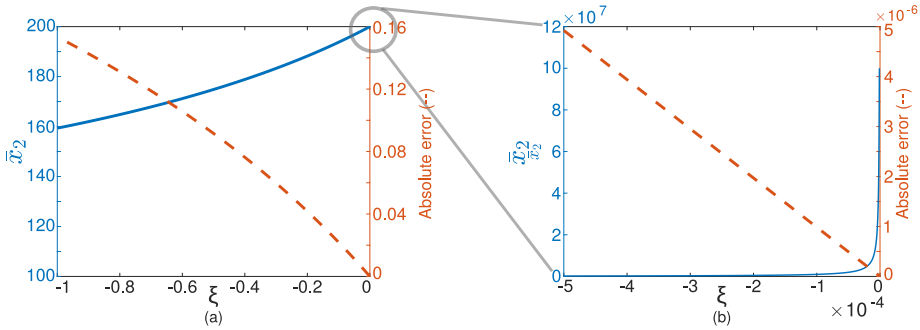


Figure 3.4: *Analysis of the safety point $\bar{x}_2$ (blue solid line) and the maximum absolute error (red dashed line), varying the real $\xi < 0$ and compared to $\xi = 0$ case. The plot (b) is the zoom of the plot (a) when ξ has a magnitude of 10^{-4}. (Original source: [215])*

simplifies the overall process, it may underestimate the pWCET value if the ξ value is bounded *a priori* of the estimation phase. Previous works neglected this aspect by assuming the upper-bounding safe. As shown by the previous sections, instead, its validity holds only for WCET values greater than a certain unknown value $\bar{x}_2$. For these reasons, we should not blindly consider exponential-tail distributions on critical systems without taking into account the ξ parameter. Rather, the ξ must be forced after the estimation phase, the error magnitude must be estimated, or the WCET must be proven to be large enough to guarantee a safe upper-bound.

3.3 The Role of Statistical Testing

The introductory section of this chapter (Section 3.1) presented the EVT estimation process and highlighted the presence of statistical tests to be run to check the EVT hypotheses described in Section 2.2.3. We can distinguish two categories of statistical tests interesting for probabilistic real-time: the tests to check the i.i.d. hypothesis (or its sub-hypotheses) and the Goodness-of-Fit tests. Before describing in detail these tests, let us recap a minimal statistical background on statistical tests. A *statistical test* is a method used in statistical inference to decide the truthfulness or the falseness of the statistical hypotheses of a hypothesis scheme, usually composed of one null hypothesis (H_0) and one or more alternative hypotheses ($H_1, H_2, ...$). A statistical test is an

algorithmic procedure performed on arbitrary input values, such as functions, scalars, etc. The output is usually the *p-value* or, alternatively, the *statistic*. This output is used to decide whether there is sufficient evidence to reject the null hypothesis or not. It is important to remark that a test can never *accept* the null hypothesis: it can only tell us if the H_0 hypothesis is false, while it cannot prove its truthfulness. This detail has a critical impact on pWCET reliability, as explained in Section 3.3.3. The p-value is an indicator which compared to the significance level of the test α (chosen by the experimenter) tells us how to decide the test result:

- p-value $> \alpha$: no evidence, H_0 cannot be rejected.
- p-value $\leq \alpha$: strong evidence, H_0 is rejected.

Alternatively, and equivalently, it is possible to use the *statistic* D (computed by the test from the inputs) and the *critical value* CV (computed given the test and given the α, but not the inputs). They are used in a similar way of the p-value:

- $D < CV$: no evidence, H_0 cannot be rejected.
- $D \geq CV$: strong evidence, H_0 is rejected.

The two strategies are exactly equivalent and interchangeable. In this thesis, to set a convention, we use the latter version, i.e., the statistic/critical value evaluation.

3.3.1 The i.i.d. tests

The i.i.d. hypothesis (described in Section 2.2.3) is checked with a statistical test, by directly executing it on the sequence of time measurements before the application of BM or PoT.

Previous works. Most of the scientific papers in probabilistic real-time field check the EVT applicability by directly verifying the i.i.d. hypothesis. The execution times independence is usually checked by performing a Ljung-Box test [4, 14, 25, 91]. This is problematic for two reasons: we already described that pure independence is a too strict requirement [230] and the Ljung-Box test checks for the presence of a particular form of independence, i.e., the serial correlation. Other approaches use the Wald-Wolfowitz test (also called *runs test*) – e.g., [152] – which suffers from the same problems: on the

one hand it is used as a "pure" independence test, on the other hand, its detection capabilities refer to a particular dependence around the median value. Finally, to test the identically distributed hypothesis, the two-sample Kolmogorov-Smirnov (KS) test has been extensively used[1] [14, 25, 91, 152, 239, 255, 256]. This test consists of dividing the sample into two parts of equal size and then comparing each other with the KS test to check whether they have the same distribution. While this can be effective against some form of violation of the identically distributed hypothesis, it is easy to build counter-examples that show this test is ineffective and improperly used in our scenario. For example, let consider $x_1, x_2, ..., x_{200}$ as our execution time trace, drawn from a sequence of random variable distributed as follow: $X_{20k+1}, ..., X_{20k+10}$, for $k = 0, ..., 9$ from a distribution $\mathscr{D}_1$, while the values $X_{20k+11}, ..., X_{20k+20}$, for $k = 0, ..., 9$ from a distribution $\mathscr{D}_2$. Applying the KS test using the first 100 elements as the first sample and the last 100 elements as the second sample, it would result in a false-negative, being unable to detect the non-identically distributed hypothesis. This because, while the two joint cumulative distribution functions of the samples of 100 elements each are similar, their inside random variables are not identically distributed.

A selection of tests for each sub-hypothesis. As previously explained, the i.i.d. hypothesis can be split into three sub-hypotheses. The first one is *stationarity*. In this regard, in literature, there is a large availability of unit-root tests – a particular case of non-stationarity – but a lower number of general stationary tests. Given a time series $\mathcal{X} = \{X_1, X_2, ..., X_n\}$, we are looking for a test with the following hypothesis scheme:

H_0 : the time series $\mathcal{X}$ is stationary

H_1 : the time series $\mathcal{X}$ is not stationary

The most used test is the Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) test [154]. A variant of KPSS considering the relaxed null hypothesis “the time series is stationary or trend stationary” exists. For the EVT hypothesis of stationarity, we are interested in the tightest one, thus we do not consider this variant. The formula for KPSS statistic is available in Appendix B.1. The critical values can be computed through the interpolation of the tabular data proposed by Kwiatkowski et al. [154] or by using Monte Carlo approaches.

[1] Not to be confused with the one-sample KS test used as Goodness-of-Fit, described in the following subsection.

To test the *short-range* dependence of data, we selected the Brock, Dechert, Scheinkman and LeBaron (BDS) test [39]. For probabilistic real-time, we decided to select this test because it is a *portmanteau test*, i.e., the null hypothesis is well specified, but the alternative hypothesis is not. Given a time series $\mathcal{X} = \{X_1, X_2, ..., X_n\}$:

H_0 : the time series $\mathcal{X}$ is independent

H_1 : the time series $\mathcal{X}$ has some sort of dependency

Most of the other available tests detect specific sort of dependency (e.g., serial correlation or deterministic chaos). Therefore, we decided to choose the test with the most general detection capability. The formula for BDS statistic is available in Appendix B.2. The critical values can be computed via numerical methods.
The *Hurst Exponent* (H) is the traditional index used to measure the long-term memory of a time series in financial applications [207]. H is a number in the range $[0; 1]$ indicating the degree of *long-term dependence*: $H = 0.5$ means a perfectly random and uncorrelated time series, while $H < 0.5$ or $H > 0.5$ indicates a negative or positive correlated time series, respectively. However, performing a statistical test on H is nontrivial [58] and, to the best of our knowledge, it does not exist a well-assessed test. The Hurst index is computed from the R/S statistic equation [132], that can, instead, be directly used as a test:

H_0 : the time series $\mathcal{X}$ has no long-range dependency

H_1 : the time series $\mathcal{X}$ has long-range dependency

This test is sensitive to long-range dependency but also to short-range dependency. An alternative to this formulation is the Lo's modified version [170] that has been developed to limit the influence of short-range dependency in the R/S equation, and it is commonly used. However, this version reduces the statistical power of the test [246], which is not desirable in this context. Therefore, by using the unmodified R/S statistic, this test may detect a short-range dependency partially overlapping the BDS test, thus providing pessimistic but safe results. The formula for R/S statistic is available in Appendix B.3. The critical values can be computed via numerical methods.

3.3.2 Goodness-of-Fit tests

The *Goodness-of-Fit (GoF)* test at the end of the EVT estimation process aims at finding any error in the estimation of the pWCET. This covers not only the MDA hypothesis

Test output	Reality	Informal Name	Formal Name	Symbol
H_0 is not rejected	H_0 is false	false-positive	Type I	α
H_1 is rejected	H_0 is true	false-negative	Type II	β

Table 3.1: *Recap of the errors a statistical test can make.*

described in Section 2.2.3, but also estimation errors, uncertainties, and mistakes in the EVT process. The most famous GoF tests are the Chi-Squared (CS) [97], Kolmogorov-Smirnov (KS) [142], Cramer-von Mises (CvM) [64], Anderson-Darling (AD) [11], and the Modified Anderson-Darling (MAD) [126]. Such tests have the same hypothesis scheme:

$$H_0 : \ G(x) = \hat{F}_n(x) \text{ (null hypothesis)}$$

$$H_1 : \ G(x) \neq \hat{F}_n(x) \text{ (alternative hypothesis)}$$

where $G(x)$ is the cdf of the extreme value distribution output of the EVT estimator and $\hat{F}_n(x)$ is the ecdf built on the n time samples output of the BM or PoT filter according to Equation (2.1).

3.3.3 How statistical power impacts reliability

The testing procedures are built to detect any violation of the null hypothesis. Such detection can fail in two ways: (1) The null hypothesis H_0 is rejected, in favor of H_1, but H_0 is actually true (false-positive); (2) The null hypothesis H_0 is not rejected, but it is actually false (false-negative). The ratio of false-positive errors, called *Type I* errors, is selected by the experimenter tuning the significance level α. The ratio of false-negative errors, or *Type II* errors, depends, instead, on several factors, and it cannot be easily controlled or estimated. We refer to it with the letter β. These errors and symbols are summarized in Table 3.1.

The impossibility to control β forces the statisticians to say that a statistical test can never "accept" the null hypothesis: If the hypothesis is not rejected and the value of β is unknown, no conclusions could be drawn. In the pWCET world, this means that if a GoF test rejects a pWCET distribution, we are sure with $(1 - \alpha)$ confidence that the actual pWCET distribution is wrong. In this case, the pWCET estimation stops, and

k	**50**	**100**	**200**	**500**	**1000**
KS	0.03	0.29	0.74	$1 - 10^{-3}$	$1 - 10^{-7}$
AD	0.49	0.86	0.99	$1 - 10^{-8}$	$> 1 - 10^{-9}$
MAD	0.26	0.80	0.95	$1 - 10^{-8}$	$> 1 - 10^{-9}$

Table 3.2: *Minimum achieved statistical power among all the test scenarios described in Section 7.2 for $\alpha = 0.05$. (Original source: [218])*

the safety is guaranteed. On the other hand, if the pWCET is not rejected, nothing can be said about its validity. The latter case is a safety problem: even if there is no clear evidence that the estimated pWCET is invalid, we have no confidence bound on this statement, making the GoF test completely useless in terms of reliability. A possible solution could be to invert the H_0 and H_1 hypotheses so that we can control the ratio of invalid distributions. However, no statistical test is known to exist with such an inverted hypothesis scheme.

Consequently, to estimate the confidence of the results, we define the following scalar value:

Definition 3.3.1 (Statistical Power)**.** The complement of the Type-II error is called *statistical power*:

$$W = 1 - \beta = P(\text{reject } H_0 | H_0 \text{ is false}) \tag{3.6}$$

The statistical power W of a GoF test depends on several factors: 1) the significance level α, 2) the test sample size, 3) the test procedure itself, 4) the shape of the real (unknown) distribution. The statistical power increases when α increases: we can decrease the false-negative rate by increasing of false-positive ratio. Another option to decrease the rate of false-negative results is to increase the number of samples used for testing. Since increasing the α is not advantageous, we determine the minimum sample size that is required to reach certain confidence on the test result. Such *power analysis*, for the best of our knowledge, has never been performed in a probabilistic real-time context. How this analysis is performed and the resulting dataset are presented later in Section 7.2 and in the paper by Reghenzani et al. [217].

The availability of the statistical power allows us to provide a lower-bound on the value of the confidence of the test result and, in turn, on the pWCET confidence. This

result is an enabler for the estimation of the overall reliability of the pWCET distribution. It is possible to compare the three tests KS, AD, and MAD (CS and CvM have been excluded for a reason explained in Section 7.2.1) and provide the minimum number of samples required to obtain the given confidence on the result of the GoF test. From the dataset, we computed the statistical power values presented in Table 3.2 varying the number of samples k. It should be noted that the number of samples refers to the size of the input of the GoF test, i.e., the size of the set of measurement outputs of the BM/PoT filtering and not to the amount of the original time measurements. From these results, it is possible to conclude the following statements: (1) KS has, in general, less statistical power than the other two tests, while AD and MAD present similar behavior for a large number of samples; (2) To obtain very high confidence on the result of the tests, it is necessary to use a sample of at least 1000 time measurements. This size refers to the sample of observed time measurements in the tail of the distribution, i.e., after applying the BM/PoT filter.

Many previous works on probabilistic real-time empirically selected a number of samples usually not higher than 100, basing their choice on statistical works. However, in probabilistic real-time, we need large confidence and a very low violation probability, and a low sample size makes the non-rejecting result of the GoF test very untrustworthy and with low significance. The experimenter should select the sample size depending on the desired confidence level, and he or she should pay attention if KS is used since the number of samples required to obtain sufficient confidence could be very large. It should also be noted that (M)AD tests rely on an internal Monte Carlo estimation; for this reason, they have an intrinsic uncertainty on the test result, that should also be considered and adequately analyzed.

3.3.4 The significance level in case of multiple tests

We already explained the significance level α, which is a parameter chosen by the designer of the statistical test procedure, and it corresponds to the false-positive ratio of the test. To check the i.i.d. sub-hypotheses with the tests presented in Section 3.3.1, it is necessary to perform a sequence of three statistical tests. In general, executing multiple hypothesis tests on the same data increases the false-positive rate on the null hypothesis rejection of the overall test [24]:

$$\alpha_{\text{global}} = 1 - (1 - \alpha)^n \tag{3.7}$$

where n is the number of tests (in our case $n = 3$).

For common values $\alpha = 0.05$ and $\alpha = 0.01$, the resulting global significance levels are respectively $\alpha_{\text{global}} \approx 0.14$ and $\alpha_{\text{global}} \approx 0.03$. The real significance level is thus higher than the single test levels, entailing a higher false-positive rate. Rejecting a sample implies that the pWCET estimation process stops because it detects that not all the hypotheses are satisfied, preventing the estimation of an unsafe pWCET. The false-positive rate makes the characterization of the capability of a hardware/software architecture to fulfill the EVT hypotheses difficult: obtaining a rejection result by running one single time a statistical test does not necessarily mean that the architecture is non-compliant with EVT hypotheses. The test has to be run multiple times to perform a correct evaluation, and the final outcome has to be decided by looking at the overall reject/not-reject ratio: a rejection ratio close to α identifies a system that verifies the EVT hypotheses; in contrast, a higher ratio represents a violation of the EVT hypotheses.

In statistical literature, several methods exist to reduce the α_{global} value when multiple tests are performed. The most famous one is the Bonferroni correction [36]. However, all such approaches have the negative effect of reducing the statistical power [193] that, as explained in the previous section, may hinder the reliability of our results. Because the number of tests is fixed and low (3), it does not worth trading a lower α_{global} value with lower statistical power.

3.4 Region of Acceptance

The GoF test is, however, not a panacea for the pWCET safety. The estimation of the extreme value distribution $\mathscr{G}$, which results from the EVT process, is naturally subject to errors: the necessary condition to obtain the exact distribution for any estimator algorithm is to have infinite measurements, that is clearly not realistic. To this extent, the estimator routine provides us the GEVD (or GPD) parameters tuple $(\overline{\mu}, \overline{\sigma}, \overline{\xi})$ that can be rewritten as $(\mu^{\circledast} + \epsilon_\mu, \sigma^{\circledast} + \epsilon_\sigma, \xi^{\circledast} + \epsilon_\xi)$, where $(\mu^{\circledast}, \sigma^{\circledast}, \xi^{\circledast})$ is the exact unknown distribution and the symbols $(\epsilon_\mu, \epsilon_\sigma, \epsilon_\xi)$ represent the unknown errors in our estimation. As discussed in the previous section, the goal of the GoF tests is to detect these uncertainties and to reject the estimated distribution when the errors are excessively high. However, because of the finite number of measurements, the GoF test is also imperfect, i.e. it is not able to reject the distributions when $(\epsilon_\mu, \epsilon_\sigma, \epsilon_\xi)$ are too low to be detected.

For this reason, a multi-dimensional cloud of points in the GEVD (or GPD) parameters space exists, which represents the distributions that are not rejected by the GoF test. In this section, we explore this region and, we describe how its statistical properties affect the reliability of our pWCET estimation.

3.4.1 Definitions and exploration

For the purposes of this section, let us assume that we have already performed the EVT estimation of the pWCET distribution: the input sequence of the execution time measurements has been filtered by the BM approach to obtain a sequence $\mathcal{X}$, from which we have estimated the GEVD distribution[2] $\mathscr{G}$. Before providing the formal formulation for the uncertainties of the GEVD parameters space, we specify the following helper function:

Definition 3.4.1 (Test result function)**.** Given a statistical test identified by its statistic function $D(\cdot)$, the sequence of time measurements $\mathcal{X}$, an estimated distribution $\mathscr{G}$, and the critical value[3] CV, its *test result function* is defined as:

$$T(\mathcal{X}, \mathscr{G}) := \begin{cases} 1 & \text{if } D(\mathcal{X}, \mathscr{G}) \geq CV(\mathscr{G}) \\ 0 & \text{if } D(\mathcal{X}, \mathscr{G}) < CV(\mathscr{G}) \end{cases}$$

■

This definition is a formal notation to state the result of a statistical test: the rejection of the null hypothesis ($T(\mathcal{X}, \mathscr{G}) = 1$) – i.e. the values $\mathcal{X}$ show a strong evidence that have not been drawn from the distribution $\mathscr{G}$ – or the non-rejection of the null hypothesis ($T(\mathcal{X}, \mathscr{G}) = 0$) – i.e. the values $\mathcal{X}$ have been *probably* drawn from the distribution $\mathscr{G}$. This definition can be used to identify the region of points in the GEVD parameter space for which the test accepts the null hypothesis. Note that this is an abuse of the common notation of hypothesis testing we described in Section 3.3. However, in this section, we abuse the *accepts* notation for clarity purposes.

By exploiting the test result function, we can formally define the three-dimensional cloud of points in the GEVD parameters space:

[2] In this section we limit the discussion to the BM/GEVD case.

[3] For some GoF statistical tests, the critical value depends on the reference distribution (e.g., the Anderson-Darling test). We write it as a function of the reference distribution $CV(\mathscr{G})$ or simply CV.

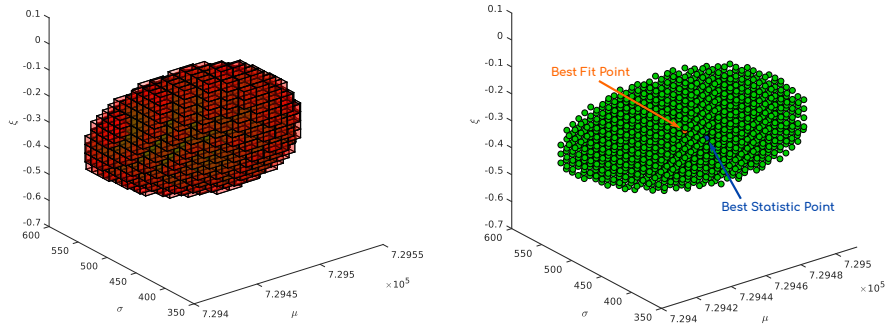


Figure 3.5: *The Region of Acceptance $R(\mathcal{X})$ plotted for a real sequence of time measurements $\mathcal{X}$: (a) by color representing the value of statistics $D(\cdot)$ (color scale red-green: the red color identifies values near the critical value, the green color identifies – in the middle of the region – values far form it); (b) by green dots where the $T(\cdot) = 0$ and the highlighted BFP and BSP points. (Original source: [222])*

Definition 3.4.2 (Region of Acceptance)**.** Given a sequence of execution time measurements $\mathcal{X}$, the Region of Acceptance of a statistical test with test result function T for an extreme distribution is the cloud of points R defined as:

$$R(\mathcal{X}) := \{(\mu, \sigma, \xi) \in \mathbb{R}^3 : T(\mathcal{X}, (\mu, \sigma, \xi)) = 0\}$$

■

A visual example of the Region of Acceptance is depicted in Figure 3.5. To shorten the notation, we sometimes avoid to write the measurements sequence parameter: $R = R(\mathcal{X})$. In the same parameters space, it is possible to identify the tuple $(\overline{\mu}, \overline{\sigma}, \overline{\xi})$ as the point that represents the output of the EVT estimator. We call this point the *Best Fit Point (BFP)*. This point may or may not be inside the region R, i.e. it may be accepted ($T(\cdot) = 0$) or rejected by the GoF test ($T(\cdot) = 1$). If this point is outside the region, the estimator failed to provide a valid distribution. The region may even not exist, e.g., when the original distribution is not in the domain-of-attraction of any GEVD distribution. It is worth reminding that the GoF test has a *false positive rate* that is equivalent to the chosen level of significance α, i.e., the GoF test wrongly rejects a distribution with α probability.

Henceforth, we consider the point BFP as part of the region, i.e., we assume to have estimated a distribution $\mathscr{G}$ that successfully passed the GoF testing procedure. This is further discussed later in the experimental evaluation (Section 3.4.4), where we obtain a BFP rejected by the GoF test during the analysis of a real dataset. According to this assumption and Definition 3.4.1, the test assigns a statistic value $D(\mathcal{X}, (\overline{\mu}, \overline{\sigma}, \overline{\xi}))$ to the BFP, that is lower than the critical value CV. In general, the BFP point does not correspond to the point with the test's minimum statistic value. In particular, we can define the following point as:

Definition 3.4.3 (Best Statistic Point, BSP)**.** Given a sequence of execution times $\mathcal{X}$, the Best Statistic Point for a statistical test with statistic $D(\cdot)$ is:

$$(\mu^*, \sigma^*, \xi^*) := \arg\min_{\mu,\sigma,\xi} D(\mathcal{X}, (\mu, \sigma, \xi))$$

■

Examples of BFP and BSP points are depicted in Figure 3.5b. According to the previous assumption, the region R has at least one point, i.e., the BFP. Thus the BSP point always exists with $D(\mathcal{X}, (\mu, \sigma, \xi)) < CV$. Before proceeding with the region's analysis, we define the following property of the statistic of a statistical test:

Definition 3.4.4 (Correct statistic)**.** Given a sample $\mathcal{X}$ of size n drawn from a distribution $\mathscr{A}$, we say that a statistic $D(\mathcal{X}, \mathscr{A})$ of a given statistical test is correct iff $D(\mathcal{X}, \mathscr{A}) \to \overline{K}$ for $n \to \infty$ with $\overline{K} \in \mathbb{R}$ and $D(\mathcal{X}, \mathscr{A}) \geq \overline{K}$ for any finite value of n.

■

To write in a less formal definition, we can say that a statistic is correct if, when applied to the exact distribution of samples and having a sample of infinite size, it provides the minimal possible value (e.g., $D = 0$ in KS test). This property and the well-known *consistent estimator* property enable the following asymptotic result:

Lemma 3.4.1. *If the estimator is* consistent *and the statistic computed by the statistical test is* correct*, then both the BFP and the BSP converge to the real unknown pWCET distribution point* $(\mu^{\circledast}, \sigma^{\circledast}, \xi^{\circledast})$*:*

$$(\overline{\mu}, \overline{\sigma}, \overline{\xi}) \to (\mu^{\circledast}, \sigma^{\circledast}, \xi^{\circledast}) \quad n \to \infty$$
$$(\mu^*, \sigma^*, \xi^*) \to (\mu^{\circledast}, \sigma^{\circledast}, \xi^{\circledast}) \quad n \to \infty$$

where n *is the size of the set* $\mathcal{X}$ *used for training or testing the pWCET distribution.*

Proof. This result is an immediate consequence of the definitions of *consistent* estimator and *correct* statistic of the test. □

This asymptotic result can be exploited to derive the following theorem:

Theorem 3.4.1. *Given a sequence of execution times $\mathcal{X}$, if the statistic of the considered statistical test is correct, the exact true distribution $P^{\circledast}$ is inside the acceptance region R:*

$$(\mu^{\circledast}, \sigma^{\circledast}, \xi^{\circledast}) \in R$$

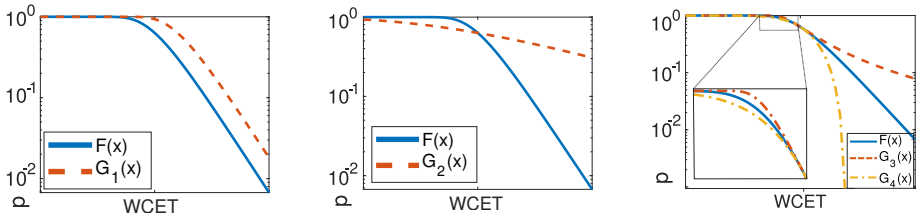
Proof. Let be n the size of the set $\mathcal{X}$ and $P_n^* \in R$ the best statistic point of Definition 3.4.3. We provide this proof by contradiction. Let assume that $P^{\circledast} \notin R$. It follows that $D(\mathcal{X}, P^{\circledast}) > CV$ and, consequently, $D(\mathcal{X}, P^{\circledast}) > D(P_n^*)$. When $n \to \infty$, $P_n^* \to P^{\circledast}$ and $D(P_n^*) \to D(P^{\circledast})$. Since the statistic is correct as Definition 3.4.4, $D(P_n^*) \to \overline{K}$, consequently $D(P^{\circledast}) = \overline{K}$. But $\forall n\ D(P_n^*) \geq \overline{K}$ and $K < D(P_n^*) < CV$, therefore $D(P^{\circledast}) < CV$ that is in contradiction with the hypothesis $P^{\circledast} \notin R$. □

From a real-time viewpoint, this theorem states that the true – but unknown – pWCET distribution is always inside the region R. This has an impact on the evaluation of the confidence of the pWCET result. In fact, each point in the region $R(\mathcal{X})$ corresponds to a pWCET distribution with parameter tuple (μ, σ, ξ). Thanks to the result of Theorem 3.4.1, we can state the following corollary:

Corollary 3.4.1. *Given a region $R(\mathcal{X})$, a violation probability $\overline{p}$, and a point $\hat{P} \in R(\mathcal{X})$ such that $\hat{P} = \arg\max_P F'_P(\overline{p})$, then either $\hat{P} = P^{\circledast}$ or the pWCET associated to $\hat{P}$ overestimates the real pWCET given by $P^{\circledast}$ at violation probability $\overline{p}$.*

In other words, given a fixed value for the violation probability $\overline{p}$, we can compute the WCET for each point of the region R. The maximum of these WCETs is either the true WCET or a safe overestimation of the WCET, at violation probability $\overline{p}$. Conversely, there is no point $P \in \mathbb{R}$ that, in general, overestimates the WCET for any probability $\overline{p}$. A possible solution to this issue is presented later in Section 3.4.2.

Exploring the Region of Acceptance. The region $R(\mathcal{X})$ describes the estimation uncertainty of the three parameters of the GEVD distribution. Its size along the three axes depends on several factors, including the distribution of the input data, the chosen test statistic, the significance level α, and the number of samples n. In particular, when increasing the sample size n, the ability of the test to detect invalid distributions improves,



(a) *Varying μ parameter* **(b)** *Varying σ parameter* **(c)** *Varying ξ parameter*

Figure 3.6: *Comparison of the complementary-cdfs $\bar{F}(x) = 1 - F(x) = P(x \geq X)$ varying the different parameters of the reference GEV distribution. (Original source: [222])*

leading to, in general, a decrease of the region size. Moreover, the three dimensions are strictly correlated. For example, experimental evidence shows that points inside the region representing a Fréchet distribution ($\xi > 0$) usually have lower values of σ than points inside the region representing a Weibull distribution ($\xi < 0$)[4]. To compare the distributions corresponding to the points inside the region, it is necessary to clearly define the order relations between two pWCETs.

The pWCET ordering via statistical dominance. In previous articles on probabilistic real-time [228, 229], the ordering relation between pWCET has been defined by using the simplest form of partial ordering between distributions:

Definition 3.4.5 (First-order stochastic dominance [208])**.** A pWCET_A *dominates* a pWCET_B iff the probability of observing a WCET larger than x is always equal or higher in pWCET_A with respect to pWCET_B, but the two distributions must not be exactly the same. In notation form:

$$\text{pWCET}_A \succ \text{pWCET}_B \leftrightarrow [\forall x : F_A(x) \leq F_B(x) \wedge \exists y : F_A(y) < F_B(y)],$$

where $F_A(x)$ and $F_B(x)$ are respectively the cdf of pWCET_A and of pWCET_B. ■

An example of first-order stochastic dominance is shown in Figure 3.6a[5]: the distribu-

[4]This is neither a formal nor a general rule, but a recurring behavior experienced by performing EVT estimations.

[5]As a reminder, the cumulative distribution function is defined as $F(x) = P(x < X)$. Instead, its complementary, $\bar{F} = 1 - F(x) = P(x \geq X)$, is depicted in the figures.

tion $\bar{G}_1(x)$ dominates the distribution $\bar{F}(x)$. However, this is a very restrictive partial ordering: it is not possible to apply it to situations like the one depicted in Figure 3.6b. Econometrics analyses frequently overcome this problem using the so-called *second-order stochastic dominance* that has been studied in [253] for extreme value distributions. Even if this dominance is widely used in financial risk analysis, it does not provide the necessary guarantees for the distribution tail. This *non-applicability* to pWCET is described in details in our paper's appendix [222]. Rather, we suggest to use a less restrictive dominance that keeps the safety of real-time requirements valid:

Definition 3.4.6 (Left tail-restricted first-order stochastic dominance [197])**.** A probabilistic-WCET pWCET_A *left dominates* a pWCET_B iff the probability of observing a WCET larger than x is always equal or higher in pWCET_A with respect to pWCET_B with $x \in [\overline{x}, +\infty)$, but the two distributions must not be exactly the same. In notation form:

$$\text{pWCET}_A \overset{\text{L}}{\succ} \text{pWCET}_B \leftrightarrow \exists \overline{x} \left[\forall x > \overline{x} : F_A(x) \leq F_B(x) \right. \\ \left. \wedge \exists y > \overline{x} : F_A(y) < F_B(y) \right].$$

Every first-order stochastic dominance is also a left tail-restricted first-order stochastic dominance:

$$\text{pWCET}_A \succ \text{pWCET}_B \implies \text{pWCET}_A \overset{\text{L}}{\succ} \text{pWCET}_B.$$

□

Although this definition is still a partial ordering, we can describe a larger set of relation, e.g. the scenario of Figure 3.6b [5]: the cdf $\bar{G}_1(x)$ left dominates the cdf $\bar{F}(x)$, i.e. fixed a WCET value, the pWCET related to $\bar{G}_1(x)$ provides an higher violation probability p for any $x > \bar{x}$ with $\bar{x} = 100$ in the depicted example.

Points dominance analysis. Having defined the orders for pWCET, we can now formalize the dominance when we move along one direction from a chosen point inside the region. In particular, Figures 3.6a, 3.6b, 3.6c depict the simplest scenarios. Using the notation inside the figures, let be $\text{pWCET}_F \sim GEV(\mu, \sigma, \xi)$ and $\text{pWCET}_G \sim GEV(\mu', \sigma', \xi')$, hence:

- If $\mu' > \mu$ and $\sigma' = \sigma, \xi' = \xi$ then $\text{pWCET}_G \succ \text{pWCET}_F$.

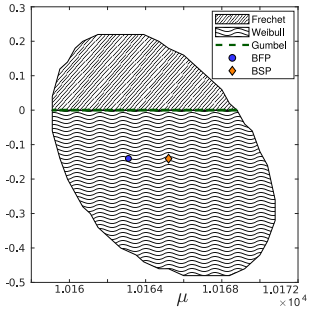


Figure 3.7: *The Region of Acceptance R generated from a Gaussian distribution sample and test CvM with the σ values collapsed. It is possible to notice that R includes all the three possible extreme value distributions. (Original source: [222])*

- If $\sigma' > \sigma$ and $\mu' = \mu, \xi' = \xi$ then $\text{pWCET}_G \overset{\text{L}}{\succ} \text{pWCET}_F$.
- If $\xi' > \xi$ and $\mu' = \mu, \sigma' = \sigma$ then $\text{pWCET}_G \succ \text{pWCET}_F$.

It is also possible to build more complex relations when two or more variables change:

- If $\xi' > \xi$ and σ' and/or μ' change in any directions then $\text{pWCET}_G \overset{\text{L}}{\succ} \text{pWCET}_F$.
- If $\sigma' > \sigma$ and $\xi' = \xi$ and μ' changes in any direction then $\text{pWCET}_G \overset{\text{L}}{\succ} \text{pWCET}_F$.

Starting from these relations and according to the previous ordering definitions, it is possible to know if, by moving from a specified point inside the region to another point inside the region, we are overestimating or underestimating the pWCET. When $\text{pWCET}_P \succ \text{pWCET}_{P'}$ the pWCET related to point P' is safely overestimated by P for any violation probability value p. Rather if $\text{pWCET}_P \overset{\text{L}}{\succ} \text{pWCET}_{P'}$ the pWCET related to point P' is safely overestimated by P for any violation probability value $p > \overline{p}$ with $\overline{p}$ that can be computed solving the equivalence equation between the iccdfs of both points. If $p < \overline{p}$, the distributions may potentially intersect in several points, making it impossible to conclude anything without further analyses.

EVT distribution classes. The Region of Acceptance may include more than one extreme value distribution classes. In order to show this, we have generated a random sample $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{50\,000}$ from a Gaussian distribution $\mathcal{N}(10000, 100)$. After applying BM, 80% of the sample is used to produce the pWCET estimation, while the remaining

20% is used to run the CvM test and build its Region of Acceptance R depicted in Figure 3.7. In this figure, we have not illustrated the scale parameter σ axis in order to clarify the variation of the ξ parameter. It can be noticed that the region includes all the three possible extreme value distributions: Fréchet ($\xi > 0$), Weibull ($\xi < 0$) and Gumbel ($\xi = 0$). Our estimator produced a Weibull distribution ($\xi < 0$), and the BSP is also in the same distribution class. From statistical theory, we know that any Gaussian is in the domain of attraction of a Gumbel; consequently, these two points, for $n \to \infty$, converge to the Gumbel line of Figure 3.7. According to the previously defined point dominance, we can define the following ordering:

$$\text{pWCET}_{\xi>0} \overset{\text{L}}{\succ} \text{pWCET}_{\xi=0} \overset{\text{L}}{\succ} \text{pWCET}_{\xi<0}$$

The fact that a region spreads over different GEVD distributions can then provide a way to increase the pWCET estimation quality. For instance, if some knowledge of the system and the task is available, e.g., it is known that the execution time must be bounded, then the exact distribution $P^{\circledast}$ cannot be in the Fréchet region. In this case, if the estimator generates a pWCET $\overline{P}$ with $\xi > 0$ and if we consider as the worst-case the point $P' \in R$ having $\xi = 0$, then P' is both safe and tighter than $\overline{P}$. Vice versa, if $\overline{P}$ has $\xi < 0$, and we know that the WCET can assume arbitrarily large values, then we must move the point $\overline{P}$ towards the Fréchet region to obtain a more pessimistic but robust estimation. The in-depth discussion and analysis of the system properties involving the MDA hypothesis are left as future work. In the experimental part of this section, a relevant result on ξ uncertainty, intrinsic in the definition of Region of Acceptance, is discussed: it is not possible to directly rely on the pWCET estimation provided by the estimator because its inaccuracies can lead to unreliable or non-tight results.

3.4.2 Bounding the distributions

In the previous paragraphs, we have seen that, in general, It does not exist one single point, i.e., one single valid pWCET distribution, that can dominate the overall region. In this section, we propose a method to overcome this problem by estimating a pWCET curve that pessimistically bounds all the possible valid distributions of the region.

The pessimistic pWCET curve. The idea behind the pessimistic bound is to obtain a robust and safe estimation of the pWCET by taking the worst-case curve generated by overlapping the cdf of all the points inside the Region of Acceptance. The requirement

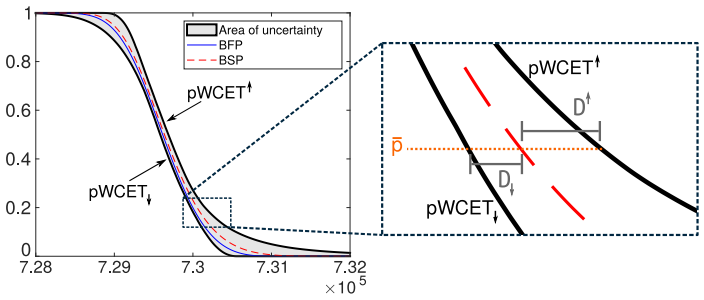


Figure 3.8: *The overlap of the cdfs of all distributions related to each point of the region of Figure 1. The upper-bound of this area is the curve pWCET$^{\uparrow}$, while the lower-bound the curve pWCET$_{\downarrow}$. (Original source: [222])*

for pessimism (safety of pWCET estimates) is illustrated in Figure 3.8, and formalized in the following definition:

Definition 3.4.7 (Pessimistic pWCET Curve)**.** The curve pWCET$^{\uparrow}$ is defined as the locus of point (WCET, p) such that $\text{WCET} \in D$ and $p = \max_{P \in R}[1 - F_P(\text{WCET})]$, where D is the domain of the worst-case execution time and F_P is the cdf corresponding to the point P in the region R. ■

As the definition clearly points out this locus of points first-order stochastic dominates all the other points $\text{pWCET}^{\uparrow} \succ \text{pWCET}_P \ \forall P \in R$, thus making the pessimistic pWCET curve a safe over-estimation of the real distribution. However, when the region is computed in a real scenario, the space of parameters cannot be explored continuously, and the set of points inside the region must be discretized. The pessimistic pWCET curve reliability also depends on the resolution selected to build the region: in the unfortunate case that $P^{\circledast}$ is in the proximity of the region boundaries, the resolution of μ, σ, ξ used to build the region may not be sufficient to include $P^{\circledast}$. More generally, to obtain a safe pessimistic pWCET curve it is sufficient to consider one more layer outside the region: $(\mu \pm \delta\mu, \sigma \pm \delta\sigma, \xi \pm \delta\xi)$ where $\delta\mu, \delta\sigma, \delta\xi$ are the parameter resolutions used in the region exploration.

The tightest pWCET curve. The same definition used for the pessimistic pWCET curve can be used to define its symmetrical tightest version, i.e., the black lower curve in Figure 3.8:

Definition 3.4.8 (Tightest pWCET Curve)**.** The curve $\text{pWCET}_{\downarrow}$ is defined as the locus of point (WCET, p) such that $\text{WCET} \in D$ and $p = \min_{P \in R}[1 - F_P(\text{WCET})]$, where D is the domain of the worst-case execution time and F_P is the cdf corresponding to the point P in the region R. ■

It is important to remark that the $\text{pWCET}_{\downarrow}$ does not necessarily represent an optimistic bound to the real pWCET distribution. This locus of points has indeed passed the GoF test, making it a valid extreme value estimation for the real pWCET. The $\text{pWCET}_{\downarrow}$ is, however, the less robust estimation in the whole set of possible distributions. We will recall and extend this concept later on in Section 3.4.3.

Area of uncertainty. The area between $\text{pWCET}_{\downarrow}$ and $\text{pWCET}^{\uparrow}$ contains all the possible pWCET distributions according to our definition of Region of Acceptance. We informally call this space *area of uncertainty*, and it is depicted as the gray area in Figure 3.8. This area is strictly correlated with the region size and parameters spread: the bigger the region R, the bigger the area of uncertainty. Since we moved from the GEVD parameters space to the ccdf space, the area of uncertainty provides a new metric to compare different possible estimations.

Definition 3.4.9 (Area of uncertainty)**.** The *area of uncertainty* is defined as the area in the ccdf-space composed of the points of all ccdf curves of all pWCET distribution belonging to the Region of Acceptance. Equivalently, it is the area between the $\text{pWCET}^{\uparrow}$ and the $\text{pWCET}_{\downarrow}$ curves. Let be $\text{pWCET}^{\uparrow}(x)$ and $\text{pWCET}_{\downarrow}(x)$ their respective curve functions $D \to [0; 1]$, then the value of this area is:

$$A := \int_0^{\infty} [\text{pWCET}^{\uparrow}(x)]dx - \int_0^{\infty} [\text{pWCET}_{\downarrow}(x)]dx.$$

□

The value of A can be easily computed numerically, and it represents a novel metric to evaluate the quality of the probabilistic analysis empirically. Large values of A suggest that our region includes large values of uncertainty not only in the parameter space but also in the pWCET space. This happens when in our region, all three GEVD models are plausible to be estimated according to the considered statistical test. Vice versa, when the value of A is small, this is a clue that at least the distribution class is correct. The value of A may also be infinite: when at least one point P with $\xi \geq 1$, the mean value of the distribution and consequently the area under the ccdf are infinite. In

this case, we can make two possible interpretations: either the analysis has been incorrectly performed, or the system behavior shows strong evidence of unbounded WCET. We could also exploit this to compare statistical tests: the area size is a direct measure of their quality because a test with a smaller area is able to detect more significant violations than another test executed with the same experimental setup but with a larger area.

3.4.3 Tightness vs pessimism trade-off

BFP vs BSP. According to Section 3.4.1, the BFP $\overline{P}$ provided by the estimator and the BSP P^* provided by the statistical test are good approximations of the unknown exact pWCET $P^{\circledast}$. In general, it is not possible to establish which of $\overline{P}$ or P^* is the closest to the exact pWCET. However, the closest point to $P^{\circledast}$ is not necessarily the *best* in a real-time context. We could, in fact, consider a different point that safely over-estimates the real distribution pWCET using one of the dominance definitions previously defined. Accordingly, the first decision criterion is to select $\overline{P}$ if $\overline{P} \succ P^*$ or vice versa. In alternative, we can consider the left tail-restricted dominance and select $\overline{P}$ if $\overline{P} \overset{\text{L}}{\succ} P^*$ or vice versa; in this case, our decision remains valid if, in the evaluation of pWCET distribution, the computed WCET at a given probability level p is higher than the value $\overline{x}$ of Definition 3.4.6. These selection criteria with the statistical dominance concepts can be applied to any other point of the region R. Unfortunately, it is not always possible to select between two points with stochastic dominance, due to its partial ordering.

The robustness ratio. To evaluate the pessimism and tightness of a given pWCET distribution belonging to a point of the region R, we propose a metric based on an empirical formula, that will be later improved.

Definition 3.4.10 (Robustness ratio)**.** Let us assume that we select a probability $\overline{p}$ and a distribution corresponding to a point $P \in R$. At this probability, we can compute three WCETs: from P, from $\text{pWCET}_{\downarrow}$, and from $\text{pWCET}^{\uparrow}$. We call: (1) $D_{\downarrow}$ the absolute value of the distance between the WCET computed in P and the WCET computed with $\text{pWCET}_{\downarrow}$; (2) $D^{\uparrow}$ the absolute value of the distance between the WCET computed in P and the WCET computed with $\text{pWCET}^{\uparrow}$ (see right-side of Figure 3.8 for clarity). We can now define the *robustness ratio* as:

$$r = \frac{D_{\downarrow} - D^{\uparrow}}{D_{\downarrow} + D^{\uparrow}}.$$

□

This ratio r is always $r \in [-1; +1]$. When $r \to -1$, the distribution of point P is near the tightest one. When $r \to +1$, the distribution of point P is instead near the pessimist one. The robustness ratio r is then a metric representing the trade-off between tightness and pessimism. The experimenter can choose among all the valid distributions based on the value of this ratio, by knowing from the results of Section 3.4.1 that the true WCET value is inside this interval.

Confidence in the pWCET analysis. If we consider the previous definition of robustness ratio, by selecting a desired value of violation probability $\overline{p}$, we can derive the WCET interval from the curves $\text{pWCET}_{\downarrow}$ and $\text{pWCET}^{\uparrow}$. This interval is written as $I_W^{\overline{p}} = [\text{WCET}_{\downarrow}; \text{WCET}^{\uparrow}]$ and, according to Figure 3.8, its size is $D_{\downarrow} + D^{\uparrow}$. Since the real distribution related to point $P^{\circledast}$ is inside the region – from Theorem 3.4.1 – and since we know that this interval represents all the pWCET distribution values – from Definitions 3.4.7 and 3.4.8 –, then the real WCET value at the given probability $\overline{p}$ is inside this interval: $\text{WCET}_{\overline{p}}^{\circledast} \in [\text{WCET}_{\downarrow}; \text{WCET}^{\uparrow}]$.

Definition 3.4.11 (Confidence of the pWCET analysis)**.** Given a probability $\overline{p}$ and a $\text{WCET} \in I_W^{\overline{p}}$, the confidence of the pWCET analysis c is defined as:

$$c = P\left[P(X > \text{WCET}) \leq \overline{p}\right].$$

□

It is important to carefully dwell on this definition. The confidence is defined as the probability that the system violation probability is underestimated. If $c = 1$, the estimated couple (p, WCET) is surely safe. If $c < 1$, there exists a certain degree of uncertainty on the safety of $(\overline{p}, \text{WCET})$. The reader should not confuse the two probabilities: $\overline{p}$ is the chosen violation probability, a run-time property of the system, i.e. the probability to experience a larger WCET than the estimated one; c is the confidence, a property of the analysis, i.e. the probability to have estimated an unsafe couple $(\overline{p}, \text{WCET})$. This confidence can be linked with the previously defined robustness ratio: if we select $\text{WCET}^{\uparrow}$, then $c = 1$ and $r = 1$, consequently $\text{WCET}^{\uparrow}$ surely upper-estimates the real WCET at probability $\overline{p}$. If the WCET value selected is not the right-most value, i.e. $r < 1$, the confidence is potentially less than the truth value: $c \leq 1$. In other words, selecting a less pessimistic, but still valid according to the chosen test, WCET may

be potentially under-estimated. Clearly, c is a non-decreasing function of WCET, i.e., higher WCETs have higher confidence. The computation of the precise c value with respect to the variation of the chosen WCET is left as future work. This computation is possible, thanks to the statistical power estimation approach of the selected GoF test presented in Section 7.2.

3.4.4 Experimental evaluation

To evaluate the benefits of the previously introduced notations and techniques, we considered four datasets representing different execution conditions and systems. The analysis of the proposed time traces showed the effectiveness of dealing with the uncertainty of the region of acceptance and related models.

We used the `chronovise` tool described in Section 7.1 to perform the MBPTA analysis on the following datasets:

D1) An industrial safety-critical application from Airbus, running on a multi-core platform. The execution time is measured on a single task execution with other tasks running in other cores and interfering on shared resources[6].

D2) The same as the previous dataset, but considering a different task of the same safety-critical real-time application.

D3) A memory-intensive task running on a multi-core T4240, stressing the data-cache with interferences on the overall cache hierarchy, shared memory and bus [77].

D4) A time trace of a GPU application running under different execution conditions, taken from the paper of Berezovskyi et al. [26].

The paper of Nolte et al. [195] proposed a classification for real-time workloads in the context of probabilistic real-time. Some proposed constraints are, however, too strict for real applications. We guaranteed A.3.1 (*Avoid usage of shared services and drives in the software architecture.*) for D1 and D2, while D3 runs on PikeOS (so we can consider valid A.3.2 that requires predictability of services) and D4 runs on CUDA, so neither A.3.1 nor A.3.2 applies for D4. Regarding the hardware states (A4 group), the cache status was disregarded (A.4.2). Finally, the tasks' execution time is not affected by the state of the environment (A.5.2).

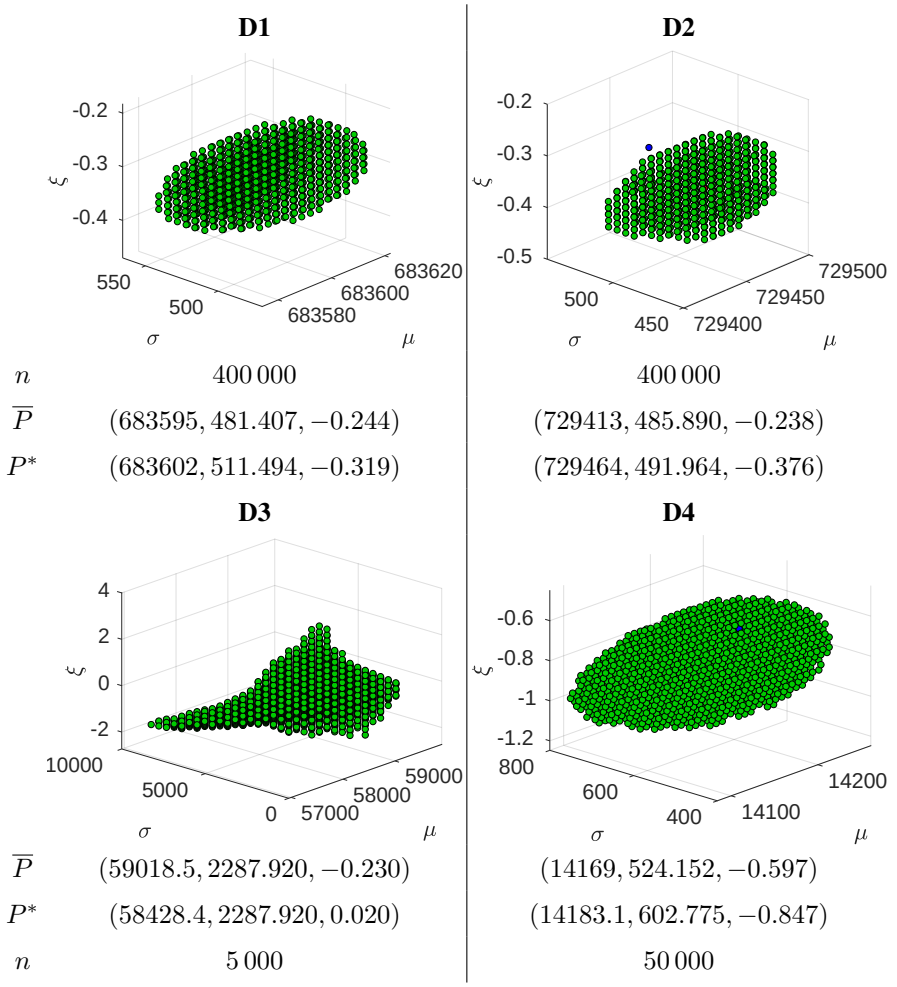
[6]No more details on the Airbus use case can be provided since it is an actual industrial application in active development.

Datasets D1 and D2 are composed of $400\,000$ time measurements, while the sample sizes of D3 and D4 are respectively $5\,000$ and $50\,000$. All the time traces in the aforementioned datasets are real measurements of the task execution time acquired with appropriate instrumentation of the applications. We verified the satisfaction of the iid hypothesis running a standard LjungBox test. Since the EVT was applicable, we could filter the data via the Block-Maxima method with a block size of $B = 20$, empirically chosen but in line with previous works [111, 167, 255].

Results – Region of Acceptance. From the sample output of the BM approach, the GEVD is fitted using the well-known Maximum Likelihood Estimator (MLE). To build the Region of Acceptance we consider as GoF test the CvM. Another possible test is KS. Both tests have *correct statistics*, i.e. they satisfy Definition 3.4.4 as proved in the appendix of paper by Reghenzani et al. [222].

To find the region R it is necessary to explore the parameter space around the estimated distribution $\bar{P}$. We have begun by uniformly exploring 40 points around each parameter, leading to a total number of $40^3 = 64\,000$ explorations. The initial interval has been set to $\pm 10\%$ of the estimated value, and it was step-by-step increased to include the whole region. The CvM test has been applied to each point, obtaining the set of accepted points, i.e., the region R, depicted in Figure 3.9. The time complexity depends on the number of points explored and how the statistic is computed for the chosen test. For CvM, the computational complexity is $O(nm)$, where n is the number of time measurements, and m is the number of explored points. Instead, for KS, the time complexity becomes $O(n^2 m)$. In this experimental evaluation, the total time required to build each region was less than 10 seconds on a standard workstation.

It is possible to notice that in the dataset D2, the estimated point is outside the region: the GEVD estimated by MLE is not a valid distribution according to the CvM test result. This is a violation of the initial assumption that the best fit estimator point $\overline{P}$ is inside the region. In this case, before beginning with the parameter space exploration, not only do we have no information on how many points should be explored, we also do not know whether the region exists or not. For example, assuming that the input time measurements are distributed according to a statistical distribution that is not in the domain of attraction of any generalized extreme value distribution – e.g., the Poisson distribution – then no point is expected to pass the test ($R = \varnothing$), whatever GEVD is estimated. In our lucky case, the region exists, and we have been able to find it because



n	$400\,000$	$400\,000$
$\overline{P}$	$(683595, 481.407, -0.244)$	$(729413, 485.890, -0.238)$
P^*	$(683602, 511.494, -0.319)$	$(729464, 491.964, -0.376)$
$\overline{P}$	$(59018.5, 2287.920, -0.230)$	$(14169, 524.152, -0.597)$
P^*	$(58428.4, 2287.920, 0.020)$	$(14183.1, 602.775, -0.847)$
n	$5\,000$	$50\,000$

Figure 3.9: *Region of Acceptance, number of samples, estimator best fit point (BFP) and best statistic point (BSP) of the datasets under analysis. (Original source: [222])*

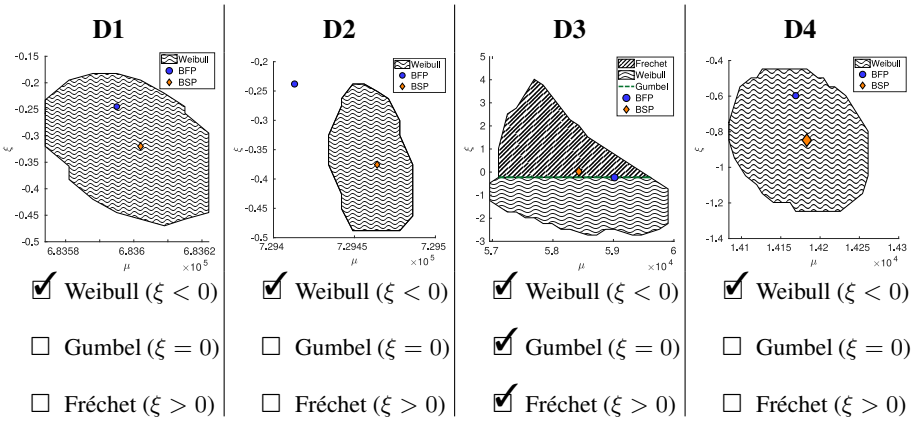


Figure 3.10: *The Regions of Acceptance for the datasets under analysis with the σ value collapsed. (Original source: [222])*

it is in the $\pm 10\%$ interval of at least one parameter. For completeness, we checked why the MLE estimator failed to obtain a valid distribution fitting the data, and we discovered the presence of a local minimum of the MLE optimization function at the estimated $\overline{P}$. One possible solution to this problem, already described in Section 2.2.2, is to initially use the Probabilistic Weighted Moment (PWM) estimator to obtain the point inside the region and improve the estimation with MLE. In the considered corner case of D2, the parameter space exploration guides us to find a set of points – the region – fulfilling the GoF test, i.e., with a valid pWCET distribution otherwise impossible to find using only the estimated point $\overline{P}$. This is another accidental advantage of using the Region of Acceptance to evaluate the pWCET output of the estimator algorithm.

Distribution shapes. Figure 3.10 shows the region collapsing the σ axis to show the spread of the ξ parameter. D1, D2, and D4 regions contain only points from Weibull distribution. In the real-time world, it means that the observed phenomenon – i.e., the execution times – has a finite maximum – i.e., a finite WCET. Instead, D3 is more problematic because it includes both Gumbel and Fréchet distribution classes. Even more, it includes Fréchet distributions with $\xi > 1$. This means that the WCET is not finite, but also its mean value is not finite, suggesting that either there is a problem in the execution time measurements, or the system has actually an unbounded WCET.

It is worth noticing the position of the BFP and the BSP in our datasets. The latter is,

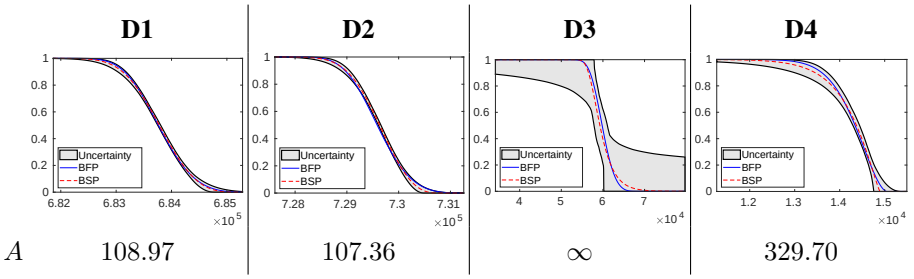


Figure 3.11: *Area of uncertainty, tightest upper-bound and pessimistic upper-bound curves, BFP and BSP distributions. (Original source: [222])*

in general, at the center of the region. The D2 case, where the estimated point is outside the region, has already been discussed in the previous Section 3.4.1. The D3 case is interesting: the estimated point represents a Weibull distribution with $\xi = -0.23$, while the point that has the best statistic is near the Gumbel line and is actually a Fréchet distribution with $\xi = 0.02$. This leads to an important conclusion: the distribution estimated by the estimator is a light-tail distribution, but the test is able to accept another distribution with even better statistic having the Fréchet tail. Consequently, we can conclude that relying on the estimator result without a sensitivity analysis on ξ may lead to unreliable or non-tight results since the real pWCET distribution can belong to another GEVD class.

Results – Uncertainty. Referring to Figure 3.11, the effect of ξ parameter spreading is clear: it is immediately visible that D3 has the largest area of uncertainty. Moreover, the absolute value of this area is infinite due to the presence of valid Fréchet distributions with $\xi \geq 1$. For the other scenarios, the area has been computed by performing the numerical integration of Definition 3.4.9. D1 and D2 have lower uncertainties compared to D4. This is the consequence of a smaller region of acceptance and, in particular, less uncertainty on ξ. We notice that there is no general rule on the domination between $\overline{P}$ and P^*. For example, in D3 $\text{pWCET}_{P^*} \overset{\text{L}}{\succ} \text{pWCET}_{\overline{P}}$ while in D4 $\text{pWCET}_{\overline{P}} \overset{\text{L}}{\succ} \text{pWCET}_{P^*}$. Instead, as expected by their definition, $\text{pWCET}^{\uparrow} \succ \text{pWCET}_{\overline{P}}$, $\text{pWCET}^{\uparrow} \succ \text{pWCET}_{P^*}$, $\text{pWCET}_{\overline{P}} \succ \text{pWCET}_{\downarrow}$, $\text{pWCET}_{P^*} \succ \text{pWCET}_{\downarrow}$ for all cases. There is only one exception that is not visible in the figure: the D3 BFP is outside the region and in this case there are some WCET values for which the $\text{pWCET}_{\downarrow}$

p	Distribution	**D1**	**D2**	**D3**	**D4**
10^{-3}	$\text{pWCET}^{\uparrow}$	685 525	731 054	$1.3 \cdot 10^{15}$	15 314
	$\overline{P}$	685 198	731 059	66 943	15 032
	$\text{pWCET}_{\downarrow}$	684 728	730 456	60 192	14 756
10^{-6}	$\text{pWCET}^{\uparrow}$	686 075	731 365	$1.5 \cdot 10^{26}$	15 368
	$\overline{P}$	684 773	731 378	68 566	15 046
	$\text{pWCET}_{\downarrow}$	685 494	730 490	60 192	14 757
10^{-9}	$\text{pWCET}^{\uparrow}$	686 231	731 425	$1.7 \cdot 10^{41}$	15 370
	$\overline{P}$	685 548	731 439	68 898	15 046
	$\text{pWCET}_{\downarrow}$	684 774	730 491	60 192	14 757
10^{-12}	$\text{pWCET}^{\uparrow}$	686 275	731 436	$2.0 \cdot 10^{50}$	15 370
	$\overline{P}$	685 558	731 451	68 966	15 046
	$\text{pWCET}_{\downarrow}$	684 774	730 491	60 192	14 757

Table 3.3: *The computed WCET from the curves of Table 3.11 at different violation probability levels. (Original source: [222])*

does not under-estimate the probability, i.e. $\text{pWCET}_{\overline{P}} \not\succ \text{pWCET}_{\downarrow}$. However, the relaxed version is still valid in this case: $\text{pWCET}_{\overline{P}} \overset{\text{L}}{\succ} \text{pWCET}_{\downarrow}$. The consequence on the WCET estimation is that the estimated distribution associated with the point $\overline{P}$ outside the region is potentially unsafe for some WCET or p values because it underestimates the curve representing the lower-bound on the distributions accepted by the GoF test.

Results – Tightness vs pessimism. Having computed $\text{pWCET}^{\uparrow}$ and $\text{pWCET}_{\downarrow}$, it is now possible to estimate the WCET according to a violation probability p. The WCET value for the different curves and some values of violation probability are presented in Table 3.3. The effect of the presence of valid points with a Fréchet distribution in D3 is evident: the WCET of $\text{pWCET}^{\uparrow}$ is clearly too large to be considered feasible in any scheduling analysis. Instead, $\overline{P}$ and $\text{pWCET}_{\downarrow}$ provide valid approximations, but with less confidence: if we select $\overline{P}$ and $\text{pWCET}_{\downarrow}$, there potentially is a non-null probability that our WCET result is unsafe according to Definition 3.4.11. The WCET values of

p	Distribution	D1	D2	D3	D4
10^{-1}	P^*	-0.093216	-0.066638	-0.994008	-0.310016
	$\overline{P}$	-0.017073	0.669008	-0.995131	-0.042482
10^{-3}	P^*	-0.255874	-0.265608	-1.000000	-0.513920
	$\overline{P}$	0.178674	1.017288	-1.000000	-0.012972
10^{-6}	P^*	-0.372082	-0.367926	-1.000000	-0.549735
	$\overline{P}$	0.107353	1.030012	-1.000000	-0.052946
10^{-9}	P^*	-0.417336	-0.395697	-1.000000	-0.551476
	$\overline{P}$	0.062515	1.031415	-1.000000	-0.055915
10^{-12}	P^*	-0.432052	-0.402085	-1.000000	-0.551556
	$\overline{P}$	0.044533	1.031659	-1.000000	-0.056071

Table 3.4: *The robustness ratios of BFP $\overline{P}$ at different violation probability levels. (Original source: [222])*

D1, D2, and D3 are distributed in a smaller interval and are all apparently feasible to be used for scheduling analysis.

To explore the differences between $\overline{P}$ and P^* we presented their robustness ratio in Table 3.4. In D1, D2, and in D4, $\overline{P}$ is more pessimist than P^* while in D3 it is the opposite. This is in line with our previous graphical result of Figure 3.11. The fact that one point is always pessimist with respect to the other point for all the considered probabilities must not be taken as a general rule: the robustness ratio is a value at a fixed probability, and it may behave differently for different values of it. The presence of valid pWCET distributions is clear in D3: both points are definitely tighter than the pWCET$^{\uparrow}$, as it is also experimentally verified in Table 3.3.

Summary of the experimental results. To summarize the experimental evaluation, we recap the major steps and the conclusions that can be drawn from the four datasets under analysis:

1. The Region of Acceptances of test CvM has been generated by exploring the space around the GEVD parameters provided by MLE estimator.

2. Even if the estimated point $\overline{P}$ for dataset D2 does not pass the GoF test, we have been able to find the Region of Acceptance in the $\pm 10\%$ of $\overline{P}$ parameter space. Despite there are no guarantees that this exploration is always successful, in our scenario, it was useful to find valid pWCET distributions that we would not have otherwise found.

3. From the analysis of shape parameter uncertainty, we noticed that one dataset (D3) includes all the three distribution types. This has a significant impact on the pWCET uncertainty: the most robust pWCET curve leads to an unrealistic WCET at small probability values.

4. If a potential reduction in the pWCET confidence is acceptable, the issue of the previous point can be easily solved by selecting another point inside the region according to its robustness ratio value and test's statistical power.

5. The estimated distribution $\overline{P}$ is not necessarily the best one, neither in terms of safety nor in terms of tightness. The same is valid for the BSP P^*. Careful evaluation must be performed by exploiting one of the decision-making tools provided.

3.4.5 Lesson learned

Any distribution estimation routine suffers from estimation errors caused by the necessarily finite number of input samples. In probabilistic WCET analyses, the safety of the results depends on several conditions imposed by the EVT conditions. Even considering all the open challenges on these conditions solved, the estimated pWCET distribution is still affected by uncertainty. This section discussed this problem by providing a set of mathematical tools to deal with the parameter uncertainty, intending to be a step towards a more reliable pWCET estimation. In particular, the *region of acceptance* has been defined on the GEVD parameter space. By exploring this region, it is possible to move the estimated pWCET distribution to more reliable or to tighter distributions. The advantages, on both the safety and the tightness of the pWCET distributions, have been shown by performing the analysis on real-time traces of different nature, including real industrial datasets.

CHAPTER 4

Applicability to Real Systems

Having discussed about the estimation uncertainty, we move in this chapter to study the applicability of probabilistic real-time techniques to real systems, discussing about the i.i.d. hypothesis and the representativity of MBPTA techniques.

4.1 The Probabilistic Predictability Index

The *reject*/*not-reject* result and the absolute value of the statistics of the three previously described tests for the i.i.d. hypothesis (Section 3.3.1) do not provide clear and straightforward information on the time predictability of the system and the applicability of probabilistic real-time. In this regard, we introduce a more meaningful unified index to provide a quantitative value that expresses the fulfillment of the statistical hypotheses of a given time trace. We refer to this index as *Probabilistic Predictability*

Index (PPI). The PPI has been designed by merging the three tests while maintaining their statistical properties to be able to use the novel index as a hypothesis test as well. The PPI is defined over the continuous range $(0; 1)$. For PPI values near 0, the time samples present strong evidence that the time series is not analyzable because it violates EVT hypotheses. Vice versa, for PPI values near 1, the time series presents good properties and adherence to EVT hypotheses. The time series should be rejected if the PPI is lower than the predefined critical value CV_{PPI}, maintaining the original statistical tests' significance. In particular, if $PPI > CV_{PPI}$, then the hypotheses are true, and the pWCET can be safely estimated; if $PPI \leq CV_{PPI}$, then at least one hypothesis is violated and any pWCET estimation would lead to unreliable results. The PPI is obtained with a set of transformations that maintain the statistical foundations of the original hypothesis tests. The capability of rejecting or not rejecting the null hypothesis is unchanged, as well as the statistical power. This is extremely important in the probabilistic real-time context, due to the critical aspect of the pWCET reliability.

4.1.1 Index formulation

The statistics of the tests – described in Appendix B – have the following ranges[1]:

$$D_{KPSS} \in (0; +\infty) \quad D_{BDS} \in (-\infty; +\infty) \quad D_{R/S} \in (0; +\infty)$$

To level off these statistics, we need to define a common domain $D = (0; 1)$, to which PPI domain belongs. By taking into account the described desired meaning for PPI and the statistic formulas, we have to find the following functions:

$$f_{KPSS} : (0; +\infty) \to D$$

$$f_{BDS} : (-\infty; +\infty) \to D$$

$$f_{R/S} : (0; +\infty) \to D$$

under the following constraints:

$$\lim_{x \to +\infty} f_{KPSS} = 0 \quad \lim_{x \to 0} f_{KPSS} = 1$$

$$\lim_{x \to \pm\infty} f_{BDS} = 0 \quad \lim_{x \to 0} f_{BDS} = 1$$

$$\lim_{x \to +\infty} f_{R/S} = 0 \quad \lim_{x \to 0} f_{R/S} = 1$$

[1] We omit the $(\{X_i\})$ parameter of statistics D_i for brevity.

Moreover, the rejection property of the test statistics against the critical value must be maintained. Let $CV_{KPSS}, CV_{BDS}, CV_{R/S}$ be the critical values of the respective tests, each null hypothesis has to be rejected if

$$|D_i| > CV_i \quad \forall i \in \{\text{KPSS}, \text{BDS}, \text{R/S}\} \tag{4.1}$$

For this reason, the Equation (4.1) must hold whatever transformation we apply. In order to satisfy this requirement, the $f_{KPSS}, f_{BDS}, f_{R/S}$ transformations must be continuous, positive, and monotonic functions. The following functions satisfy the aforementioned properties:

$$\begin{aligned} f_{KPSS}(x) &= e^{-K_{KPSS} \cdot x} \\ f_{BDS}(x) &= e^{-K_{BDS} \cdot |x|} \\ f_{R/S}(x) &= e^{-K_{R/S} \cdot x} \end{aligned} \tag{4.2}$$

Hence, we have to select $K_{KPSS}, K_{BDS}, K_{R/S}$ in order to be compliant with the previous constraints and to get the same critical value for each test. We assign an empirical value to $K_{KPSS} = \frac{1}{4}$, then the critical value for KPSS test can be computed $C^*_{KPSS} = e^{-\frac{1}{4} CV_{KPSS}}$. Since the critical value should be the same for the other two tests, their constants are computed as follows:

$$\begin{aligned} k_{BDS} &= -\frac{\log C^*_{KPSS}}{|CV_{BDS}|} \\ k_{R/S} &= -\frac{\log C^*_{KPSS}}{CV_{R/S}} \end{aligned} \tag{4.3}$$

eventually obtaining:

$$CV_{PPI} := C^*_i = f_i(CV_i) \quad \forall i \in \{\text{KPSS}, \text{BDS}, \text{R/S}\}$$

Assigning different values to k_{KPSS} would produce different statistic PPI values. However, it does not change its statistical meaning because the critical values would change consistently. Choosing a higher value of k_{KPSS} shifts the PPI to produce values towards 0, vice versa, a lower value of k_{KPSS} shifts the PPI to produce values towards 1. We selected the value of $k_{KPSS} = \frac{1}{4}$ such that the obtained CV_{PPI} is 0.89 for $\alpha = 0.05$, i.e. about of 10% of fraction of PPI values $(0.9 - 1.0)$ are dedicated to values representing valid hypothesis, while the remaining 90% fraction (values $0.0 - 0.9$) can represent the violation degree of the hypotheses. The experimenter can change k_{KPSS}

$f_{KPSS}(D_{KPSS})$	$f_{BDS}(D_{BDS})$	$f_{R/S}(D_{R/S})$	PPI	
0.96	0.91	0.92	0.92	✓
0.50	0.91	0.92	0.50	✗
0.50	0.91	0.70	0.425	✗

Table 4.1: *Example of PPI values for three cases of statistical tests. The critical value is $CV_{PPI} = 0.89$. (Original Source: [220])*

at will, without losing statistical properties, but modifying the human perception of the index value. In Table 4.1, we can see an example of PPI value computation for three cases of statistical tests statistics.

Having uniformed the three statistics in the $(0, 1)$ range with the same critical values, it is possible to merge them into a unique index by applying the following conservative approach:

- if *all* test statistics are higher than their respective critical values, PPI must be higher than PPI critical value;
- if *any* of the three test statistics is lower than its critical value, PPI must be lower than the PPI critical value;
- if *more than one* test statistics are lower than their respective critical values, PPI must be lower than the minimum of the three statistics.

This approach ensures that the statistical test meaning is not changed: we can compare PPI with the critical value to assess all three hypotheses, assuming that if any EVT hypothesis is violated, the test rejects the null hypothesis.

To ensure this behavior of the test, we apply the following merging transformation:

$$PPI := \begin{cases} \min\limits_{\forall i} f_i(D_i) \cdot \prod\limits_{i \in v^*} [1 - (CV_{PPI} - f_i(D_i))] & v \neq \varnothing \\ \dfrac{1}{3} \sum\limits_{\forall i} f_i(D_i) & v = \varnothing \end{cases} \tag{4.4}$$

where v is the violation set, i.e. $v = \{i | f_i(D_i) < CV_{PPI}\}$, and v^* is the violation set without the minimum, i.e. $v^* = \{v \setminus \arg\min_{\forall i} f_i(D_i)\}$. If no violation occurs in the three tests, the result is the arithmetic mean of the three values, which is greater than

CV_{PPI}. Otherwise, the PPI is equal to the minimum statistic potentially multiplied by other statistics that violate CV_{PPI}. This leads to a PPI value lower than CV_{PPI}, guaranteeing the statistical hypothesis testing property. The arithmetic mean uses the same *weight* for the three tests to compute the PPI because each test verifies one different hypothesis.

To summarize, the PPI value can be computed by using Equation (4.4) and compared with the critical value CV_{PPI}. In probabilistic real-time scenarios, if $PPI < CV_{PPI}$ the following null hypothesis (H_0) has to be rejected in favor of the alternative one (H_1):

H_0 : the time trace $\mathcal{X}$ verifies the EVT hypotheses

H_1 : $\mathcal{X}$ violates at least one of the EVT hypotheses

4.1.2 Experimental evidence

This subsection presents the experimental evaluation of the chosen statistical tests and the proposed index PPI. The datasets have been analyzed thanks to the open-source software `chronovise` (see Section 7.1). This software has been updated to include the algorithm computing the PPI index. The algorithm is also available separately as MATLAB script [219]. The correct assessment of the EVT hypotheses with a statistical testing inference (e.g., PPI) requires to acquire several time traces of execution time, run the statistical tests for each time trace, and, finally, look at the results. In particular, the ratio of rejection/non-rejection of the null hypothesis provides the deduction of the statistical property searched for. A single time traces meeting the EVT hypotheses can not provide any insights about the statistical process generating it. In fact, even if a single time trace may pass the checks, this must not be interpreted as the system to be compliant with the EVT hypotheses. In fact, the hypotheses can be considered fulfilled when the rejection/non-rejection ratio settles around the significance level for the test, or the α_{global} in case of multiple testing as described in Section 3.3.4. The expectation is to get high rejection rates for time traces that do not satisfy the hypotheses. On the other hand, if the source distribution of the samples verifies the EVT hypotheses, then the rejection rate should settle around the significance value α.

Time trace sources. For characterizing the properties of the proposed test, we used both synthetic time samples and real benchmark executions. The first class of time

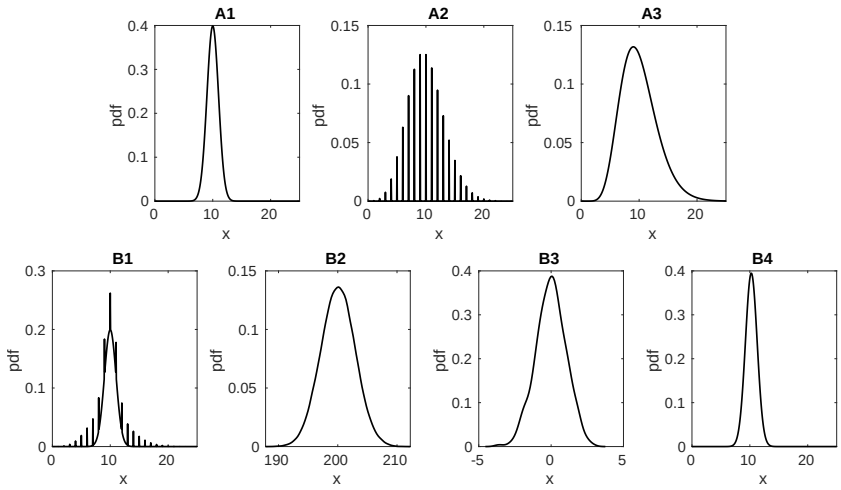


Figure 4.1: *The Probability Distribution Functions of the synthetic benchmarks considered. (Original Source: [220])*

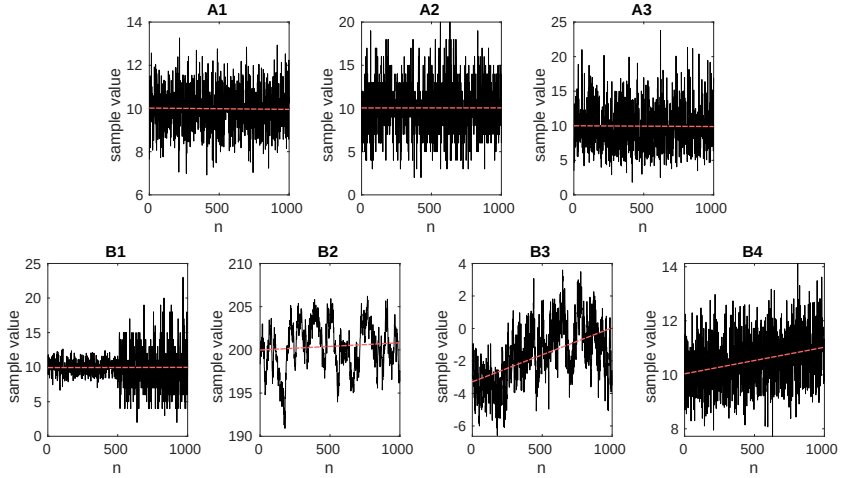


Figure 4.2: *Plots of the realization of 1000 random variables with the distributions of the synthetic benchmarks considered. The red dashed line shows the long-term trend, computed as a linear interpolation of all the points. (Original Source: [220])*

traces has been designed to stress the detection capability of each statistical test by using synthetic distributions with well-known statistical properties. The real benchmarks are instead executed on different hardware platforms, with known real-time capabilities, to evaluate the probabilistic predictability of the target system. Without losing generality, we evaluated the tests with a level of significance $\alpha = 0.05$. This means that we expected, for each test, a type I error (i.e. false-positive rate) of 5%. In our scenario, this is a conservative error: each test excludes 5% of the times a dataset that is actually valid for EVT estimation. The overall type I error can be computed using Equation (3.7), obtaining 14% ($\alpha_{global} = 0.14$).

Synthetic sources. Let $\mathcal{X}_{a:b}$ be an ordered subset of the full time trace $\mathcal{X}_{1:n}$. For synthetic and controlled time traces, we used both i.i.d. and non i.i.d. sources. For the former, we selected the following EVT-compliant distributions that are expected to pass the statistical tests:

A1 $\mathcal{X}_{1:n} \sim \mathcal{N}(10, 1)$: Gaussian (normal).

A2 $\mathcal{X}_{1:n} \sim \mathcal{P}(10)$: Poisson.

A3 $\mathcal{X}_{1:n} \sim \Gamma(10, 1)$: Gamma.

Then, we tested four non-compliant distributions that are expected to violate at least one of the i.i.d. sub-hypotheses:

B1 $\mathcal{X}_{1:\frac{n}{2}} \sim \mathcal{N}(10, 1); \mathcal{X}_{\frac{n}{2}+1:n} \sim \mathcal{P}(1)$: a normally distributed time trace for the first half part and then a Poisson distribution; it represents a sequence of independent but not identically distributed samples.

B2 $\mathcal{X}_{1:n} \sim AR(2)$: an auto-regressive model of order 2, with constant 10 and auto-regressive coefficients $(0.7, 0.25)$. This class represents a short-range dependent time source.

B3 $\mathcal{X}_{1:n} \sim ARFIMA(\frac{1}{2}, 0, 0, 0, \frac{1}{4})$: an auto-regressive fractionally integrated moving average model with AR, MA, and I coefficients zero, constant $\frac{1}{2}$ and $d = \frac{1}{4}$. This class represents a time source with a long memory.

B4 $\mathcal{X}_{1:n} = \{\forall i \in [1; n] | X_i \sim N(10 + 0.001 \cdot i, 1)\}$: non identically distributed samples with long-range dependence, but short-range independent.

We drew a total of 1 000 000 samples for each distribution, and then we split into groups of size 1 000 for a total of 1 000 evaluations. The pdfs of these distributions have been plotted in Figure 4.1, as well as an example of traces in Figure 4.2. It is possible to note that A1, A2, A3 appear as random, B1 is composed of two modes, B2 presents a clear short-range dependence, while B3 and B4 have long-term trends.

Real sources. Concerning the experimental evaluation on real platforms, we run four state-of-the-art benchmarks of the WCET Mälardalen suite [118]: `sqrt`, `minver`, `fdct`, `complex`. We implemented each benchmark onto five different platforms, whose well-known architecture characteristics introduce different degrees of unpredictability:

R1 `PIC`: a PIC18F45K50 microcontroller without operating system.

R2 `STM`: time-deterministic platform with a L1D and L1I cache: STM32F7 board programmed bare-metal without operating system.

R3 `MIO`: time-deterministic platform with a real-time operating system: the STM32F4 with Miosix operating system[2].

R4 `ODR`: embedded development-board unpredictable platform: multi-core Odroid XU-3 with a Linux OS (vanilla kernel).

R5 `INT`: a desktop system, completely unpredictable platform: multi-core Intel i7 with a Linux OS (vanilla kernel).

R1 is a simple processor, time-deterministic, and constant instruction timing. R2 and R3 are also time-deterministic platforms, with no features that can affect the execution time predictability, except the L1 caches of R2, which introduce a timing dependence among the benchmark execution. R4 is, instead, an embedded development board with several advanced features, making the execution time unpredictable. R5 is even more unpredictable because it is a general-purpose machine and, consequently, contains several unpredictable hardware features, such as SMIs.

The benchmarks have been slightly modified to add: (1) a PRNG for input data generation (except for `complex` where the input is constant), (2) an external loop to run the benchmark multiple times, (3) a toggling mechanism for a GPIO to signal the start and

[2] `http://miosix.org/`

	E[PPI]	VAR[PPI]	Rej.$_{PPI}$	Rej.$_{KPSS}$	Rej.$_{BDS}$	Rej.$_{R/S}$
A1	0.9374	1e-3	13.9%	6.8%	5.5%	4.5%
A2	0.9388	1e-3	12.3%	5.3%	4.9%	4.3%
A3	0.9393	1e-3	11.4%	4.9%	5.3%	3.4%
B1	0.6302	2e-3	100%	4.7%	100%	6.9%
B2	0.0058	1e-5	100%	100%	100%	100%
B3	0.5228	3e-2	100%	83.1%	99.2%	98%
B4	0.1319	3e-3	100%	100%	5.5%	100%

Table 4.2: *Tests rejection results of synthetic time traces analysis. (Original Source: [220])*

stop of benchmark execution. To maintain consistency among all platforms, the PRNG has been initialized with the same seed. In such a way, each platform generates the same sequence of pseudo-random inputs to the benchmarks. The time measurements have been acquired by measuring the GPIO interval between the rising edge (start of the computation) and the falling edge (end of the computation), using a commercial logical analyzer with a $10ns$ resolution. Each benchmark then has been executed $100\,000$ times by using time series of size $1\,000$ for statistical testing, for a total number of 100 estimations for each benchmark.

Results. The results on time traces from synthetic sources are shown in Table 4.2. For i.i.d. datasets (A1-A3), it is possible to notice a rejection rate based on evaluations of single tests around 5%, which actually matches the chosen significance level α. The rejection rate of the composed index PPI is slightly below 14%, that is the significance level value computed by using Equation (3.7). This value represents the false-positive error rate, i.e., the percentage of time series discarded even if generated by compliant sources.

Regarding the results of time traces that do not satisfy at least one EVT condition (B1-B4), we can notice the PPI rejection rate is always 100%. We can observe that the power of BDS is high for B1-B3, but it is not for B4, where KPPS and R/S are able to reject the hypothesis. On the contrary, for B1, only BDS appears to be sufficiently powerful. Moreover, it is worth highlighting that B1 is a non-identically distributed

		Reject_{PPI}	Reject_{KPSS}	Reject_{BDS}	$\text{Reject}_{R/S}$
sqrt	R1	23%	7%	15%	6%
	R2	24%	2%	20%	3%
	R3	29%	6%	23%	7%
	R4	14%	1%	13%	1%
	R5	34%	20%	12%	21%
minver	R1	16%	6%	6%	6%
	R2	15%	9%	5%	6%
	R3	17%	10%	7%	8%
	R4	44%	1%	44%	1%
	R5	78%	61%	31%	66%
fdct	R1	8%	2%	5%	2%
	R2	20%	4%	15%	4%
	R3	100%	99%	100%	99%
	R4	62%	16%	48%	15%
	R5	81%	67%	45%	71%
complex	R1	79%	19%	72%	16%
	R2	56%	5%	52%	3%
	R3	100%	0%	100%	0%
	R4	92%	5%	92%	1%
	R5	100%	60%	95%	71%

Table 4.3: *Tests rejections of R1-R5 real hardware time traces. (Original Source: [220])*

time series, but KPSS is not able to detect it, while BDS provides for it. This is due to the lack of statistical power of KPSS in case of weak stationary, but not strict stationary time series [168].

Table 4.3 and Figure 4.3 show the results when time traces are generated by executing benchmark applications on the aforementioned platforms. It is possible to observe the expected trend of generating *less-compliant* time traces, with the increasing of the hardware complexity. The traces generated by the `complex` benchmark are hardly analyzable for all platforms due to the lack of variability. This contrasts with the common logic behind the WCET analysis for which a more stable timing is preferable. The

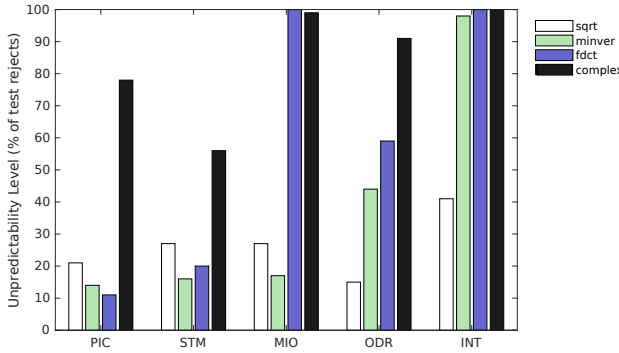


Figure 4.3: *The reject rates of the four benchmarks executed onto the described platforms. (Original Source: [220])*

statistical tests described and the EVT, instead, generally require a minimal degree of variability, as also shown by Lima et al. [166]. The benchmark `complex` lacks variability as it is the only benchmark one – out of the four benchmarks – that performs simple computation on the same input data for each iteration. For example, in the PIC microcontroller case (R1), the variability of time measurements of `complex` benchmark is due only to the measurement errors of the instrumentation: the input is constant for this benchmark, and the PIC microcontroller has a constant instruction timing architecture, making the actual execution time constant. The same measurement errors affect the other experiments, but the variability from software and hardware sources dominates the measurement errors.

For all the other benchmarks, the simple PIC microcontroller generates deterministic time traces that lead to a low rejection rate, close to the significance level, i.e., the false positive rate. The MIO and STM platforms present higher values of rejection, caused by the presence of the operating system and cache memories, respectively. As expected, and with the only exception of `sqrt` case, the probabilistic theory cannot be used for the Odroid, and least of all, the Intel CPU based machine. The only unexpected outlier is the `fdct` benchmark on the Miosix board. Here the rejection rate is 100% without an apparent reason. By observing the time traces, we hypothesize that the instruction prefetcher, the board is equipped with, causes large recurrent variations compared to the intrinsic variability of the `fdct` benchmark, that triggers the detec-

		ACET	WCOT	DI	Rej_{PPI}	Rej_{KPSS}	Rej_{BDS}	$\text{Rej}_{R/S}$
sqrt	P1	408 262 ns	938 217 ns	7 508.3	89%	39%	67%	48%
	P2	390 508 ns	426 728 ns	77.5	88%	0%	88%	0%
	P3	388 413 ns	413 123 ns	65.6	6%	1%	5%	1%
minver	P1	485 854 ns	1 114 823 ns	6 938.3	94%	48%	72%	47%
	P2	466 040 ns	1 914 380 ns	128.5	89%	0%	89%	0%
	P3	464 227 ns	542 293 ns	69.4	43%	0%	43%	0%
fdct	P1	470 208 ns	789 290 ns	6 482.7	92%	39%	67%	48%
	P2	450 845 ns	478 049 ns	46.8	100%	0%	100%	0%
	P3	450 561 ns	487 578 ns	35.5	100%	0%	100%	0%
complex	P1	429 737 ns	764 678 ns	7 206.3	87%	20%	71%	22%
	P2	410 894 ns	447 453 ns	63.9	97%	0%	97%	0%
	P3	410 585 ns	443 200 ns	46.7	100%	13%	100%	33%

Table 4.4: *Linux PREEMPT_RT result for the four WCET benchmarks considered. (Original Source: [220])*

tion of a short-range dependence. However, this conclusion requires a more in-depth analysis of the specific system that falls outside this section's scope.

Generally, the results shown in Table 4.3 highlight an important fact: the single PPI hypothesis test result depends on both the system and the considered workload. Moreover, the result is a random variable, that consequently requires to be sampled several times to assess the capability of analyzing a system by using EVT. Again, the PPI hypothesis test result (or the result of any single test) is a property of the time traces provided, not of the system generating such traces.

The Linux kernel has been built as a general operating system and thus not appropriate for real-time computing since its main performance goal can be considered maximizing the average throughput. For this reason, the PREEMPT_RT patch has been developed since the first decade of the 2000s, to improve the time predictability of the Linux kernel. A comprehensive survey of scientific works related to PREEMPT_RT is available [214] and later presented in Section 5.1.

In this experimental evaluation, we run the same benchmarks used for R1-R5 but on an Odroid H2, a quad-core x86-64 platform based on COTS components and consequently subject to unpredictable latencies. The goal is to verify if the introduction of PREEMPT_RT improves the predictability of execution times and which effects PREEMPT_RT has on the probabilistic theory's applicability. We exploit the PPI index previously defined in three scenarios:

P1 On a plain vanilla Linux and the task having no special configuration (like the previous R5 case).

P2 On a plain vanilla Linux but the task configured with real-time priority and with core pinning.

P3 On a PREEMPT_RT kernel and the task configured with real-time priority and a core pinning.

The task under analysis runs together with contenders on other cores that cause interferences at both architecture and operating system levels. The contention has been generated thanks to the *stress-ng* tool[3]. Like the previous tests, each benchmark has been executed $100\,000$ times by using time series of size $1\,000$ for statistical testing, for a total number of 100 estimations for each benchmark and scenario.

The results are shown in Table 4.4. As common in PREEMPT_RT works, we also computed the Average-Case Execution Time (ACET), the Worst-Case Observed Time (WCOT), the Dispersion Index (DI), the PPI value, and its components. DI is computed as follows: $\frac{\mu}{\sigma}$, where μ is the mean value of the time trace and σ the standard deviation. Looking at the traditional ACET, WCOT, and DI values, it is possible to notice that applying the correct task real-time priority configuration and the $PREEMPT\_RT$ patch are essential to obtain low variability. In particular, the dispersion index of P2 is at least one order of magnitude lower than P1, and P3 is slightly better than P2. This means that the execution times vary in a smaller interval, making it more predictable. It is possible to notice that the WCOT is much lower in P3 than the other cases. Setting real-time priority is not sufficient to reduce sporadic high-level latencies, as proved by the large WCOT of `minver` benchmark, even larger than the P1 scenario. To reduce these latencies is crucial for the inclusion of the PREEMPT_RT patch in Linux kernel. Focusing on the PPI index, there is no direct link between DI, WCOT, or ACET concerning the satisfaction of EVT hypotheses. In fact, even if PREEMPT_RT seems to improve the satisfaction of hypothesis for `sqrt` and `minver`, this is not true for `fdct` and `complex`. Comparing the PPI index with its critical value, it is possible to conclude that only the scenario sqrt running on PREEMPT_RT satisfies the EVT hypothesis and makes possible the estimation of a correct distribution.

This example of PREEMPT_RT shows that it is not sufficient to improve the average-case, worst-case, nor the predictability of the platform to improve the satisfaction of the

[3] `http://kernel.ubuntu.com/~cking/stress-ng/`

EVT hypotheses. A possible future work may be investigating why PREEMPT_RT is not able to improve the `fdct` and `complex` cases and which are the internal kernel mechanisms and/or architecture components that prevent this.

4.1.3 Lesson learned

Considering the synthetic time traces, the PPI resulted in being very effective in detecting the violation of i.i.d. property. Indeed, the non-compliant time traces have been rejected with a 100% rate, while the rejection rate of the compliant ones settled in the range 11.4% – 13.9%. This range represents the false-positive rate of the test, which is, however, lower than the expected theoretical value of 14%.

For real-time traces from real benchmark applications, we can notice that the PPI rejection rate trend is coherent with the index meaning. In fact, for predictable platforms, we experienced low rejection rates, while for complex platforms, this is very close to 100%, as expected.

Finally, we computed the PPI index for the same benchmarks running on a Linux embedded platform to observe if the PREEMPT_RT patch can make the system EVT-compliant. The results suggest that it is not possible to claim a priori satisfiability of the hypotheses, nor we can generally conclude that improving the usual average-case or worst-case metrics improves the satisfiability as well. The inability to obtain a priori such information makes the PPI analysis essential.

4.2 Representativity

We already pointed out the importance of the representative hypothesis of probabilistic real-time. This section provides the reader with a clear and formal definition of this hypothesis, often loosely defined in previous works. In the last subsections, we instead describe the barriers in reaching the representativity of the time measurements and which can be the future research directions.

4.2.1 The WCET problem as a dynamic system

For the purposes of this section, we model the WCET in a control-theory fashion with a dynamic system. We restrict our analysis to a single periodic task τ running on a single-core processor. We will show that even with this simplistic assumption, the rep-

resentativity problem of probabilistic approaches is already very challenging. Without loss of generality, we consider task τ as a task (periodic or aperiodic) activated multiple times and spawning a sequence of jobs. Its execution time of the k-th job of the task τ can be modeled like the following time-invariant discrete system:

$$\begin{cases} x(k+1) & = f(x(k), i(k)) \\ T(k) & = g(x(k), i(k)) \\ o(k) & = h(x(k), i(k)) \end{cases} \tag{4.5}$$

where $x(k) \in X$ is the machine state, $i(k) \in I$ is the input, $o(k) \in O$ the output, and $T(k)$ is the execution time of the k-th job. The machine state $x(k)$ includes all values intentionally or unintentionally stored by the task across job executions – e.g., register values, the memory content, the available data in the cache. When a job is activated, it begins the execution with the machine in the state $x(k)$, it processes the input $i(k)$ to produce an output $o(k)$, and finally, it leaves the system in a new state $x(k+1)$. The system's logical correctness depends on the value of the output $o(k)$, while the temporal correctness depends on $T(k)$. Because we can model any traditional computing system as a Turing machine, the function f, g, and h are deterministic functions.

The WCET of this model is the maximum value of $T(k)$ for any valid state and input. However, in a computing platform, it is usually difficult to *set* a state, and $x(k)$ is usually explored by providing an input sequence $i(1), i(2), ..., i(n)$. Consequently, we write the WCET as the following maximization problem:

$$C = \max_{\forall \{i(1), i(2), \ldots, i(n)\} \in I^n} T(k) \tag{4.6}$$

This formulation can be simplified if:

- The state $x(k)$ is constant, e.g., a stateless program running on a simple microcontroller without any cache or other memory items influencing the execution time. In this casem, $C = \max_{\forall i(k)} T(k)$ because the WCET is just the maximum for any input (and not for any sequence of inputs).

- The program does not have any input, or it has an input not affecting state and execution time. In such scenario, the WCET depends only on the state and is the maximum execution time measured for any possible state of the machine: $C = \max_{\forall x \in X} T(k)$.

4.2.2 Formal definitions of representativity

The *representativity* is a term that is sometimes defined in a vague manner. This section provides formal definitions of deterministic and stochastic representativity for, respectively, the MBDTA and MBPTA approaches, in addition to linking the concepts to practical situations.

Deterministic representativity. In the MBDTA case, the representativity is simply the fact that we choose a sequence of inputs able to capture the real WCET. This requirement is formally expressed similar to Eq. (4.6) as follows:

$$C^* = \max_{\forall\{i(1),i(2),\ldots,i(n)\}\in I^{*n}} T(k)$$

where $I^* \subset I$ is a set of input sequences. The input subset I^* is representative if and only if $C^* = C$. Unfortunately, it is usually very difficult to find a subset I^* so that it is possible to prove that the previous condition holds. If the system is considered in a black-box approach (i.e., with no or limited knowledge of f and g), the only way to find I^* is to exclude input sequences not compliant with input specifications, and thus that cannot occur at run-time. For example, if input specifications tell us that two identical inputs cannot occur in a row, we can exclude all the sequences having $i(k) = i(k-1)$, for any k. In a black-box approach, it is essential to know the value of n: how many inputs are necessary before the state returns to an already visited value. In real systems, it seems quite difficult to get such information. In some industrial environments, systems are periodically rebooted at a given rate, making, in this way, the estimation of n possible. However, this rate is usually measured in terms of days or weeks, making in this way I^{*n} still too large. We have an alternative to the black-box approach when we know something about the functions f and/or g in our model of Eq. (4.5). For example, a common approach to reducing the input space is to guarantee that at least all the possible paths in the control flow graph are covered [61], and if, in the considered architecture, the execution time of the single instruction is independent of the data, we can then reduce the number of input values to try. However, it does not provide any information on the representativity concerning the state, so we still need some information on g, or we have to be able to observe $x(k)$ and detect when we reach an already visited state. Both solutions present several issues to be effectively used in a real industrial scenario. The black-box approach is limited to very simple architectures and programs (e.g., stateless with a small input space). Instead, knowing something

about f and/or g may help, but the required effort to obtain software and hardware characteristics is similar to the static analysis, jeopardizing the advantages of MBDTA.

Stochastic representativity. The representativity, in the case of the pWCET, refers to estimating a distribution that upper-bounds the real distribution. Let $\mathscr{G}$ be the probability distribution computed by the MBPTA algorithm from the sequence of observed execution times $\{T(1), T(2), ..., T(n)\}$ generated by the input sequence $\{i(1), i(2), ..., i(n)\}$. This input sequence is said to be *stochastic representative* if $\forall \bar{C}$:

$$\bar{F}_{\mathscr{G}}(\bar{C}) \geq p^* \tag{4.7}$$

where p^* is the *real* (and unknown) probability of observing an execution time larger than $\bar{C}$, and $\bar{F}(\bar{C}) = P(X \geq \bar{C})$ is the ccdf function of the distribution $\mathscr{G}$. An alternative and equivalent formulation is possible by inverting the ccdf function and obtaining $\forall p$ it must be true that $\bar{F}_{\mathscr{G}}^{-1}(p) \geq C^*$, where C^* is the *real* (and unknown) execution time at the given probability. It is possible to restrict this condition for only some values of $\bar{C}$ or p, instead of having the $\forall$ operator; in such case, the representativity is valid only for some values of the WCET or probability of violation.

In MBPTA, the estimation of the distribution is, in general, subject to estimation errors because of the finiteness of the number of samples $\{T(1), T(2), ..., T(n)\}$. For example, by considering the EVT method, the distribution estimator (e.g., Maximum Likelihood Estimation) is imperfect or the statistical tests used to check the EVT hypotheses have a non-zero error rate. For this reason, we define a new concept of representativity by adding to the previous formula a *confidence term* φ. The input sequence is said to be *weak stochastic representative* if $\forall \bar{C}$:

$$P(\bar{F}_{\mathscr{G}}(\bar{C}) \geq p^*) \geq \varphi \tag{4.8}$$

The outer probability refers to committing an error during the offline MBPTA analysis (with probability $1 - \varphi$), while the inner ccdf $\bar{F}$ refers to a run-time property of the system, i.e., that the estimated probability of exceeding the execution time $\bar{C}$ is not higher than the real one p^*.

4.2.3 Representativity and WCET safety

The concept of representativity of the WCET is strongly connected to the safety of the WCET estimation. The MBDTA case is simple to describe: any measurement campaign

must trigger and observe the worst-case scenario, i.e., the system must be executed at least one time with the state and input values $(x(k), i(k))$ that maximize $T(k)$. If we can guarantee that this scenario is observed, then the schedulability analysis can be safely performed because $C^* = C$. Otherwise, we have to consider that the worst-case observed time is potentially smaller than the real WCET by a quantity $\delta C = C - C^*$. If this quantity δC can be (over-)estimated, then it can be added to the maximum observed time, and the schedulability analysis can again be safely performed. However, the value of δC is usually unknown and hard to estimate. Expert-provided values or empirical upper-bounded are sometimes used, but they cannot formally guarantee a safe overestimation of this δC, hindering the certification requirements for safety-critical systems.

Regarding the MBPTA case, the pWCET is safe if the probability to observe a higher execution time than $\bar{C}$ is correctly computed. The value $\bar{C}$ is selected from the distribution function and used in the schedulability analysis. The violation probability can be considered as a probability of our task to fail in the safety analysis of critical systems. For example, an extra event for software failure, with the annotated pWCET probability, can be added to the Fault Tree Analysis. Using reliability analysis for logical correctness software faults is definitely not a new concept [162] and can also be used for temporal correctness violations. However, it is mandatory to formally prove that the computed probability of failure – i.e., the probability of observing an execution time larger than $\bar{C}$ – matches the real one or over-estimates it. Conversely, the safety analysis potentially incorrect, hindering the certification process.

4.2.4 How do you know what you know?

The title of this section refers to the traditional question of epistemology. This philosophical question has a crucial impact on the evaluation of the safety of the measurement-based approach and, in particular, on MBPTA approaches. This section aims to discuss the possibility of having any formal proof for pWCET safety or, alternatively, whether it is possible to make statistically sound conclusions on the safety of pWCET.

Epistemic vs aleatory uncertainty. The epistemic uncertainty is the uncertainty related to a model or, in general, the lack of knowledge of a phenomenon. Instead, the aleatory uncertainty is the intrinsic natural variability of a phenomenon. Let us explain these two concepts by considering a simple experiment: “Roll a dice”. If we perform

this experiment several times, we obtain a random sequence of numbers: let us assume we observed $\{2, 1, 6, 4, 4\}$. The reader can apply any statistical estimation technique on this sequence, but he or she must be aware that it contains both the epistemic and the aleatory uncertainty. The epistemic uncertainty is because we did not tell the reader how many faces the dice has, or if it is balanced or weighed. On the other hand, the aleatory uncertainty is the natural uncertainty of the throw, expressed as a probabilistic distribution. The epistemic uncertainty can be reduced by increasing the number of measurements and improving our knowledge of the model. However, increasing the number of measurements does not reduce the aleatory uncertainty, but it makes our estimation more precise. If we perfectly know all the dice parameters (number of faces, weights, etc.), the aleatory uncertainty still exists. However, the most problematic is the epistemic one. A good example is from seismology: while statistical techniques can be used to deal with the aleatory uncertainty, scientists cannot forecast earthquakes location and time because of the presence of epistemic uncertainty, due to the lack of data and the high complexity of the model of such phenomena.

Is representativity achievable in a real system? Probabilistic real-time strategies model the execution time variability with a probabilistic distribution computed using a statistical technique (e.g., a simple Empirical-CDF or the EVT). Provided that the proper statistical hypotheses are respected, these techniques find an estimation of the pWCET distribution parameters. However, it is easy to show that such techniques cannot deal with epistemic uncertainty. Let us provide a counterexample by considering the program with the trivial control flow graph of Figure 4.4. If we consider this system as a black-box, we observe for all the inputs, with the exception of $i(k) = 1234$, a normal distribution $N(10, 1)$ of execution times. If we never observe $i(k) = 1234$, we have no idea what is going on in the other branch, and whatever statistical technique we select, it cannot predict something we do not observe. In the case depicted in the figure, the other branch has a larger execution time (on average), but it could be that the branch we do not observe contains an infinite loop. Such a silly situation would be undetected by an MBPTA tool, and its probability is non-estimable. Someone may argue that at least all the branches should be analyzed. However, the same figure can be rewritten with states $x(k)$ instead of inputs $i(k)$ and, consequently, also states must be considered. If all states and inputs have to be observed, then the problem is going back to deterministic representativity (with all the disadvantages described in the previous section),

and statistical tools are not even needed if there is no aleatory uncertainty. Researchers should not confuse the representativity with the *independent and identically distributed (i.i.d.)* hypothesis common of many statistical techniques, including Empirical-CDF and EVT. In fact, in the example of the figure, assuming $i(k)$ to be i.i.d. or constant, $T(k)$ is also i.i.d., but with a mixture distribution. In addition, if $i(k)$ can not be assumed i.i.d., there is epistemic uncertainty also on how $i(k)$ varies. It should also be noted that the other condition of EVT (the maximum domain of attraction hypothesis) is usually not satisfied with mixture distributions, while they are satisfied, e.g., by using an Empirical-CDF. If the case $i(k) = 1234$ is never observed, then $T(k)$ appears to a normal distribution, i.e., it would pass the i.i.d. statistical test. Consequently, even if they are somehow linked concepts, the i.i.d. condition does not necessarily imply representativity. To better exemplify this concept, let us use the previous example of "Roll a dice" and use the output face value as our execution time. Let us also assume that the dice has six faces with values $\{1, 2, 3, 4, 5, 100\}$ with non-uniform weights such that the face 100 appears with a very low probability, for instance 10^{-6} per throw. We observe the outputs of the rolls several times without having any knowledge of the last sentence, as we will do in a black-box measurement-based approach of a computing system. In other words, we have a non-null epistemic uncertainty. If we never observe 100, the output values appear i.i.d. for any test we use, and we will probably estimate a WCET 5 or slightly higher. The measurements not only *appear* to be i.i.d., they actually *are* i.i.d.: there is no dependence in the throws, and the probability distribution does not change. This example clearly shows that i.i.d. is a necessary condition for many estimation methods, but it is not sufficient for achieving representativity.

We discussed epistemic uncertainty, but a question on aleatory uncertainty arises: what is aleatory uncertainty in today's computing systems? Basically nothing. By excluding unexpected hardware effects, such as clock skew or random faults, current industrial systems are deterministic Turing machines; consequently, the only uncertainty regarding WCET is epistemic. This uncertainty is because, in a measurement-based approach, we do not know the perfect model of the architecture (in particular f and g of Equation (4.5)), the model of the algorithm, and their interactions. The processor clock may be affected by aleatory uncertainty, but it is usually irrelevant for the WCET computation. DRAM controllers often require a variable time to access a memory cell, with a timing behavior appearing stochastic, also due to DRAM refreshing. However, this is again epistemic uncertainty and not aleatory. Some researchers added a pseudo-aleatory

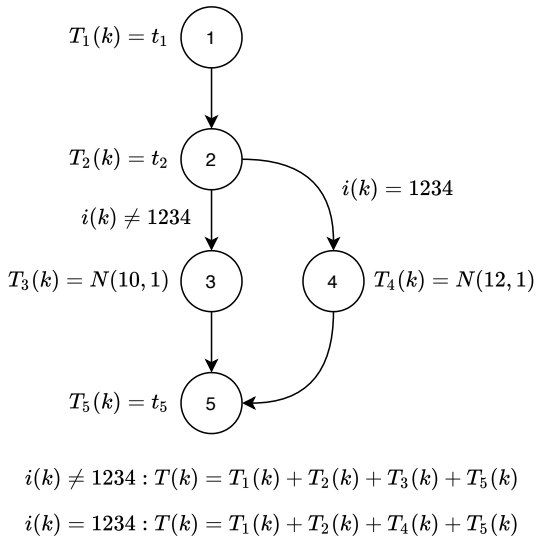


Figure 4.4: *An example of control-flow graph with two different timing behaviors depending on the input value.*

behavior to the caches [209]. This was done to improve the satisfaction of the i.i.d. hypothesis for EVT. This technique may have a positive impact also for representativity, as discussed in the next paragraphs.

Estimating the uncertainty. In statistics, many recent works, not related to probabilistic real-time computing, discussed the problem of estimating epistemic uncertainty in extreme environments. For example, in 2018, Jones et al. [140] applied Bayesian uncertainty analysis to quantify the epistemic uncertainty of extreme distributions in an ocean engineering problem. However, the quantification of epistemic uncertainty is still an estimation and, by definition, subject to uncertainty. Other methods to estimate the uncertainty in the estimation of the epistemic uncertainty are required, but they would be, in turn, subject to uncertainty. This nested structure of uncertainties is called the *errors on errors* concept. In practical applications, someone may argue that such a recursion can stop at some level. However, taking any estimation as an objective truth is potentially dangerous, especially when looking at the distribution tails, e.g. our pWCET, as recently showed by Taleb et al. [245]: Ignoring errors on er-

rors introduce a significant risk that the tail of the distribution is under-estimated. The uncertainty estimation problem does not only affect representativity. In probabilistic real-time literature, statistical tests are the traditional way to verify the EVT hypotheses. The results of statistical tests are often considered valid without an investigation of the inevitably present uncertainties, as showed in Section 3.3.3. Even when uncertainties have been estimated, *errors on errors* concept appears, hindering the safety of the pWCET approaches. The estimation of epistemic uncertainty as a possible solution to representativity has never been considered in probabilistic real-time. However, this is still a controversial topic in theoretical statistics, and some researchers consider it more a philosophical problem rather than a possible realistic estimation. For this reason, nowadays, it does not appear a realistic solution to integrate the estimation of epistemic uncertainty in the analyses of safety-critical systems. Alternative solutions to uncertainty estimation have to be then considered.

Possible future solutions. A possible solution is to eliminate or hide the epistemic uncertainty by design. If this goal is achieved, then probabilistic real-time could be used for the certification of safety-critical systems because we can easily estimate the aleatory uncertainty and safely predict the p^* of Equation (4.7) and Equation (4.8). One possible solution to achieve representativity is to make the functions f and g stochastic rather than deterministic. This is partially achieved by randomized caches, at least for the state function f. We propose a set of possible solutions promising for future research studies to zero epistemic uncertainty of the states and the inputs in order to achieve representativity. To remove the state epistemic uncertainties, possible solutions are:

1. Using stateless architectures: This is the trivial solution, and many processors are already available, such as microcontrollers or simple System-on-Chip. Clearly, these architectures are not very computational powerful, and they are usually easy to be analyzed with non-probabilistic static tools, vanishing the advantages of probabilistic real-time.

2. Making all components of the system time-randomized, i.e., in such a way they produce a correct $o(k)$ but they expose $T(k)$ as a random process with aleatory uncertainty only (possibly with a single non-mixture distribution, so that EVT can be applied and stochastic representativity possible). To expose $T(k)$ as a random process, $x(k)$ should be randomized. The randomization should be done

	Solution	Pro	Cons
State	Stateless systems	• Simple • Many commercial products already available	• Low computational capabilities • Easy to analyze with SDTA
	Randomized architectures	• Very effective • Implementations for caches already exist	• Hard to randomize everything • Loss of performance
	Unconventional architectures	• Potential, but uncertain, benefits for MBTPA	• Early stage of development
Input	Random inputs	• No need to implement anything	• Limited use in real applications
	Randomized algorithms	• Effective, make the input problem irrelevant • Some strategies already available from security field	• Not easy to be implemented on large software • Loss of performance

Table 4.5: *Possible solutions to remove or hide epistemic uncertainty.*

by using True (hardware) Random Number Generators (TRNGs), to avoid introducing epistemic uncertainty that would otherwise be present in Pseudo-Random Number Generators (PRNGs).

3. Adopting unconventional computing architecture paradigms – for example stochastic computing [99], hyper-dimensional computing [143], or residue number computing [188] – may change how we conceive the computation and, therefore, the WCET estimation. Stochastic computing was proposed from the early years of computer science, but it remained substantially an academic exercise due to the numerous complexity and poor performance issues. However, in the last years, it is reborn thanks to its use in machine learning environments. It is difficult to establish if this architecture, the others previously mentioned, or the future ones may reduce or not the epistemic uncertainty on execution time compared to today's architectures. It should be noted that the previously mentioned architectures also have an effect on $o(k)$, possibly introducing issues on functional correctness, which must not be neglected.

Regarding the input epistemic uncertainties, possible solutions to remove or hide them are:

4. Using probabilistic real-time for systems having only physical quantities as inputs (such as sensor readings), that can be assumed to belong to a statistical distribution. In this case, even if f and g are deterministic, like in traditional computing systems, they are applied to random variables (inputs $i(k)$), and then the processes $x(k)$ and $T(k)$ are also random variables. The inputs $i(k)$ have only

aleatory uncertainty and no epistemic uncertainty, leaving the epistemic uncertainty only on the models of f and g.

5. Using algorithms that are time-independent with respect to states and/or inputs. For example, cryptographic algorithms are often randomized in time to avoid side-channel attacks that would infer the key by analyzing the execution time. In this case, the goal is to hide the epistemic uncertainty (that would expose the cryptographic key) and show as much as possible the aleatory variability to make the attack harder. Similar algorithms could be exploited in probabilistic real-time to obtain the pWCET, even if we have to take into account that current time-randomized cryptographic algorithms make considerably worse both the average-case and the worst-case performance.

To achieve representativity, a combination of the previous solutions can be used. Whether these solutions can become actually feasible and advantageous is an open question, as also summarized in Table 4.5. For example, *can solution 2) be implemented without significantly reducing the actual computational capabilities of the platform?* The same question applies to case 5). Solutions 1) and 4) are, instead, very restrictive. Finally, solution 3) still requires to demonstrate the abilities and convenience of unconventional architectures and may require several years to have a paradigm shift in industrial products.

Another option exists, but it raises practical and ethical issues. In many other engineering fields, epistemic uncertainty is present and cannot be removed. Engineering margins are applied based on expert-provided thresholds and past experience. This also happens for safety-critical systems: for example, in nuclear power plants, statistical techniques (e.g., EVT) are applied to determine the probability of extreme weather events and to, consequently, drive the design choices; however, the models of weather phenomena are still known to be subject to a large and non-quantifiable epistemic uncertainty. Similarly, the embedded systems community may consider the option to do not perfectly model the WCET, because finding an accurate model for modern architectures is already too complicated.

CHAPTER 5

Exploitation for Non-Traditional Real-Time Systems

The previous techniques of probabilistic real-time can be exploited for other systems rather than computing the WCET for traditional hard real-time setups. The first section describes how the PREEMPT_RT patch adds real-time capabilities to Linux systems. Probabilistic methods are probably the only WCET estimation methods that can be used for a complex kernel like Linux, where the formal proofs of static analyses would not be affordable. Then, the second section explains how probabilistic information can be exploited to improve the energy-optimization of mixed-criticality systems. Finally, the application of probabilistic-WCET estimation to HPC clusters is presented, another scenario where the traditional WCET estimation tools would fail due to excessive complexity.

5.1 Linux PREEMPT_RT

The moving towards COTS multi-core platforms described in Section 1.2 makes the porting of time-unpredictable *General-Purpose Operating Systems (GPOSs)* to real-time environments an interesting alternative. Among the GPOSs, the Linux operating system has evident advantages: tons of scientific and industrial research solutions are available for a wide number of problems. Thanks to the high number of available libraries, often open-source, the adoption of Linux as an operating system can considerably reduce costs and development effort [210]. Complex applications, like Graphical User Interfaces or machine learning algorithms, require an immense work to be re-implemented on custom real-time operating systems, due to the common absence of state-of-the-art libraries.

The advantages and the limitations of using Linux for non-desktop markets are presented in this section, specifically focusing on real-time applications. The source code accessibility and portability, the years of industrial and scientific research, and the amount of already available algorithms and libraries have made Linux a strong alternative to commercial and specialized approaches, also in embedded environments [124]. As evidence of this trend, real-time Linux has been considered by both the European Space Agency (ESA) and the National Aeronautics and Space Administration (NASA) for space and ground applications, including mission-critical software [37, 160, 244].

5.1.1 The real-time patch

The PREEMPT_RT patch of the Linux kernel is actually composed of a set of patches developed by a group of kernel developers. The project was started by Ingo Molnár, and the first release was based on kernel version 2.6.11 (March, 2005). This project aims to trade the throughput of the system with low latencies and predictability, while maintaining the single-kernel approach, to allow the developers to easily write (user-space) real-time applications. Among the changes introduced, it is worth mentioning the introduction of two additional preemption levels (a fourth and fifth one): *Preemptible Kernel* or *Basic RT* (`PREEMPT_RTB`) and *Fully Preemptible Kernel* (`PREEMPT_RT_FULL`). More in detail, the latter allows the real-time tasks to preempt the kernel everywhere, even in critical sections. However, some regions can still be made non-preemptible, like the top-half of interrupt handlers and the regions protected by *raw spinlocks*. The distinction of which spinlock is raw and which is not is a prerogative of the PREEMPT_RT

patch.

During the development of the PREEMPT_RT patch, besides the new features introduced, several positive side effects were observed. Systems with this patch applied are more sensitive to bugs due to latency constraints [105], allowing the kernel developers to easily discover the bottlenecks of the kernel itself. Moreover, the patch introduced several scheduler improvements and analysis tools in the kernel mainline.

PREEMPT_RT and co-kernels. The most evident difference between the state-of-the-art approaches based on co-kernel mechanisms – such as RTAI [176], Xenomai [101], and RTLinux [259] – and the PREEMPT_RT patch is the absence of a second kernel dedicated to the management of the real-time applications. This makes the implementation of real-time processes in user-space similar to non real-time ones, apart from having a special scheduling class and priority. Generally speaking, the PREEMPT_RT software development is entirely different from co-kernels: an application can be executed in "real-time mode" without being rewritten. The programming model of co-kernel approaches uses specialized system calls, which usually provide the functions to deal with cyclic period tasks, typical of real-time applications. Even considering the advantages of specialized programming models, they remove one of the most important advantages of using Linux in real-time environments: exploiting the already available huge set of drivers and libraries to speed up the application development process. It is worth mentioning that even if these drivers and libraries are available in Linux, additional effort may be required to make them real-time compliant. Overall, the PREEMPT_RT patch allows the developers to operate in a real Linux environment in which they can easily reuse most of the existent libraries and tools, including all the set of functions specified by the POSIX standard.

5.1.2 Literature review

Over the years, several contributions in this sense have been integrated into the Linux kernel mainline to support real-time. The remarkable ones have been summarized in the timeline shown in Figure 5.1. A detailed description of the improvements and further insights of the early RT patches can be found in the articles by Rostedt and Hart [224], Henriques [125], and Edge [80].

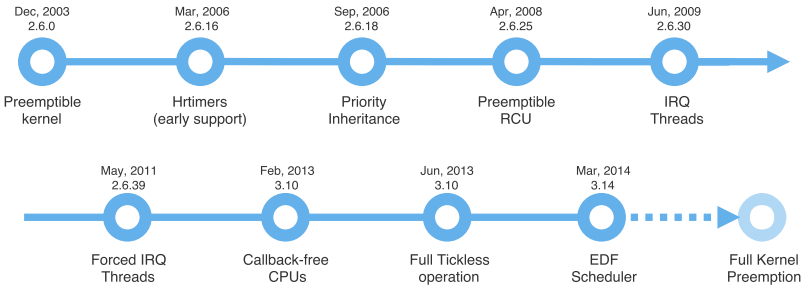


Figure 5.1: *Timeline of merged real-time features in the mainline Linux kernel, most of them coming from PREEMPT_RT patch. (Original Source: [214])*

Interrupts. Linux may incur in unbounded or poorly predictable latencies when dealing with interrupts by the following causes [144, 6, 82]:

- The presence of the well-known priority inversion problem.
- The impossibility to run the worker threads at a proper frequency. The delayed execution of kernel utility threads may impact on the overall performance of the system and also on the real-time tasks. A couple of examples are the `pdflush` daemon (the thread controlling the writeback mechanism from memory to the disk) and the `kswapd` (swap management).
- The problem of assigning priority to interrupt threads in charge of managing the peripherals shared by tasks of different priority. An interrupt thread providing services to a real-time task should have equal or higher priority with respect to the latter in order to serve it with sufficiently low latency. However, when a low priority task shares a peripheral with a high priority task, it may indirectly preempt the high priority one with the execution of the interrupt thread.

Some solutions to these problems were proposed. Priority inheritance, to solve the first problem, was implemented in the early PREEMPT_RT patch [56], since priority inversion is able to cause unbounded and unpredictable latencies to real-time tasks [144, 224]. More recently, alternative solutions include priority ceiling protocol [44], migratory priority inheritance [38] and Probabilistic-Write/Copy-Select mechanism [263].

RCU. The *Read-Copy-Update (RCU)* synchronization mechanism [184] was introduced in Linux kernel version 2.6 and then ported by Desnoyers et al. [71] to user-

space. RCU had an impressive success thanks to the performance improvements in synchronization. As a consequence, RCU is nowadays extensively used in all kernel subsystems [185]. The basic idea is to replace the standard read-write locks into RCU primitives to prevent read-side locks from blocking and to maintain multiple versions of the object updated by the writers. The RCU technique is particularly useful in scenarios featuring multiple readers and one writer. Guniguntala et al. [113] showed how RCU offers deterministic overhead, and it can be effectively used for real-time applications in PREEMPT_RT based systems. We omit a detailed description of the mechanism since it is accurately described in the cited papers. The RCU subsystem is still an active topic in research. Recent works include the modeling and verification of the RCU kernel code [147, 164], high-level user-space abstractions [71, 192] and proposal for possible variants of RCU [183, 13, 204].

Tickless. In the Linux kernel, a periodic timer triggers the main event interrupt routine at a constant rate depending on the HZ value. It updates the internal data structures, such as the scheduler counters for the process time slices. The advent of high-resolution timers and the RCU improvements previously described make the inhibition of the periodic timer event possible. This feature is known as *Full Tickless Operation*, and, typically, it is not enabled by default. The first steps towards this achievement were described in the article of Siddha et al. [237] and some years later on `lwn.net` by Corbet [57]. This introduction had extreme importance for the following reasons: (1) it reduces the power consumption of battery-powered devices when inactive, and (2) it decreases the extra latencies due to the rate of preemptions triggered by the periodic timer.

Schedulers. Since kernel version 2.6.23 (October 2007), the *Completely Fair Scheduler (CFS)* [198] is the Linux default scheduler. The system administrator can assign to each process a scheduling policy and a priority. In Linux, the scheduling policies traditionally follow the POSIX standard [108]:

- `SCHED_OTHER` (sometimes called `SCHED_NORMAL`): default time-sharing scheduling for best-effort workload.
- `SCHED_RR`: a priority-based Round Robin algorithm.
- `SCHED_FIFO`: First-In First-Out policy.

The real-time classes were tested by Sousa et al. [243]. The authors showed that the best-effort workload of the system negatively impacted the performance of the real-time workload. They tested a multi-core system with over 50% of load, and they showed that the scheduler was not able to guarantee deadlines. Subsequently, they proposed the scheduling policy *Real-time TAsk Splitting (ReTAS)* inspired by the state-of-the-art *Notional Processor Scheduling-F (NPS-F)* policy [35]. However, this policy was never receipted by kernel developers. In 2010 the `SCHED_DEADLINE` policy was presented [87, 175] and subsequently integrated in the kernel mainline (version 3.14, March 2014). This scheduler is based on the *Earliest Deadline First (EDF)* and the *Constant Bandwidth Server (CBS)* algorithms.

Memory allocators. Traditionally, for most hard real-time systems, the memory allocation is statically performed. This because dynamic memory allocators routines may have unpredictable behaviors, making the estimation of the WCET upper-bound hard [206]. For example, the regulation DO-178B of avionics software [225] excludes the possibility to use dynamic memory allocation for safety-critical systems. The more recent version DO-178C [226] introduced general guidelines on dynamic memory management verification. The major problems affecting memory allocators with respect to real-time constraints are memory fragmentation and exhaustion, dangling references, and possible unbounded worst-case (re)allocation time [206, 235], that are clearly unwanted effects. In less critical real-time systems, the operating system usually provides a memory allocator to help the application developers. A noticeable example is the Real-Time Java Specification that provides a sophisticated hierarchy of memory allocators dedicated to real-time software [129]. Another example – implemented in several operating systems, including RTLinux and PREEMPT_RT – is the *Two-Level Segregate Fit (TLSF)* proposed by Masmano et al. [178]. The key point of this algorithm is the capability to allocate memory with $O(1)$ time complexity. It is often called the *O(1) memory allocator* in Linux communities. However, a previous work exists, in fact, the first $O(1)$ allocator developed was the Half-fit algorithm [196]. A constant time complexity entails a bounded worst-case execution time of the memory allocator independently of applications and data. Regarding general-purpose memory allocators, Linux has different techniques at the kernel-level and user-level. At kernel side, the *slab* allocator [171] guarantees the absence of memory fragmentation, while in user-space several implementations are available, mostly having linear complexity [92].

Performance and real-time capabilities. The common trend is not to consider Linux PREEMPT_RT as a hard real-time compliant solution. The work of Brown and Martin [40] and the technical report of Chalas [50] led to the conclusion that real-time Linux systems can be considered *95% hard real-time*. These studies compared the latencies resulting from the execution of specific applications under certain system configurations. The general idea of a *95% hard real-time* system is that deadline miss are tolerated if they occur with a probability lower than 5%. However the timespan to be considered is not specified, making this classification not really convincing. A similar classification, but with 99% constraint, was proposed for the Ach IPC RT Library [63]. All the works in literature agree on the benefits introduced by the PREEMPT_RT patch in terms of improvements of the scheduling capabilities for real-time tasks. Betz et al. [28] argued that these improvements become evident when the best-effort workload exceeds the 75%. However, this result is in contrast with the previously cited paper [243], in which the authors noticed a real-time performance degradation with a system workload greater than 50%. Unfortunately, the two papers make use of very different kernel versions – respectively 2.6.25 and 3.2.11 – and both lack of details on the actual system configuration, making the experiments hard to be replicated and compared. Neglecting the workload effects, in first approximation, we can state that interrupt and scheduling latencies reported are in the order of micro-seconds (μs) with worst cases smaller than $50\mu s$ [15, 83, 49, 89, 120].

Possible uses of the PREEMPT_RT. The PREEMPT_RT patch is promising when adopted in industrial environment only if the hard real-time requirements are not dictated by safety-critical constraints. Linux could not in fact provide any formal guarantee of WCET using classical static analysis tools. This is due to both kernel complexity and intrinsic unpredictability of modern architectures. Besides embedded scenarios, other scenarios can take advantage of the latency reduction and increased predictability provided by PREEMPT_RT, e.g. multimedia, HPC, and network applications. Time predictability has gained interest in HPC community, as explained later in Section 5.3. For academic and research purposes, PREEMPT_RT is a potential candidate for the development of both applications and test-benches. In the first case, any type of application can be implemented and tested in a real-time environment with less effort rather than using complex RTOS. In test-bench cases, the experimenters can use Linux to test the performance of scheduling algorithms, IPC calls and any other operating system

mechanisms. It is important that researchers correctly configure the Linux system in order to have realistic latencies. This activity has been sometimes neglected and may lead to erroneous or unreliable conclusions.

PPI analysis. Previously, in Section 4.1.2, we reported an analysis based on the PPI of a Linux board equipped with PREEMPT_RT. The patch improves the satisfaction of the conditions only for certain benchmarks, but further investigations are needed to assess the possible use of PREEMPT_RT for probabilistic real-time systems, and, in particular, for MBPTA.

5.2 Mixed-Criticality Energy Optimization with Probabilistic Information

Designing a real-time system with the assumption that a task executes up to its WCET may lead to system over-provisioning, low utilization, high costs, and excessive power/energy consumption [180]. This contrasts with the requirement of improving performance while maintaining the non-functional property at an acceptable level.

5.2.1 Introduction to mixed-criticality systems

To efficiently utilize the non-negligible gap between the WCET and the actual execution time, and to minimize energy consumption, resource over-provisioning, and cost, the *Mixed-Criticality (MC)* framework [249] received attention from both the scientific and industrial community. In an MC setup, different software components with different criticality levels are integrated into a common platform. To each task, a criticality level is assigned together with multiple execution time thresholds (at different certification/pessimism levels), as described by Vestal's seminal paper [249]. This value is inspired by the industry standards for safety-critical systems: for instance, the DO-178C standard [226] sets 5 levels of criticality: $\{A, B, C, D, E\}$, where A is the criticality level referring to functions that may cause catastrophic failures, while at level E, to functions that do not affect safety. In real-time computing, this value is interpreted as the level of assurance of the WCET. To illustrate this concept, let us consider a dual-criticality system, where the tasks are classified in `LO`-criticality and `HI`-criticality. The tasks belonging to the latter category have two values for the WCET: one value is pessimistic, but safe, while the other value is computed with an analysis that provides a

lower level of assurance, and hence less pessimistic. In this case, the less pessimistic value may not correctly (over-)estimate the real WCET. Consequently, the execution time of a task may overrun this value. When this condition happens, i.e., the overrun, we say that a *mode switch* occurs.

Existing MC task-set scheduling strategies aimed at: (i) correctly scheduling all the tasks when the system exhibits less pessimistic behaviors (in this case, the system is said to be in LO-criticality mode), and, (ii) correctly scheduling the more important (HI-criticality) tasks under more pessimistic behaviors, while no scheduling guarantee is given to the less important (LO-criticality) tasks (in this case, the system is said to be in HI-criticality mode). The system starts running in LO-criticality mode by scheduling the tasks under optimistic assumptions. When a HI-criticality task violates its assigned execution time threshold, the system switches to the HI-criticality mode, and it drops all the LO-criticality tasks to guarantee the deadlines of HI-criticality tasks. However, discarding (or providing degraded services) to the LO-criticality tasks may result in severe performance loss and it violates the task independence requirements for safety-critical systems [85].

A recent work by Bhuiyan et al. [31] handled the *precise scheduling* of MC systems, where full service is provided to all tasks under both the pessimistic and optimistic assumptions. They incorporated the *Dynamic Voltage and Frequency Scaling (DVFS)* scheme to the precise scheduling strategy and derived the energy-aware CPU speed to execute in normal mode. However, the optimized energy consumption is in accordance to the *worst-case* behaviors under the normal mode. The energy consumption should be optimized for the average/expected scenarios instead, and the existing MC task model (with multiple WCETs) does not provide sufficient information to perform such optimization.

5.2.2 Related works

To date, a significant amount of works studied the energy minimization scheduling technique considering both the parallel and sequential real-time tasks, in a non-MC platform, few to mention [53, 34, 200, 116, 30, 117, 33]. On the other hand, extensive research has been done on real-time scheduling of the MC task model considering both the sequential and parallel workload model (e.g., [18, 21, 81, 115, 32, 163]), without considering the energy-awareness.

The majority of the above-mentioned works considered the EDF-VD scheduling

policy, while Lee et al. [157] proposed the MC-Fluid model. In this model, each task receives a share (dependant on its criticality level) of the available resources. They also proposed MCDP-Fair, an implementable (on a real hardware platform) variant of the MC-Fluid. Considering the dual-criticality platform, Baruah et al. [22] proposed MCF, which provided an improved speedup bound of no greater than 1.33. However, all these above-mentioned models received criticism (from Ernst et al. [85] and Esper et al. [86]) of being impractical and unable to maintain the run-time robustness. Also, a vast majority of the existing works have another limitation, upon a system mode-switch, no service guarantee is provided to any of the LO-criticality tasks [19, 20, 114, 41, 85]. Some recent works [23, 169, 158] proposed the imprecise mixed-criticality (IMC) model, that provides degraded service to the LO-criticality tasks even after a mode-switch.

Little work has been done [130, 194, 216] that considered both energy awareness and MC scheduling. All these papers assumed that all LO-criticality tasks are dropped after a mode-switch. The recent work in [31] proposed the technique to provide a full-service guarantee to all LO-criticality tasks (even after a mode switch) and most related to this work. However, it is also based on the pessimistic assumption that all the tasks will execute up to their WCET (at the corresponding system criticality level) and did not consider the probabilistic information to derive the LO-criticality execution time thresholds.

5.2.3 The proposed solution

In the work presented in this section, we propose to exploit the probabilistic information not to directly estimate the WCET for scheduling analysis purposes, but to use them for the optimization of the system energy consumption. Differently from previous probabilistic approaches, the focus of this work is the satisfaction of real-time requirements with a *deterministic* approach and on top of that, exploiting the probabilistic information to minimize the expected energy consumption of the system.

Existing works aimed at minimizing energy consumption at LO-criticality mode, but considered a pessimistic assumption that all the tasks execute up to their WCET, at their respective criticality levels [31]. Since a task rarely needs to execute up to its WCET, we integrate the probabilistic based prediction strategy and the DVFS scheme to the precise scheduling of MC tasks. The main innovative concepts are:

1. An energy-aware scheduling strategy that selects the proper (optimistic) WCETs

for tasks and the processor speeds under both (LO- and HI-criticality) modes to minimize the overall average energy consumption. This optimization is performed via a novel probabilistic analysis of the execution time, coupled with a dedicated response time analysis which guarantees the timing correctness of all tasks under both the pessimistic and optimistic assumptions.

2. A variant of an existing MC non-preemptive fixed-priority uniprocessor scheduling policy is proposed to integrate the speed changing between different modes and to remove the undesirable dropping of LO-criticality tasks.

3. Based on a randomly generated task sets, we conduct extensive simulation studies which supports the effectiveness of our algorithm (with respect to energy consumption).

5.2.4 Problem formulation

We consider a task-set $\overline{\tau} = \{\tau_1, \tau_2 \cdots \tau_n\}$, running on an uniprocesssor, where each task $\tau_i \in \overline{\tau}$ is an implicit deadline periodic task, i.e., the task deadline equals its period. Each task $\tau_i \in \overline{\tau}$ generates an unbounded number of jobs $\tau_{i,j}$, where j can be an arbitrarily large number with $j \geq 1$. We represent τ_i by a 5-tuple $\{T_i, C_i^{\text{LO}}, C_i^{\text{HI}}, pET_i, L_i\}$, where T_i is the inter-arrival time between two subsequent jobs of τ_i, C_i^{LO} and C_i^{HI} respectively denote the LO and HI-criticality Worst-Case Execution Time (WCET) of τ_i, pET_i the probabilistic profile, and L_i is the criticality level, where $L_i \in \{\text{LO}, \text{HI}\}$. We also define, for convenience of the subsequent notation, the two complementary subsets $\overline{\tau}_{\text{LO}} = \{\tau_i \in \overline{\tau} : L_i = \text{LO}\}$ and $\overline{\tau}_{\text{HI}} = \{\tau_i \in \overline{\tau} : L_i = \text{HI}\}$.

The proposed approach works with non-preemptive fixed-priority schedulers. The necessity of this restriction on scheduling will be detailed later in this section, while the extension to dynamic priority schedulers is left as future work.

The system is assumed to be equipped with some sort of power/energy control techniques of the hardware, such as the DVFS. To simplify the theoretical analysis, we normalize the system speed such that it is assumed that the system executes all jobs at speed $s_{\text{HI}} \in (0, 1]$ under HI-criticality mode, while $s_{\text{LO}} \in (0, 1]$ denotes the system speed under LO-criticality mode. Specifically, $s = 1$ when the system is running at its maximum speed; $s < 1$ when the system is slowed down to the fraction s of the maximum speed. When $s < 1$, the execution times are linearly scaled according to s. Note that the WCET and the probabilistic information of the task model always refers to

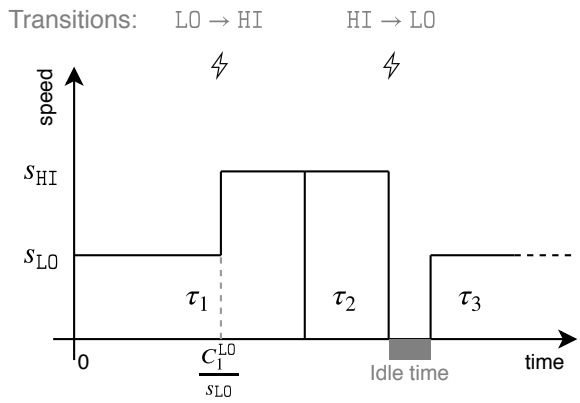


Figure 5.2: *Exemplification of the speed-change mechanism during mode transition to* HI*-criticality and the switch back to* LO*-criticality mode after idle. (Original Source: [29])*

$s = 1$ condition, and they can be considered the *amount of work* to be executed. Given the WCET C_i of the i-th task, the actual execution time in the system becomes $\frac{C_i}{s}$ for under executing speed of s. The period (and the deadline) does not scale according to s and they remain fixed. Since, in our strategy, the speed can change only once per job execution, any overhead to change the DVFS setting is assumed included in the WCET and negligible with respect to the probabilistic execution time. The energy consumed by the job of a task depends on s as well as the actual execution time of the job. We use the symbol $\varepsilon(x, s)$ to indicate the energy function that relates the execution time x and the speed s to the energy consumption. Such function is hardware dependent and can be arbitrary: the following analysis works with any energy model, including nonlinear formulations for $\varepsilon(x, s)$.

Mode switch mechanism and correctness requirements. Most of the existing papers on MC systems adopt the system mode switch effect, i.e., dropping LO-criticality tasks when a HI-criticality task overruns its C_i^{LO}. Instead, we adopted the speed-change mechanism, so that *each* task τ_i (including LO-criticality ones) is guaranteed to be executed under any condition. The system mode switch to HI-criticality causes an increase of the processor frequency to guarantee that all jobs are correctly schedule. The pro-

posed approach is depicted in Figure 5.2. Resource/Energy efficiency is achieved in LO-criticality mode by optimizing the trade-off between the minimum system speed and probability of mode-switch. Let s_{LO} (and s_{HI}) denote the processor speed when the system is in LO-criticality (and HI-criticality, respectively) mode. Same as the classical MC setting, the system switches its mode to HI when a HI-criticality task overruns its C_i^{LO}. During the HI-criticality mode, the processor runs at high frequency s_{HI} in order to be able to schedule all the jobs even in the most pessimistic assumptions. The system reverts to LO-criticality at the end of each hyperperiod, or when the system is idle, whichever comes earlier. The values of s_{LO}, s_{HI}, and C_i^{LO} depend on the schedulability of the task set and they impact the system energy consumption. For this reason, the goal of our approach is to select such parameters so that the energy is minimized, according to probabilistic information, while guaranteeing the schedulability of the whole task-set according to deterministic WCET.

Definition 5.2.1 (Probabilistic profile)**.** The probabilistic profile of the execution time of a task τ_i is identified by the symbol pET_i and it is a $3 \times k$ matrix defined as follows:

$$pET_i = \begin{pmatrix} e_1 & e_2 & \cdots & e_k \\ f_i(e_1) & f_i(e_2) & \cdots & f_i(e_k) \\ F_i(e_1) & F_i(e_2) & \cdots & F_i(e_k) \end{pmatrix} \tag{5.1}$$

where $e_1, e_2, ..., e_k$ are execution time values, $f(\cdot)$ is the pmf, and $F(\cdot)$ the cdf. Even if the last two rows are redundant (pmf from the cdf is computable and vice versa), this simplifies the notation in the subsequent sections. ■

Example 5.2.1. Let consider the following execution profile:

$$pET_1 = \begin{pmatrix} 5 & 7 & 12 & 19 & 20 \\ 0.10 & 0.60 & 0.25 & 0.04 & 0.01 \\ 0.10 & 0.70 & 0.95 & 0.99 & 1 \end{pmatrix}$$

It represents the statistical distribution of the execution time of the task τ_1. The probability that a task requires 5 unit of execution time is 0.1, for 7 unit of execution time case the probability is 0.6 and so on. The last row represents the cdf, for example the probability that the execution time is less or equal to 12 is 0.95. The icdf in this case would be $F^{-1}(0.95) = 12$. ■

In this section, we considered the probabilistic profile, and the related statistical functions, as discrete. To estimate the f_i and F_i of the tasks, an experimental campaign must be carried out to directly measure the execution time of the task. Since, in this section, we are not interested in the probabilistic-WCET, but on the full probabilistic profile of the execution time, the ecdf method described in Section 2.1.1 is used. This method enables to estimate the values F_i by directly measuring n-samples of the execution time. To obtain a precise estimation of the probabilistic profile, the task should experience as much as possible execution conditions during the measurements, i.e. inputs and states. The precision of the ecdf estimation is derived from the Dvoretzky-Kiefer-Wolfowitz inequality described in Section 2.4.1. For example, if $n = 10000$ samples are acquired, with a confidence of $c = 10^{-6}$ the maximum absolute error on the cdf is lower than $\epsilon \leq 0.02$. More sophisticated statistical techniques can obtained better results with a smaller number of samples, but this analysis would be out of scope with respect to the goals of this work. It is important to remark that this probabilistic information is used in this work only for the optimization of the energy and it does not impact the schedulability analysis. Consequently, inaccuracies in the probabilistic execution time estimations affect only the optimality with respect to the energy consumption and it does not affect the satisfaction of the real-time constraints.

WCET in LO-criticality mode. In most of traditional MC works, the value of WCET in LO-criticality mode, i.e. C_i^{LO}, is assumed to be given or empirically selected according to a defined percentage of the HI-criticality WCET. In this work, instead, we compute the C_i^{LO} so that it minimizes the energy consumption[1]. In fact, there is a trade-off between the probability to switch to HI-criticality mode and the minimum achievable speed in LO-criticality mode. Large values for C_i^{LO} would delay the activation of the HI-criticality mode, thus reducing probability of this event to happen, but it requires to increase the minimum speed in LO-criticality mode in order to guarantee the schedulability. While, a small value for C_i^{LO} would decrease the minimum speed in LO-criticality mode, but increases the chances of a mode-switch and, if it happens too frequently, it may produce the opposite effect of increasing the energy consumption. The approach to select C_i^{LO} in this work is based on the probabilistic profile, and in particular, by exploiting the icdf: a value $p^{\text{LO}\rightarrow\text{HI}}$, that represents the probability per job to switch from

[1] Although the calculation under such a purpose would technically lead to execution thresholds that have nothing to do with 'worst-case', we nevertheless still follow the traditional MC work in the real-time and embedded systems community by calling them WCET under the LO-criticality mode.

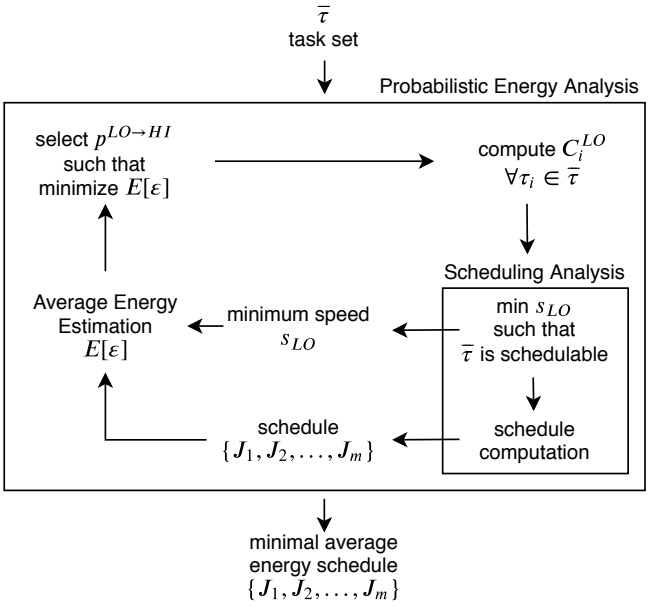


Figure 5.3: *The overview of the proposed energy-optimization approach.*

LO-criticality to HI-criticality, is selected according to the optimization problem and used to compute the WCET with the icdf.

Solution overview. The flow of the approach proposed in this section to solve the previously defined problem is outlined in Figure 5.3. Note that, real-time task scheduling requires a strict timing guarantee. To ensure that the WCET prediction does not compromise the system schedulability, we propose an approach composed of two optimization algorithms that contribute to obtaining the schedule that requires a minimum average energy consumption while guaranteeing that no task will miss the deadline. The probabilistic information is used only for the energy minimization and not for the schedulability test. Our schedulability test remains safe by using the deterministic WCET, i.e., as far as the worst-case estimations are trustworthy, our solution will guarantee worst-case correctness.

1. The outer optimization that selects the WCET of the task in LO-criticality mode,

i.e. C_i^{LO}, by choosing the best value for mode switch probability $p^{\text{LO}\to\text{HI}}$ that leads to the minimum average energy. This goal can be formally written with the following optimization problem:

$$\min_{p^{\text{LO}\to\text{HI}}} E[\varepsilon(x, s_{\text{LO}})] \tag{5.2}$$

where $E[\cdot]$ is the expected value operator over the execution time x and s_{LO} is computed by the inner optimization algorithm from $p^{\text{LO}\to\text{HI}}$.

2. The inner optimization chooses the minimal s_{LO}, which ensures the task set to be schedulable according to the response time analysis. This analysis is not reported in this thesis but available in the original paper [29]. Once we compute s_{LO} and generate an output schedule, the energy estimation procedure takes this schedule as input. This computes, thanks to the probabilistic information, the average energy, which is, in turn, used in the outer optimization algorithm.

3. Finally, the optimal values for s_{LO} and $p^{\text{LO}\to\text{HI}}$ are computed, together with the optimal schedule, with respect to the average energy minimization, to be applied online.

5.2.5 From probabilistic execution time to average energy consumption

According to the output of the response time analysis described in the original paper [29] and a fixed priority assignment protocol, it is possible to build an ordered schedule of m periodic jobs $\{J_1, J_2, ..., J_m\}$ forming a complete hyperperiod. Note that $J_i \in \{\tau_{j,k}\}\ \forall j, k$ where $\tau_{j,k}$ is the definition of job of the previous section. The notation change is necessary to highlight the job position in the overall job schedule. Formally, we also assume that each job has the same properties of its parent task, e.g., the WCET C_i^{HI} of the job J_i is the same value C_k^{HI} of the task τ_k such that $J_i = \tau_{k,j}$ for some k, j. Because all the jobs are periodic and the scheduling is fixed-priority non-preemptive, the job schedule is known at design-time and fixed. This restriction allows us to perform the following energy analysis.

5.2.6 Determining a mixture distribution for pET

As part of the minimization problem, we assume in this section to have selected a system-level probability $p^{\text{LO}\to\text{HI}}$, as the probability of a HI-criticality job to overruns its C_i^{LO}. The value C_i^{LO} is computed with the icdf of the pET_i distribution: $C_i^{\text{LO}} = F_{X_i}^{-1}(1 - p^{\text{LO}\to\text{HI}})$, where X_i is the random variable of the execution time distributed according to pET_i. In the task model we assumed that pET_i, C_i^{LO}, and C_i^{HI} have been computed for a processor speed of $s = 1$. According to the linearity assumption of the execution time with respect to the processor speed, we define the random variable X_i^{LO} as the execution time in LO-criticality mode of the i-th task. The probabilistic profile of its execution time is:

$$pET_i^{\text{LO}} = \mathcal{T}_x \left[\begin{pmatrix} s_{\text{LO}} \\ 1 - p^{\text{LO}\to\text{HI}} \\ 1 - p^{\text{LO}\to\text{HI}} \end{pmatrix}^{\odot -1} \odot pET_i \right] \tag{5.3}$$

where $\odot$ is the symbol of Hadamard product[2] and Hadamard power[3], x is the column number of C_i^{LO} in pET_i and $\mathcal{T}_x$ is the function that truncated the matrix from $3 \times k$ to $3 \times x$ by removing the upper part. For LO-criticality tasks, $x = k$, thus no truncation occurs. Similarly we define the probabilistic execution time when the task starts in HI-criticality mode pET_i^{HI} and when it starts in LO-criticality but a mode switch occurs to HI-criticality pET_i^{TR}:

$$pET_i^{\text{HI}} = \begin{pmatrix} s_{\text{HI}} \\ 1 \\ 1 \end{pmatrix}^{\odot -1} \odot pET_i \tag{5.4}$$

$$pET_i^{TR} = \mathcal{T}'_x \left[\begin{pmatrix} s_{\text{HI}} \\ p^{\text{LO}\to\text{HI}} \\ p^{\text{LO}\to\text{HI}} \end{pmatrix}^{\odot -1} \odot pET_i + \begin{pmatrix} K & \cdots & K \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{pmatrix} \right] \tag{5.5}$$

where $K = C_i^{\text{LO}}/s_{\text{LO}} - C_i^{\text{LO}}$, x is the column number of C_i^{LO} in pET_i and $\mathcal{T}'_x$ is the function that truncated the matrix from $3 \times k$ to $3 \times (k - x)$ columns by removing the

[2]Each element in the i-th row of the left matrix is multiplied by the i-th number of the right vector; the output has the size of the matrix.

[3]Element-wise power operation; the output has the size of the vector.

lower part. The pET_i^{TR} represents only the upper-part of the execution time, i.e. after the transition to HI-criticality mode. This means that it has already executed for a time period of length $C_i^{\text{LO}}/s_{\text{LO}}$ and the amount of work already completed is C_i^{LO}.

Example 5.2.2. Given the task of Example 5.2.1 and choosing a HI-mode switch probability $p^{\text{LO}\to\text{HI}} = 0.05$ with $s_{\text{LO}} = 0.5$ and $s_{\text{HI}} = 1$, the WCET for the LO-criticality mode will be $C_i^{\text{LO}} = 12$ and the probabilistic execution time in LO-criticality mode is:

$$pET_1^{\text{LO}} = \begin{pmatrix} 10 & 14 & 24 \\ 0.11 & 0.63 & 0.26 \\ 0.11 & 0.74 & 1 \end{pmatrix}$$

The probabilistic execution time when the task starts HI-criticality mode is equivalent to the original distribution because $s_{\text{HI}} = 1$:

$$pET_1^{\text{HI}} = pET_1 = \begin{pmatrix} 5 & 7 & 12 & 19 & 20 \\ 0.10 & 0.60 & 0.25 & 0.04 & 0.01 \\ 0.10 & 0.70 & 0.95 & 0.99 & 1 \end{pmatrix}$$

And the probabilistic execution time after a transition of the task from LO-criticality to HI-criticality is:

$$pET_1^{TR} = pET_1 = \begin{pmatrix} 31 & 32 \\ 0.8 & 0.2 \\ 0.8 & 1 \end{pmatrix}$$

When the transition occurs, the task has already executed 12 amount of work in 24 time units, while 7 and 8 units remain to be executed in HI-criticality mode, for a total of 31 and 32 units of time. ■

Figure 5.4 depicts the probability tree diagram for a generic HI-criticality job J_i. In particular, at the beginning of the execution of the job J_i, the system can be in LO-criticality mode or in HI-criticality mode, respectively with probability $1 - p_i^{\text{HI}}$ and p_i^{HI}. In the latter case, the job runs in HI-criticality mode and its execution time is distributed according to the pET_i^{HI} distribution. In the first case, the job starts to run in LO-criticality mode and possibly switches to HI-criticality if it overruns its C_i^{LO}. This happens with the already defined probability $p^{\text{LO}\to\text{HI}}$ universally set for all tasks. The

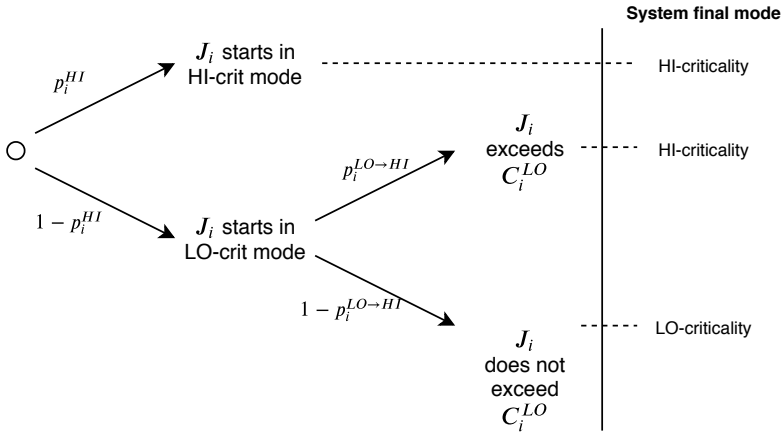


Figure 5.4: *The probabilistic event tree of an* `HI`*-criticality job. (Original Source: [29])*

task execution time is distributed in the `LO`-criticality part according to the pET_i^{LO} distribution, while after mode switch it is distributed according to the pET_i^{TR} distribution. Consequently, we can write the probability mass function of the execution time of the i-th job starting to run in `LO`-criticality mode as the following *mixture of mass functions*:

$$f_i^{\text{LO+HI}}(x) = (1 - p^{\text{LO}\to\text{HI}}) f_i^{\text{LO}}(x) + (p^{\text{LO}\to\text{HI}}) f_i^{TR}(x) \tag{5.6}$$

where $f_i^{\text{LO}}(x)$ is the pmf of pET_i^{LO} and $f_i^{TR}(x)$ is the pmf of pET_i^{TR}. This function includes both the case that J_i exceeds its threshold and the case that it finishes before it. To characterize the execution time distribution of J_i, we need to know the probability that the job starts in `HI`-criticality mode: p_i^{HI}, which depends on the previously executed jobs. In particular, since the $p^{\text{LO}\to\text{HI}}$ is the same values for all tasks, it depends on the number of previous `HI`-criticality jobs and the probability of incurring in idle time (that would switch back the system mode to `LO`-criticality). Informally, the probability p_i^{HI} can be written as:

$$p_i^{\text{HI}} = P[(A \cup B) \cap C] \tag{5.7}$$

where the following events have been considered:

- $A : J_{i-1}$ starts in `HI`-criticality mode

Algorithm 5.1 Estimation of the probability for a job to start in HI-criticality mode (p_i^{HI}). (Original Source: [29])

1: $p_1^{\text{HI}} \leftarrow 0$
2: $pET_{\text{conv}} \leftarrow pET_1^*$
3: **for** $j \leftarrow 2$ **to** $i-1$ **do**
4: ▷ Compute probability for $P(A \cap C)$
5: $P(A) = p_{j-1}^{\text{HI}}$
6: $P(A|C) = 1 - F_{j-1}^*\left(a_j \left| \bigcup_{k<j}(X_k > C_k^{P_{\text{LO}}})\right.\right)$
7: $P(A \cap C) = P(A)P(C|A)$
8: ▷ Compute probability for $P(B \cap C)$
9: $P(B) = P(X_{j-1} \geq C_{i-1}^{\text{LO}}) = 1 - F_j^{\text{LO}}(C_{j-1}^{\text{LO}})$
10: $P(C) = P(\sum_k^{j-1} X_j \geq a_i) = 1 - F_{\text{conv}}^*(a_j)$
11: $P(B \cap C) = P(B)P(C)$
12: ▷ Update variables
13: $p_j^{\text{HI}} = P(A \cap C) + P(B \cap C)$
14: $pET_{\text{conv}} \leftarrow$ convolute($pET_{\text{conv}}, pET_j^*$)
15: **end for**

- B : J_{i-1} exceeds C_{i-1}^{LO}
- C : system not idle after J_{i-1}

The events are not independent, but the probability p_i^{HI} can be computed incrementally as follows. Equation (5.7) can be rewritten as $p_i^{\text{HI}} = P(A \cap C) + P(B \cap C)$, as the events A and B are disjoint (consequently, the probability of the union of events is the sum of their probability). P(B) is always 0 if the job is belonging to a LO-criticality task. The computation of p_i^{HI} is showed in Algorithm 5.1. The algorithm starts with the fact (line 1) that the first job would run in LO-criticality mode (any HI-criticality mode is reset to LO-criticality at the end of the hyperperiod). From this, it is possible to compute the $pET^{\text{LO+HI}}$ thanks to Equation (5.6) and consequently compute the cdf needed for $P(A|C)$ (line 6). Then, $P(B \cap C)$ can be easily computed as B and C are independent: the execution time of each job is independent with the previous ones. To compute C we need the distribution of the sum of execution time, that is incrementally built using the convolution operator (line 14).

Once the value p_i^{HI} is recursively computed, we can compute the final mixture of

probability mass distribution of the execution time of the job J_i that takes in account any system mode conditions and switches:

$$f_i^*(x) = (1 - p_i^{\text{HI}}) f_i^{\text{LO+HI}}(x) + p_i^{\text{HI}} f_i^{\text{HI}}(x) \tag{5.8}$$

Average energy consumption. In our assumptions, the energy consumption of a job is a function $\varepsilon(x, s)$, where x is the execution time and s the processor speed. This function can be either linear or superlinear (which is the case for almost all existing energy models). To compute the average energy from the probabilistic information, it is sufficient to apply the energy function to each element of the first rows of pET_i^{LO}, pET_i^{HI}, and pET_i^{TR}, with respectively $s = s_{\text{LO}}$ for the first and $s = s_{\text{HI}}$ for the last twos, obtaining 3 new probabilistic energy profiles pEC_i^{LO}, pEC_i^{HI}, and pEC_i^{TR}. The energy analysis is performed by recomputing Equation (5.6) and Equation (5.8) using the new matrices pEC_i. Consequently, the final $f_i^*(x)$ function becomes the statistical distribution of energy. From this distribution we can compute the expected value of the energy $E[\varepsilon] = \sum x \cdot f_i^*(x)$ that is, in turn, used as optimization cost function in Equation (5.2). The computational complexity of the energy analysis is mainly dominated by the convolution operator in line 14 of Algorithm 5.1. The perfect convolution has exponential complexity, however, state-of-the-art approximated solutions exist with complexity $O(m \log m)$ where m is the number of columns of the previous pET matrices. Algorithm 5.1 is executed for each job, so the overall complexity is $O(n \cdot m \log m)$, where n is the number of jobs. Even if the value of n is not necessarily small, the analysis is performed at design-time, it has no overhead at run-time, and the scheduler is very simple.

Example 5.2.3 (Probabilistic energy computation)**.** To better clarify the whole probabilistic algorithm, we describe a toy, but complete, example. Let us consider a simple task-set composed of one `HI`-criticality task and two `LO`-criticality tasks $\bar{\tau} = \{\tau_1, \tau_2, \tau_3\}$ with the following characteristics:

$$\tau_1 = (15, C_1^{\text{LO}}, 6, pET_1, \texttt{HI})$$
$$\tau_2 = (30, 5, //, pET_2, \texttt{LO})$$
$$\tau_3 = (30, 3, //, pET_3, \texttt{LO})$$

and the following probabilistic profiles[4]:

$$pET_1 = \begin{pmatrix} 3 & 6 \\ 0.95 & 0.05 \\ 0.95 & 1 \end{pmatrix}, pET_2 = \begin{pmatrix} 2 & 5 \\ 0.95 & 0.05 \\ 0.95 & 1 \end{pmatrix}$$

$$pET_3 = \begin{pmatrix} 1 & 3 \\ 0.95 & 0.05 \\ 0.95 & 1 \end{pmatrix}$$

We set $p^{\text{LO}\to\text{HI}} = 0.05$ and $s_{\text{HI}} = 1$. We compute the LO-criticality WCETs for the HI-criticality task: $C_1^{\text{LO}} = 3$. The minimum speed computed according to the schedulability analysis is $s_{\text{LO}} = 0.7$. Then, we obtain – similarly to Example 5.2.2 – the three probabilistic profiles in LO-criticality mode:

$$pET_1^{\text{LO}} = \begin{pmatrix} {}^{30}\!/_{7} \\ 1 \\ 1 \end{pmatrix}, pET_2^{\text{LO}} = \begin{pmatrix} {}^{20}\!/_{7} & {}^{50}\!/_{7} \\ 0.95 & 0.05 \\ 0.95 & 1 \end{pmatrix}$$

$$pET_3^{\text{LO}} = \begin{pmatrix} {}^{10}\!/_{7} & {}^{30}\!/_{7} \\ 0.95 & 0.05 \\ 0.95 & 1 \end{pmatrix}$$

Since $s_{\text{HI}} = 1$, the HI-criticality profiles are equivalent to the original ones: $pET_i^{\text{HI}} = pET_i$. The probabilistic profile for transitions is defined for the HI-criticality task only (because LO-criticality tasks cannot generate a mode switch):

$$pET_1^{TR} = \begin{pmatrix} 6 + {}^{30}\!/_{7} - 3 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{51}\!/_{7} \\ 1 \\ 1 \end{pmatrix}$$

We consider a Rate Monotonic scheduler with hyperperiod $H = 30$. The schedule is then: $\{J_1, J_2, J_3, J_4\} \in \{\tau_1, \tau_2, \tau_3, \tau_1\}$. The Algorithm 5.1 can now be used to obtain

[4]Please note that this is an abuse of notation for pET_i, because its suffix i refers to the job and not to the task. However, as subsequently presented, the first three jobs J_1, J_2, J_3 correspond to the three tasks τ_1, τ_2, τ_3.

the final probabilistic time profile. The simplicity of the example allows us to explain step-by-step the solution:

1. $p_1^{\text{HI}} = 0$: the first job starts for sure in LO-mode
2. $p_2^{\text{HI}} = p^{\text{LO}\to\text{HI}}$: referring to Equation (5.7), the event B is not possible because J_2 belongs to a LO-criticality task; C is always verified because the arrival times of J_2 and J_3 are the same and the scheduler is work conserving
3. $p_3^{\text{HI}} = p^{\text{LO}\to\text{HI}}$: the considerations of the previous step are valid also for J_3
4. $p_4^{\text{HI}} = 0.000125$, as subsequently explained

To compute p_4^{HI} we should have computed the pET^* and then the convolution of them, to obtain the probability $P(C)$, i.e. there is no idle time after J_3. For the sake of this example, this probability could be easily computed by the following reasoning: J_4 can start in HI-mode only if J_1 switched to HI-mode (with probability 0.05) and there is no idle time after J_3. This happens when both J_2 and J_3 runs until their WCET 5 and 3: the probability of this event is $0.05 \cdot 0.05 = 0.0025$. Then, $p_4^{\text{HI}} = 0.05 \cdot 0.0025 = 0.000125$. Assuming, for simplicity, that $\varepsilon(x, s) = x$, we compute the average energy per job:

$$E_1[\cdot] = \sum_x x(0.95 f_1^{\text{LO}}(x) + 0.05 f_1^{TR}(x))$$

$$E_2[\cdot] = \sum_x x(0.95 f_2^{\text{LO}}(x) + 0.05 f_2^{\text{HI}}(x))$$

$$E_3[\cdot] = \sum_x x(0.95 f_3^{\text{LO}}(x) + 0.05 f_3^{\text{HI}}(x))$$

$$f_4^{\text{LO}+\text{HI}}(x) = 0.95 f_4^{\text{LO}}(x) + 0.05 f_4^{TR}(x)$$

$$E_4[\cdot] = \sum_x x \left[(1 - 0.000125) f_4^{\text{LO}+\text{HI}}(x) + 0.000125 f_4^{\text{HI}}(x)\right]$$

For example, the first term is computed as:

$$E_1[\cdot] = {}^{30}\!/_{7}(0.95 \cdot 1 + 0.05 \cdot 0) + {}^{51}\!/_{7}(0.95 \cdot 0 + 0.05 \cdot 1) = 4.44$$

The total average energy is then the sum of the individual energy components:

$$E[\varepsilon] = E_1 + E_2 + E_3 + E_4 = 4.44 + 3.02 + 1.54 + 4.39 = 13.39$$

■

5.2.7 Simulation evaluation

In this section, we report the evaluation result of our algorithm regarding the energy optimization. The results related to the scheduling analysis are available on the original paper [29]. To simplify the experimental evaluation, without loosing generality, we considered $s_{\text{HI}} = 1$, i.e., the maximum achievable speed in `HI`-criticality mode is set at the maximum processor speed.

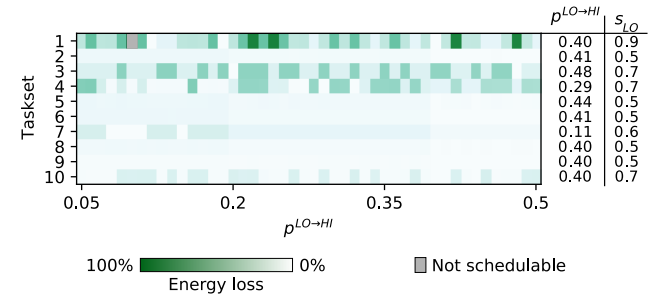
Simulation parameters. In order to perform the simulation, we selected the NXP i.MX6 SABRE board, for which the energy function and the relation voltage-frequency is known [179]. The speed is assumed proportional to the frequency[5]. The considered energy function is:

$$\begin{aligned} \varepsilon(x, s) &= x\left(aC \cdot V^2 f + P_{\text{leak}}\right) \\ aC &= 3.4 \cdot 10^{-10}, \quad P_{\text{leak}} = 0.052 \\ V^2 f &= \left(0.95 + 0.0005 \frac{f - 396 \cdot 10^6}{10^6}\right)^2 \cdot f \end{aligned}$$

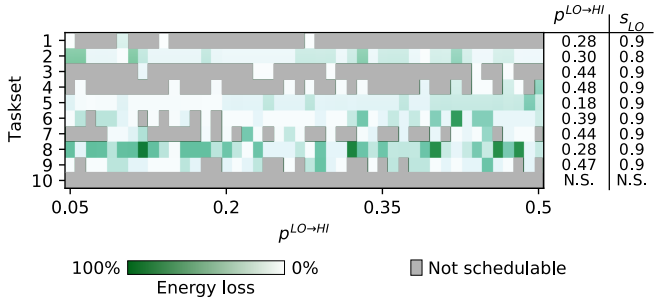
where f is the frequency computed as $f = s \cdot \text{max\_freq}$, aC has been experimentally obtained, $V^2 f$ comes from [179] and P_{leak} has been taken from the SoC datasheet. The selected values for exploration are $p^{\text{LO}\to\text{HI}} \in [0.01; 0.50]$ and $s_{\text{LO}} \in [0.5; 0.9]$. The considered fixed priority assignment algorithm is Rate Monotonic (RM).

Results. Figure 5.5 depicts the exploration of 10 random task-sets by varying $p^{\text{LO}\to\text{HI}}$, i.e., the results of the external optimization algorithm. Figure 5.5a refers to a total utilization of $U = 0.25$ and Figure 5.5b refers to a total utilization of $U = 0.50$. The figures show the schedulability and the amount of energy saved with respect to the minimum energy consumption of each dataset. When utilization is low, the task-sets are schedulable for any value of $p^{\text{LO}\to\text{HI}}$ (with only one exception for task-set 1). Instead, when the utilization increases, it is possible to notice that $p^{\text{LO}\to\text{HI}}$ represents a critical choice to the schedulability: the schedulability for $U = 0.50$ increases to 90% compared to the range of 50%-70% from the schedulability analyses presented

[5]This is not necessarily true in architectures with variable timing instructions. However, in this case, the workload would play a key role in the definition of the energy function, as also in the WCET analysis. In any case, the presence of variable instructions timing would not invalidate our analysis, but it would add excessive verbosity without adding any innovative content.



(a) $U = 0.25$



(b) $U = 0.50$

Figure 5.5: *Energy of the 10 datasets analyzed with our framework, depicted by the energy loss percentage with respect to the minimum found for each dataset. The* $(p^{\mathrm{LO}\to\mathrm{HI}}, s_{\mathrm{LO}})$ *couples with the minimal energy consumption are showed on the right columns. (Original Source: [29])*

in the original paper [29]. The choice of $p^{\mathrm{LO}\to\mathrm{HI}}$, as expected, also impacts on energy optimization. The optimization function is clearly non-convex and numerical methods are necessary. Some choices of $p^{\mathrm{LO}\to\mathrm{HI}}$ are very far from the optimal, for example, dataset 1 in the $U = 0.25$ case, the value $p^{\mathrm{LO}\to\mathrm{HI}} = 0.24$ leads to a energy consumption value 1.79x times larger than the optimal value at $p^{\mathrm{LO}\to\mathrm{HI}} = 0.4$. For other task-sets, like 6-9 in Figure 5.5a, the saved energy is instead small. No direct relationship can be derived with respect to the s_{LO} value: increasing utilization requires a larger s_{LO}, but no general conclusions can be drawn for a fixed utilization value.

We run a simulation with 2500 task-sets by varying the utilization in the range $(0, 1]$.

The analysis carries out the optimal value for $p^{\text{LO}\to\text{HI}}$ that corresponds to the minimal energy. Then, this value is compared to the baseline of randomly select $p^{\text{LO}\to\text{HI}}$. The improvements on energy consumption corresponds to 10.0% on average and 46.4% on maximum energy saved values.

5.2.8 Lesson learned

Previous works usually assumed given the value of LO-criticality WCET C_i^{LO}, or fixed to a given proportion of the HI-criticality WCET. However, when energy plays a role in the analysis, that choice is critical to reach the optimality. This is especially evident when DVFS mechanism is used to reduce the power consumption during the LO-criticality mode of the system, because the choice of s_{LO} impacts on both schedulability and energy consumption. Choosing these values to optimize the energy for the average-case (the most common) is non-trivial. Our approach optimized the energy consumption by selecting the best value for $p^{\text{LO}\to\text{HI}}$, and consequently the value of C_i^{LO}, and the best value of s_{LO}, obtaining a significant value of saved energy.

Applicability to real systems. Despite this work is highly theoretical, the deployment of this approach is straightforward. In fact, both scheduling and probabilistic analyses are performed offline and a well-known fixed-priority scheduler is used at run-time, minimizing the impact on software development and integration. The probabilistic profiles are obtained by measuring the execution time of the tasks directly on the real platform, while the WCET is computed with a state-of-the-art static timing analysis tool. Even if our analysis does not take into account scheduling and DVFS-change overheads, the number of frequency changes is upper-bounded to only one time per HI-criticality job. Assuming that the overheads are small compared to the execution time of the tasks, they can be simply added to the tasks WCET, without a significant impact on the solution optimality.

Future works. This work may trigger several future research directions. Here, we restrict our attention to the uniprocessor platform and dual-criticality levels. We plan to extend our work to adopt the multi-processor platform and to consider multi-criticality levels. Another research direction includes the study of how to apply the analysis to variable priority and/or preemptive schedulers, e.g., Earliest Deadline First. This may include an on-board implementation of our algorithm to measure the actual energy sav-

ings and to study online reactive algorithms. Such algorithms can potentially tune the scheduling parameters at run-time in order to reach the optimal energy consumption even in the case when external factors modify the initial assumptions.

5.3 Heterogeneous Computing and HPC

Section 1.4.2 discussed the emerging timing requirements of HPC applications. To satisfy such requirements, the current approaches described in Section 2.5.3 are not sufficient to obtain real-time performance, and traditional real-time techniques from the embedded world are not applicable because moving HPC resources towards specialized components and custom real-time operating systems is not only hardly feasible from the cost standpoint, but it would probably reduce the overall system utilization. With specialized components, the general-purpose capability of the architecture is lost, reducing the number of executed jobs and requiring applications to be specifically developed for the specialized architecture. Moreover, real-time software and hardware components would sacrifice the overall average performance, which is not acceptable for shared HPC systems, where the throughput and the system capacity are the flagship performance metrics.

5.3.1 The proposed solution

To guarantee the timing predictability of the applications, without impacting the HPC infrastructure and its overall average performance, we propose relying on probabilistic real-time and, in particular, on MBPTA. In HPC, the safety requirements are less tight than embedded applications because a full certification process is not required. Moreover, for several HPC applications, the scale of the time constraints is typically more coarse-grained compared to embedded systems: the order of magnitude of the execution time of a typical HPC job is minutes or hours. Conversely, in embedded systems, typical applications include tasks with deadlines below one second. In practice, this means that missing a deadline for few processor cycles is negligible for HPC requirements – while it may not be for some embedded scenarios. For these reasons, the HPC scenario can still benefit from probabilistic real-time approaches, despite the work-in-progress status of its theoretical framework. Finally, it is worth considering that accurate WCET estimations are not only needed for satisfying application requirements, but they can be very attractive to support the run-time resource management and job scheduling sys-

tem. Suitable resource allocation and scheduling policies may, in fact, benefit from the possibility of knowing in advance the number of computing resources to allocate to a ready task, in order to guarantee a given WCET (or vice versa). This strategy would implicitly minimize the resource over-provisioning, and thus costs while maximizing the workload in execution.

In this section, we use the term *Cyber-Physical Systems (CPS)* to refer to the systems where MBPTA is usually applied, such as the embedded systems, as opposed to the HPC.

5.3.2 From cyber-physical systems to high-performance computing

The use of probabilistic real-time techniques in CPS differs from the HPC case. This section presents the difference in the task models, the current status and limits of probabilistic real-time research in CPS, and the advantages of using this technique for HPC.

The differences of system models. Before proceeding with the following discussion, it is necessary to clarify a difference in the terminology currently used in the respective contexts: CPS and HPC. In real-time CPS, a task is part of an application and it is characterized as we presented in Section 2.4, i.e., in its simplest form, by the following task model: $\tau_i = (T_i, D_i, C_i)$. The task "spawns" jobs at (a)periodic intervals, that are usually considered single-thread and they are characterized by an arrival time $a_{i,j}$ and a finishing time $f_{i,j}$. The difference $s_{i,j} - a_{i,j}$ represents the waiting time for the job, where $s_{i,j}$ is the starting time of the j-th job spawned by the i-th task. In HPC, a job is a different concept: the user sends a request to the resource manager to run a particular job with pre-defined resource requirements, such as the number of nodes and processing cores. Once the resources are available, the resource manager launches the job execution on one or more nodes. The HPC job is much more complex than the CPS one. It usually consists of multiple threads and/or multiple processes that usually need to synchronize and share data. Regarding the timing properties, similar to CPS, a job has an arrival time $a_{i,j}$, a starting time $s_{i,j}$, and a finishing time $f_{i,j}$ that should be $f_{i,j} \leq a_{i,j} + D_i$. The difference $f_{i,j} - a_{i,j}$ represents the turnaround time. The finishing time in the HPC, and consequently the turnaround time, depends on many factors, even considering the job to be non-preemptible. The starting time $s_{i,j}$ is equal for all the

processes and threads composing the job[6], but the (worst-case) execution time depends on the interaction among the processes and threads. If we consider the unusual case of having the execution of processes and threads not synchronizing or sharing data among each other, the WCET C_i is defined as the maximum WCET among all the threads/processes. Otherwise, synchronization and data sharing have to be taken into account in the model. To summarize, the job in the CPS sense is a simple single execution of a single thread, while the job of in the HPC sense is a single execution of a complex entity, composed of multiple threads and/or processes, possibly interacting with each other, likely running on different processors and nodes. While the CPS timing properties are easy to formalize, the HPC scenario presents several challenges even in the formal modeling, due to the interactions among threads, processes, and computational resources. From now on, in this section, any reference to *job* means the HPC definition: the single application instance (possibly running on multi-cores and/or multi-nodes) (a)periodically launched by the resource manager onto the computing nodes. The waiting time in HPC depends not only on the scheduler itself but also on external factors, such as administrative decisions and the availability of nodes. For this reason, in this section, we focus on the probabilistic estimation of the WCET C_i only.

A recent work by Fusi et al. [98] studied the applicability of probabilistic real-time techniques for a geophysical exploration HPC application. In particular, they employed a memory-placement randomization technique in software to reduce the cache interference and improve the satisfaction of the iid hypothesis. The paper is the most similar work to ours and the only one applying probabilistic real-time to HPC. However, there is an essential difference in the two works: In the cited paper, the authors assumed the task to run several jobs in the same process execution, i.e., they assumed the traditional CPS concept of job. For this reason, the two papers are pursuing different goals: the paper by Fusi et al. [98] studied the applicability of probabilistic real-time on the executions spawned by a single HPC job, while our paper deals with execution times across job executions. In our work, the cache randomization would have no effect because no cache data is maintained across job executions.

On the advantages of using pWCET in HPC. The novel idea proposed is to apply the probabilistic WCET analyses for Cyber-Physical Systems (CPS) to HPC environ-

[6]Network delays and other overheads may delay the starting of the execution on some nodes, making the starting time $s_{i,j}$ to be slightly variable across the nodes.

ments, as a solution to address the timing requirements of new emerging application domains. The availability of a new WCET metric in HPC systems would open the resource management research field to new possibilities, i.e., to investigate policies aiming at guaranteeing minimal latency or task completion time in HPC. In this section, we introduce the potential benefits of using the probabilistic WCET approach in HPC, followed by a discussion on the applicability barriers that may prevent it from actually being adopted. While the development process of CPS is focused on certifications, in HPC the focus is entirely on performance metrics. It is then possible to explore more innovative approaches, even if they are not mature enough to be able to pass rigorous certification processes. For this reason, the employment of probabilistic-WCET approaches in HPC would also open the door to novel improvements:

- The first goal aims at providing a way to specify the application time constraints. The typical example is a maximum response-time latency constraint, where a task must provide an output within a certain time frame. As previously discussed in Section 1.4.2, this can be a critical requirement for some HPC applications, e.g., imminent disaster detection or real-time medical imaging.

- The second goal aims at providing the resource manager, both at node-level and at cluster-level, with an estimation of the WCET value with a higher level of confidence. In most current resource managers – e.g., SLURM [260] – the WCET value is usually provided by the user and used to assign job priorities. Frequently, the user either underestimates this value, leading to the forced termination of the application and, consequently, waste of the resources or overestimates this value, leading to unfair resource allocation. Computing the WCET thanks to a measurement-based approach can enable the resource manager to autonomously infer the WCET when the same application (or job) is executed several times, instead of asking the user to provide an estimate of it, which is usually inaccurate.

- Finally, the last goal can be seen in several aspects. For example, from the WCET it is possible to compute other non-functional metrics such as the Worst-Case Energy Consumption (WCEC) and its probabilistic version pWCEC described in the next Chapter 6. Similar reasoning can be applied for power or thermal behaviors. Another different possible use case scenario can be the evaluation of time budgets of HPC users. Novel time accounting mechanisms can be studied, for instance, by giving latency-sensitive applications higher priority on resource allocation, but

also a larger budget cost rather than non-time-sensitive applications.

5.3.3 Qualitative analysis of HPC systems

The complexity of HPC infrastructure makes it practically impossible to formally verify the EVT hypotheses previously described. In this section, we qualitatively discuss the general applicability of such conditions to HPC systems. Then, in Section 5.3.4, we show experimental results collected by executing benchmark applications on a real HPC infrastructure.

General considerations. Modern HPC systems are composed of an extensive set of computing nodes connected through a high-speed network. Each node usually includes multiple multi-core processors. In most of the cases, the hyper-threading configuration of such processors is disabled, because it is not always favorable to HPC applications [159, 203]. Some nodes may include, other than general-purpose processors (CPUs), a heterogeneous set of processing units, like GPUs or specialized hardware accelerators. The use of such resources requires the application to use specific programming models, like OpenCL or CUDA, to enable the *heterogeneous computing* paradigm, which offers more chances of improving performance and energy efficiency. From the user perspective, the goal is to minimize the job turnaround time. This time span includes the overall time spent by the application in the waiting queue of the job scheduler and the actual time spent on running. While the former contribution depends on the availability of resources and, consequently, the contention due to the need to serve other users, the execution time depends mainly on the application itself and how much it can scale on parallel architectures. However, there is a variability in execution times, which depends on several factors. First, a multi-threaded job is potentially affected by higher uncertainty. This is because threads concurrently running on the same multi-core processor can cause contention while accessing shared resources (e.g., cache memories, peripherals, memory buses). Second, the operating systems and other background services are a further source of variability. This uncertainty can also be observed in the execution of maximum priority real-time tasks in general-purpose operating systems. Third, modern COTS motherboards and CPUs may run unpredictable, and non-maskable SMI routines, to manage the occurrence of low-level hardware events [146]. This problem becomes worse when we move from an intra-node to inter-node parallelism. This is because network latencies are substantially unpredictable and affected by several factors:

the operating system drivers, network congestion, hop distance between nodes, packet loss, network devices' internal logic, etc.

EVT hypotheses. In order to apply the pWCET and EVT process to the HPC scenario, the hypotheses described in Section 2.2.3 must be verified. Guaranteeing such hypotheses a priori is not easy and probably not practical on HPC systems due to their complexity. For this reason, in the subsequent experimental evaluation, we use statistical test procedures to verify the hypotheses. In the following paragraphs, instead, we identify the possible barriers that may hinder their satisfaction.

- **The iid hypothesis** As we have already explained in Section 2.2.3, the i.i.d. hypothesis can be divided into three sub-hypotheses: stationarity, short-dependence, and long-dependence. The stationarity hypothesis is the major problem in HPC systems. Let us consider a single-node job execution; if the job does not occupy the whole node, other jobs running on the same node can affect its execution time. Thus, in general, observing the execution times on a non-shared node with respect to a shared node means observing two different distributions, and the execution times can potentially be a non-stationary statistical process. The same happens if nodes are not homogeneous in terms of configuration, e.g., different processor models, current cores frequency, or disk latencies. However, supercomputing facilities are often composed of homogeneous nodes. Therefore, the hardware could be not a real issue in practice. The management software, instead, may represent a critical component, e.g., energy-saving strategies may perform Dynamic Voltage Frequency Scaling (DVFS) while jobs are in execution. Regarding the two dependency requirements, it is hard to make any general conjecture. The jobs are spawned potentially on different nodes of the HPC cluster, thus short- or long-range dependence among them is difficult to imagine. Traditional problems, such as cache locality dependence, do not affect our scenario because the goal is to estimate the pWCET at the job-level. In fact, a job instance is unlikely to exploit the cached data of a previous job, which is a different fully-fledged process. Moreover, there is no guarantee that a job is spawned on the same node of the previous one. On the other hand, network and data locality may still play a role, depending on the resource manager's decision. For example, if the resource manager assigns the same node to all the jobs of our application, it is possible that some data might already be available in subsequent executions. The *Storage*

Area Network (SAN) plays a role here: as shown by Unat et al. [248], recent SAN strategies are created to guarantee data locality in HPC. The locality may hinder our independence hypothesis. Luckily, independence is not a strict requirement because, as shown by previous works [69], EVT can correctly estimate pWCET distribution even in the presence of moderate dependency.

- **The MDA hypothesis**: This requirement has no direct correlation with hardware or software features, it is rather a statistical property. From previous statistical works, it is known that most of all continuous distributions satisfy this hypothesis, provided that the measurements are correctly acquired [236]. During the real measurements of our system, we clearly need to discretize the elapsed time. However, if the time sampling has a sufficient resolution, this hypothesis can be considered valid with a reasonable degree of confidence, as also shown by the subsequent experimental evaluation.

- **Representativity**: The representativity hypothesis depends directly on the application itself. The developer must make sure to have covered all the *application behaviors*, so the time measurement samples must be acquired from all the possible statistical distributions that we can observe. Further details on this hypothesis have been provided in Section 4.2.

Heterogeneous hardware and predictability. Despite the increasing complexity of the HPC infrastructures, we believe that the presence of a heterogeneous set of computing devices can offer some opportunities, in terms of allocation of tasks or jobs with real-time requirements. Given this scenario, we formulate the following hypothesis: the presence of heterogeneous processors in the infrastructure can help the fulfillment of the EVT hypotheses and minimize the WCET overestimation. In the following experimental section, we aim to verify, through statistical testing procedures, this hypothesis and the whole the discussion proposed in this section.

5.3.4 Experimental Setup

The previous section focused on a qualitative empirical analysis of the applicability of EVT to the HPC domain. The goal of this subsection is to experimentally verify: (1) how different parallelization strategies affect the result; (2) the degree of fulfillment of the hypotheses of probabilistic real-time, by running appropriate statistical tests; (3) the

ID	Processing Unit	Multi-Node MPI	Stressers	Extended Data Size
1	CPU	N	N	N
2	CPU	N	Y	N
3	CPU	N	N	Y
4	GPU	N	N	N
5	GPU	N	Y	N
6	CPU	Y	N	N
7	CPU	Y	Y	N
8	CPU	Y	N	Y

Table 5.1: *Execution conditions of the benchmarks running during the experimental evaluation.*

tightness and usability of the estimated pWCET distributions. The following paragraphs are structured as follows: the first two parts describe the experimental setup and the analyzed metrics; the last two parts describe the results and discuss the applicability of probabilistic real-time techniques to HPC.

Benchmarks. For the tests, we used the NAS Parallel Benchmarks (NPB) suite, which is a very common choice for HPC performance estimation [16]. For each benchmark application different implementations are available: MPI, OpenMP and CUDA [76]. Among these, we selected the three pseudo-applications[7]: `bt`, `lu` and `sp`. This choice was based on two main reasons: first, the availability of stable and tested MPI, OpenMP and CUDA versions of the benchmarks, and second because the pseudo-applications better emulate real applications, rather than simple kernels triggering very specific hardware responses. The three benchmarks can leverage both intra-node and inter-node parallelism, by sharing data and synchronizing among OpenMP threads, CUDA kernels, and MPI processes. They also contain correctness checks that ensure the validity of the output and, consequently, the execution time. The time value is measured as *wall-clock*

[7]Please refer to the cited technical report for a detailed description of these benchmarks

time via the Time Stamp Counter register, with a theoretical resolution of 1ns[8].

Platform. We ran the benchmarks on the computing nodes of the GALILEO HPC cluster located at the CINECA supercomputing facility. Each node is equipped with a 2*18-core Intel Xeon E5-2697 v4 @ 2.30GHz, 128 GB RAM, and a Linux-based operating system. In addition, some nodes include an NVIDIA K80 GPUs for General-Purpose GPU computing. The nodes are connected via an InfiniBand Intel OmniPath (100Gb/s) multi-switch network. For each benchmark, we tested several different configurations, including CPU vs. GPU computing (CUDA framework), single-node vs. multi-node (MPI framework, running on 8 nodes for a total of 288 cores), the presence of stressers to introduce artificial interference (*stress-ng* program), and the different benchmark classes B or C. The execution conditions are summarized in Table 5.1. We refer to the C class of NPB benchmarks with the term *Extended Data Size*.

Execution parameters. For each execution condition, each benchmark was executed 500 times by using the default resource manager SLURM [260], which randomly selects the nodes according to the resource requirements selected by us for each experiment. This large number of executions made it possible to estimate the pWCET and run the statistical tests with a reasonable degree of confidence. Regarding the other parameters of the EVT process, we selected the Block-Maxima approaches with a block size of $B = 20$, coherent with previous works in probabilistic real-time. The traditional value of $\alpha = 0.05$ was chosen as the critical value for the statistical tests.

Methods and metrics. Defining the correct metrics to be used for comparison is crucial in an experimental evaluation based on a statistical framework. The metrics have been divided into three categories, to identify three different characteristics to explore: the variability, the adherence to EVT hypotheses, and the quality of the final pWCET outcome.

Methods and metrics – General statistical information. The first set of computed metrics comprises the traditional statistical measures: average and standard deviation. Instead of providing these direct absolute values that would have had little significant

[8]The accuracy and precision is definitely worse when measured in software. The measurement error can be in the order of hundreds of nanoseconds. In any case, conversely to many CPS applications, a lower resolution is acceptable because the HPC tasks duration is usually in terms of several minutes or hours.

informative content, we computed the *coefficient of variation* (CV) in order to make the comparison of different benchmarks and different execution conditions possible. This is because the CV is an indicator of the variability of the execution time, independently of its magnitude. The CV indicator is computed as follow:

$$CV = \frac{\sqrt{VAR[X]}}{E[X]}$$

Even if the CV value does not have a direct impact on the EVT hypotheses satisfaction, it gives us a way to compare, in a rough manner, the predictability of different benchmarks and computing platforms.

Methods and metrics – Compliance with EVT hypotheses. The second set is composed of five metrics representing different statistical tests to test the EVT hypotheses. In this experimental evaluation, we did not need to check the input representativity hypothesis: as previously discussed, its validity depends on the specific application, and the NPB benchmarks fulfill it by design. This because their inputs are already based on random uniformly distributed data. The iid hypothesis was checked by using three tests: the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test to verify the stationarity sub-hypothesis, the Brock-Dechert-Scheinkman (BDS) test for short-range independence and the ReScaled range (R/S) test for long-range dependence. Then, the Kolmogorov-Smirnov (KS) and the Anderson-Darling (AD) tests were used to check the MDA hypothesis.

Methods and metrics – Tightness of the pWCET distribution. The last set of metrics contains the three parameters of the estimated GEVD. While μ and σ strictly depend on the benchmark itself, ξ is extremely interesting for whatever concerns understanding the ability of the EVT process to obtain a distribution that can effectively be used in practice, as previously explained in Section 2.4.1. To make its effect clear to the reader, instead of providing the absolute values of μ, σ, and WCET, we computed the difference between the estimated WCET at violation probability $p = 10^{-6}$ and the Worst-Case Observed Time (WCOT), presenting to the reader the percent increment: INC $= 100\frac{\text{WCET}-\text{WCOT}}{\text{WCOT}}$. This value is not affected by the absolute values of the execution time of the particular benchmark, and it makes the comparison of different scenarios and benchmarks easy.

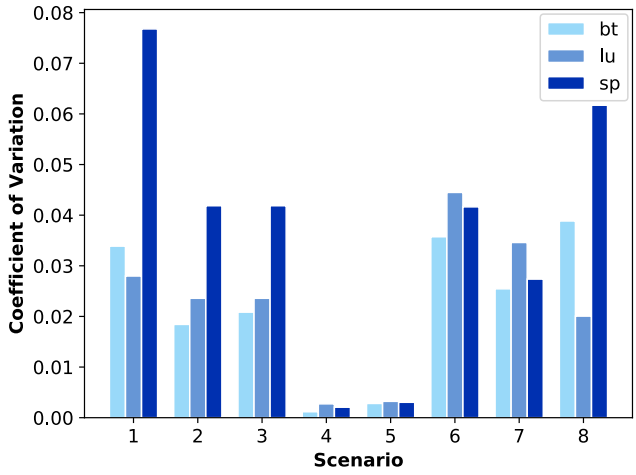


Figure 5.6: *The Coefficient of Variation for the three NAS benchmarks running with the different execution conditions of Table 5.1. These data are not affected by the absolute value of execution times.*

5.3.5 Experimental results

In the following discussion and the presentation of the data, we voluntarily omitted the absolute values of execution times, including average, variance, or maximum values. This because it is not our goal to perform an analysis of the platform and the benchmark itself. Instead, we are interested in their statistical metrics previously discussed. All the raw data of such experiments is made available online for any further analyses[9].

General statistical information

Figure 5.6 shows the Coefficient of Variation for the three benchmarks. We can easily notice that the GPU version of the benchmarks presented much less variability than the other ones. This result was expected, as we said, since GPU threads are usually less affected by external interference. Regarding the other scenarios, it is not possible to find a general trend. On average, network communications contributed to a larger variation, but this was not true for all the benchmarks, e.g., the case of the `sp`.

[9]Repository URL: `https://doi.org/10.5281/zenodo.3743352`

	Stationarity			Short-range dep.			Long-range dep.		
	KPSS			BDS			R/S		
CV	0.46149			1.9600			1.7470		
Sc.	bt	lu	sp	bt	lu	sp	bt	lu	sp
1	0.2726	0.8849	0.6410	2.0971	1.7994	1.5721	1.2790	1.8467	2.9726
2	0.1430	0.0226	0.1028	0.6964	2.2011	0.6857	1.3991	0.7438	1.2659
3	0.5082	0.0741	1.1961	1.9268	1.6072	3.0267	2.4314	1.09445	2.6035
4	0.1830	0.0603	0.1630	1.5743	0.9996	0.2335	1.0960	1.0960	1.1574
5	0.3406	0.0787	0.0952	0.5110	0.4240	0.4024	1.2927	0.9580	1.0843
6	0.0887	0.3647	2.4540	0.6279	2.3357	2.1319	1.1944	1.8665	2.9344
7	0.0455	0.0727	0.2126	1.1789	3.1038	2.4183	0.8945	1.2801	1.2332
8	0.5370	0.8509	0.3584	2.9611	1.3055	1.8211	1.9413	1.9291	1.1731

Table 5.2: *Result of the statistical tests used to check the i.i.d. EVT hypotheses, for each benchmark and for each execution condition of Table 5.1. A statistical test fails when the value of its statistic is greater than the critical value (cells in red).*

Finding 5.3.1. The execution time variability when GPU computational units are used is considerably lower than using CPU.

Compliance with EVT hypotheses

The adherence of the measured execution time datasets to EVT hypotheses is presented in Table 5.2 and Table 5.3. Starting from the stationary property, both the GPU scenarios (4 and 5) were able to pass the test and fulfill the stationary hypothesis with a good result (mostly all under 0.1 over a critical value of 0.46). We can also see that the presence of stresser tasks (2 and 7) had a positive effect on the stationarity hypothesis, making the CPU version also compliant. This is due to the fact that the interference caused by the stressers reduced the effect of spurious latencies. Previous works on cyber-physical systems showed similar results [111].

Finding 5.3.2. The stationarity property of EVT is satisfied for all benchmarks of GPU scenarios and when stresser tasks are present. It is found to be invalid for at least one benchmark in the other scenarios.

Looking at the raw data and resource manager logs, we observed how the network

	KS			AD		
CV	0.2716			variable (≈2.50)		
Sc.	bt	lu	sp	bt	lu	sp
1	0.1093	0.0608	0.0604	0.2575	0.2167	0.1525
2	0.1066	0.1102	0.0696	0.4767	0.4599	0.2650
3	0.1644	0.0683	0.2536	1.3099	0.3275	2.5158
4	0.1637	0.1594	0.0979	0.8369	0.9340	0.5791
5	0.1696	0.1179	0.0953	1.7366	1.4799	0.3255
6	0.1335	0.2090	0.2173	1.1849	2.5319	1.8093
7	0.0947	0.0883	0.0673	0.2984	0.5376	0.1543
8	0.1313	0.1665	0.0604	0.8118	1.2691	1.013

Table 5.3: *Result of the statistical tests used to check the MDA hypothesis, for each benchmark and for each execution condition of Table 5.1. A statistical test fails when the value of its statistic is greater than the critical value (cells in red).*

latencies affected the execution time, depending on the topological location of the node: the distribution of execution times when the jobs were scheduled on nodes under the same network switch was significantly different from the distribution of execution times when the jobs were scheduled on far nodes. This made the stationarity hypothesis harder to be achieved.

Finding 5.3.3. The stationarity property is negatively affected by a complex and hierarchical network topology.

The dependency tests, both short- and long-range, failed in several cases. The cause was mainly the resource manager choices: jobs were often spawned on the same nodes, thus causing the network storage locality to create a dependency. This locality was initially observed in GPU datasets (4 and 5) because the resource manager was configured with a strict policy that always provided the same node for the same user. After eliminating this limitation, the obtained results, shown in Table 5.2, are definitely compliant with the dependency hypothesis.

Finding 5.3.4. The two dependency properties are affected by the data locality of the SAN.

Finding 5.3.5. If data locality is removed, the GPU scenarios pass the dependency tests

	ξ			WCET/WCOT ratio		
Sc.	bt	lu	sp	bt	lu	sp
1	0.213	0.213	0.282	+162%	+62%	+549%
2	0.282	-0.065	0.249	+48%	+3%	+317%
3	0.287	-0.064	-0.481	+260%	+6%	+6%
4	-0.735	-0.421	-0.668	<0.1%	<1%	<0.1%
5	-0.213	-0.342	-0.0803	<0.1%	<0.1%	1%
6	0.584	0.577	0.562	+11994%	+12644%	+9336%
7	-0.169	0.365	-0.0180	+7%	+619%	+14%
8	0.574	0.371	0.885	+8873%	+991%	+57911%

Table 5.4: *The estimated pWCET for each benchmark and for each execution condition of Table 5.1. Only the ξ parameter and the WCET increment with respect to the WCOT are shown. The green cells indicate Weibull distributions, the red cells indicate a Fréchet distributions, and the yellow cells indicate Gumbel distributions.*

for all the benchmarks.

It should be noted that in any case, the scenarios that resulted in a statistical test reject, have low or moderate dependency (the value of the statistic is not too far from the critical value). It was then possible to estimate a correct distribution even in the presence of dependency, as shown by the GoF tests in Table 5.3. In fact, the resulting distribution is valid for any scenarios according to KS and failed in only one scenario (6/`sp`) with the AD test. The rejection of the case 6/`sp` is due to several possible reasons, including the MDA hypothesis being indeed false in this case. Another possibility is a false negative: in fact, the test level of significance was set to 5%, so it is possible that it was a spurious test failure. The non-rejection of most of the final distributions makes us confident that the estimation process was correctly executed and that the final pWCET distributions match the real data.

Finding 5.3.6. KS and AD tests do not reject any distribution (with one exception), corroborating the MDA hypothesis.

Tightness of the pWCET distribution. The pWCET distribution parameter ξ, estimated for the different scenarios, is shown in Table 5.4. Weibull distributions are highlighted by green cells ($\xi < 0$), near-Gumbel distribution with the yellow cells ($\xi \approx 0$), and Fréchet distributions with red cells ($\xi > 0$). The two GPU versions generated a clear Weibull distribution, which in turn provided a very tight WCET than the worst-case observed. In particular, the computed WCET, at the probability of violation $p = 10^{-6}$, for all the three benchmarks had a very low overestimation ($\leq 1\%$), that made the estimation very tight. Conversely, all the Fréchet distributions generated large values for WCETs, with the worst for scenario 8/`sp` that has about $+58000\%$ of overestimation. So large values would make the duties of the resource manager very difficult, since the estimated worst-case time is far from the observed ones and probably the real one. Finally, the two near-Gumbel distributions were able to overestimate the WCET by less than 10%, which is definitely acceptable.

Finding 5.3.7. GPU scenarios and a few CPU scenarios generated a Weibull distribution, which allows us to estimate a tight WCET. In the other scenarios, a Fréchet distribution has been estimated, making it difficult to obtain a tight and non-pessimistic WCET.

Discussion. From the experimental results we notice that there is no direct correlation between the three groups of metrics: for example, 8/`lu` has a lower CV than 3/`lu`, but 8/`lu` failed in both stationarity and R/S tests, while 3/`lu` is compliant with all the tests. The same scenarios show that the relation is not valid for the ξ metric as $\xi > 0$ for 8/`lu` and $\xi < 0$ for 3/`lu`, but this is the opposite for 4/`bt` and 1/`bt`.

The introduction of heterogeneous computing is winning for all the metrics: the GPU benchmarks presented the lowest variation, satisfied all the EVT hypotheses, and produced a tight estimation for the WCET, even in the presence of stressers. The stressers have beneficial effects on the CPU cases because they can mask the variation caused by external factors, such as the operating systems or different node configuration. They also improved the final distribution ξ value, even if not sufficient in all the cases (the overestimation in 2/`sp` and 7/`lu` is still very high).

The introduction of network communications did not provide a clear answer to the hypotheses' satisfaction: the specific benchmark and scenarios must be considered and tested. We noticed that in some cases, even if the short- and/or the long-range independence hypotheses are not satisfied, it is possible to obtain good pWCET estimations

(cases 2/`lu`, 3/`sp`, 7/`sp`). Some CPU benchmarks (3/`lu`, 7/`bt`) satisfied all the hypotheses and produced a tight pWCET. This result suggested that, in any case, proper statistical tests must be run during, before and after the estimation of the pWCET, to verify that the system is compliant with the EVT hypotheses. Besides the evident improvements of exploiting heterogeneous computing, no general conclusions can be drawn for CPU-only computing, and each case must be verified.

Table 5.5 summarizes, for each hypothesis, the qualitative discussions, the experimental method and the results obtained on the real platforms.

5.3.6 Lesson learned

The experimental results of the previous section showed us how the resources are allocated to HPC applications impact the ability to satisfy the EVT hypotheses and the importance of heterogeneous resources for probabilistic real-time. Assigning proper resources to the jobs is essential to estimate a correct pWCET and consequently guarantee the constraints of time-sensitive applications.

Job scheduling and resource management exploitation. In a heterogeneous scenario, the decisions of the resource allocation strategy, performed by the job scheduler/resource manager or by the operating system, play a key role in the WCET characterization of the applications, and, consequently, in the satisfaction of their timing requirements. It is possible to summarize the following guidelines for future resource management policies based on pWCET:

- the resource manager should avoid spawning jobs of the same applications on the same nodes in a row, to avoid the introduction of data locality and dependency due to the SAN network;
- the network topology can introduce significant latencies in node-to-node communications, and a non-uniform allocation of the jobs may produce different timing behaviors due to the interconnection hierarchy, and consequently invalidate the stationarity hypothesis of the pWCET;
- the heterogeneous non-CPU resources should be preferred because they are less affected by uncontrollable latencies;

- a promiscuous allocation of different jobs on the same node can be considered as masking the system-level (software and hardware) interference, provided that the average performance degradation is an acceptable price;
- an a priori evaluation of the pWCET hypotheses satisfaction is not easy in complex HPC systems, making mandatory the verification of the statistical properties of the timing measurements to ensure the correctness, reliability, and tightness of the pWCET distribution. A non-tight pWCET distribution makes resource management or job scheduling policies ineffective.

Conclusions. In HPC centers, resource management is a strategic activity to improve resource utilization and application performance satisfaction. This aspect becomes critical with emerging applications having timing requirements beyond the simple average QoS guarantees. In this section, we proposed to exploit the probabilistic real-time theory for applications running on an HPC infrastructure. We discussed the advantages and limitations of this analysis when applied to HPC and the possible benefits for heterogeneous platforms. The experimental evaluation showed the degree of applicability of probabilistic approaches to HPC configured in different execution conditions. The considerations of this work can lead to the development and study of resource management policies aiming to guarantee the mixed timing requirements of the applications.

Hypothesis		Qualitative assessment	Quantitative assessment	
			Method	Results
Representativity		It depends on the application itself and, in particular, on its input space.No a priori considerations can be made.	-	-
iid	Stationarity	Shared nodes, inhomogeneous hardware across the cluster, thermal and power management strategies, and network topology may hinder the satisfaction of this hypothesis.	KPSS	The use of GPU processing units make this hypothesis true, because it limits the effect of operating system thermal/power management strategy. In the other cases the hypothesis is often invalid, with the exception of the case when stressers are present.
	Short-range dependence	These two hypotheses can be made invalid by the network topology and data locality in the SAN. Conversely to CPS scenario, these hypotheses are unlikely to be influenced by processor cache locality.	BDS	This hypothesis appears true only when GPU computational units are used, while it is false for at least one benchmark in all the other cases.
	Long-range dependence		R/S	This hypothesis appears true only when GPU computational units or stressers are involved..
Maximum Domain of Attraction		Being a pure statistical property with no direct correlation with hardware or software, it is not easy to draw any qualitative conclusion.	KS AD	The MDA hypothesis, verified with both tests, is substantially true for all the scenarios and benchmarks. The only exception is the 6/`lu` case. It can be a false positive, a consequence of the failure of the dependence tests, or the actual MDA hypothesis being invalid. Also for this hypothesis, GPU scenarios satisfy this hypothesis.

Table 5.5: *Summary of the qualitative and quantitative analysis performed related to the satisfaction of EVT hypotheses.*

CHAPTER 6

Beyond Timing Constraints

The probabilistic real-time theory and the EVT can be exploited not only for the execution time but also for other metrics. This chapter shows how the EVT and the previous analyses can be applied for energy estimation in embedded systems. This can be very useful in a certain class of applications because modeling the power/energy of a system is usually very challenging.

6.1 Introduction to the Worst-Case Energy Problem

The increasing complexity of the computing architectures described in Chapter 1 makes not only the WCET problem challenging, but it also hinders the development of accurate power and thermal models for the processors and other computational resources. Moreover, without a proper characterization of the workload, such models usually lead

to unrealistic results [122]. Similar to the timing problem, the difficulties in the power model estimation for both hardware and software characteristics make the problem of estimating the *Worst-Case Energy Consumption (WCEC)* hard. The WCEC value is extremely valuable for some applications, for instance, the example later described in Section 6.3.

6.1.1 Related works

To the best of our knowledge, the first work on WCEC analysis was presented in 2006 by Jayaseelan et al. [137]. The worst-case energy estimation was based on static code analysis and energy models at the micro-architectural level, similar to the SPTA approach for the WCET. More recently, the *0g* tool [251] presented two approaches to WCEC estimation: a static code analysis and a measurement-based technique. The former was used to characterize the energy consumption of critical tasks, while the latter was used to estimate the consumption in non-critical tasks, due to the unreliability of the results. Similar to the previous approach, the employment of the static code analysis requires the availability of the platform-specific per-instruction characterization of energy consumption. The same authors proposed recently static WCEC system-wide methods, including the analysis of peripherals [252]. In 2017, a work by Pallister et al. [199] proposed an estimation of the WCEC, at instruction-level, performed by fitting a Weibull distribution. However, selecting a single-class distribution and the focus on instruction-level limit the applicability to the accurate and safe WCEC estimation of the whole system. Finally, Morse et al. [190] presented an analysis of the limitations of the current WCEC analysis techniques, demonstrating that it is an NP-hard problem and no efficient approximation algorithms exist. The authors suggested the possibility of moving towards statistical methods, explicitly citing the EVT, that is exactly the cornerstone of our approach proposed in Section 6.2. Before describing the application of EVT to the energy metrics, we define in the next subsection the models related to the energy-constrained problem.

6.1.2 Task and system model

This section presents the task and the system model used in this chapter to deal with energy requirements. A set of n periodic tasks is identified as: $T = \{\tau_1, \tau_2, ..., \tau_n\}$. Each task activation is called *job* and it is represented by the notation $J_{i,j}$, to iden-

tify the j-th job of the i-th task. Each task is defined with the following tuple: $\tau_i = (C_i, D_i, T_i, E_i, L_i)$, where C_i is the worst-case execution time[1], D_i is the task relative deadline, T_i is the period (or the minimal inter-arrival time in case of sporadic tasks), E_i is the WCEC, and L_i is the criticality level.

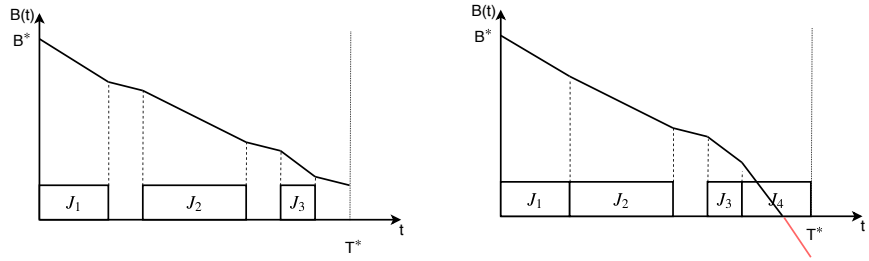
The criticality level can assume different types of values. In this subsection (and similar to the mixed-criticality approach of Section 5.2), we use two levels labeled `HI` and `LO`, to respectively identify *high-criticality tasks*, for which job's execution must be guaranteed, and *low-criticality tasks*, for which the job's execution can be guaranteed only if enough energy budget is actually available. If a job is not executed due to lack of energy, we say that this task is *dropped*; conversely, we say that the task is *admitted* to the scheduling queue.

Energy-constrained system model. Following the same approach proposed by Völp et al. [250], the system model is characterized by two parameters: the energy budget B^* and the survival period T^*. This notation denotes that the system is allowed to consume a maximum amount B^* of energy in the continuous period of time T^*. In order to make the subsequent theoretical discussion easier, we define the *budget function* $B(t)$ as the total amount of remaining energy budget as a function of time, with $B(0) = B^*$. A couple of examples of the budget function are depicted in Figure 6.1. This function decreases at least at the rate of the power consumption in the idle state. We use $E_{\texttt{IDLE}}$ to indicate the energy consumption of the system in idle for the whole timespan $(0, T^*)$. Consequently, the inequality $B(0) - B(T^*) > E_{\texttt{IDLE}}$ holds regardless of any job admission or scheduling decision.

Possible use cases of the presented energy-constrained mixed-criticality models include several battery-powered applications, such as devices powered by unreliable energy harvesting sources, mobile devices, and space applications [84].

Scheduling model. The *High-Level Schedule (HLS)* is defined as the set $S = \{S_{1,j}, S_{2,j}, ..., S_{n,j}\}$ where $S_{i,j}$ is a boolean value indicating if the job $J_{i,j}$ can be scheduled or not. Therefore, $S_{i,j} = 1$ means that $J_{i,j}$ is allowed to run, while $S_{i,j} = 0$ indicates that the job is dropped. The traditional notion of *schedule*, including the task execution ordering, is out of the scope of this work and it can be performed by any state-of-the-art scheduler having the S set as input.

[1] In traditional mixed-time criticality task models C_i is a set of WCETs, one for each criticality level. However, in this chapter we consider only one WCET.



(a) *Energy-feasible scenario*

(b) *Non energy-feasible scenario*

Figure 6.1: *Budget function $B(t)$ examples varying the number of tasks in the survival period T^*.*

Definition 6.1.1. An HLS S is said to be *energy-feasible* if the energy budget is sufficient to run all the jobs, thus $B(T*) \geq 0$.

Definition 6.1.2. An HLS S is said to be *correct* if and only if all the `HI`-crit tasks are scheduled: $S_{k,j} = 1 \quad \forall \tau_k$ s.t. $L_k =$`HI`.

These definitions are the basis for the first schedulability requirement, called *minimal energy-schedulability condition*:

Lemma 6.1.1. *An energy-feasible and correct HLS exists only if:*

$$\sum_{k \text{ s.t. } L_k = HI} \sum_{j} E_{k,j} < B(0) - E_{IDLE} \tag{6.1}$$

where $E_{k,j}$ is the energy consumed by the j-th job of the k-th `HI`-crit task. This energy does not include the static energy consumed by the system in idle E_{IDLE}. □

Lemma 6.1.1 states that the energy budget must be sufficient to guarantee at least the execution of the `HI`-crit tasks, otherwise no energy-feasible and correct HLS exists. Since the energy of the single job $E_{i,j}$ is hard to predict, we rely on the upper-bound represented by the WCEC of the task E_i. Unfortunately, as already discussed in Section 1, estimating E_i in modern architectures running complex workload is either not trivial or leads to very over-approximated results. Moreover, the introduction of the WCEC value in Lemma 6.1.1 would mean to consider an overly pessimistic scenario, in which all the jobs consume the worst-case amount of energy, that seldom occurs on real systems.

6.2 Probabilistic Worst-Case Energy

In the following paragraphs, we show how to determine a reliable not underestimated WCEC value (E_i) by exploiting the same probabilistic measurement-based approach for WCET described in the Chapter 2 and used by the other chapters of this thesis. A measurement-based approach comes with important advantages:

1. The system is considered as a black-box (accurate platform models are not required).

2. No need to perform in-depth static analyses of the task source code.

3. The output is a *worst-case* estimation (and not a *mean-case* estimation) within a given level of confidence, that represents the probability of failure to meet the energy budget requirement B^*.

While point (2) enables the possibility to estimate the WCEC for previously unknown tasks, point (3) is a fundamental requirement for mission-critical systems.

6.2.1 Formal definition

Let us assume we measured several times the energy consumed for executing the jobs of a specific task. The energy samples $x_1, x_2, ..., x_n$ represent the realization of the input random variables of the Glivenko-Cantelli theorem (described in Section 2.1.1). These values can be fed into the previously described EVT process to obtain a GEVD representing the tail of the distribution, i.e., the probability of extreme events. We call this function *probabilistic-WCEC (pWCEC)*. Through this distribution, it is possible to exploit its iccdf to obtain an estimation of the WCEC, given a violation probability level p.

The pWCEC model. By exploiting the EVT, it is possible to compute the two random variables $\mathcal{E}_i$ and $\mathcal{E}_{\texttt{IDLE}}$, respectively representing the worst-case value of E_i and $E_{\texttt{IDLE}}$. Thanks to this measurement-based approach, we can verify the condition of Lemma 6.1.1 without having accurate models of system and workload. The introduction of random variables complicates the application of the schedulability condition of Lemma 6.1.1. To combine those random variables, two possible approaches can be considered:

1. Setting a value for the probability of failure p and use the iccdf of $\mathcal{E}_i$ to obtain scalar worst-case values for E_i.

2. Performing the actual sum of the random variables.

The first option is the simplest one, but it probably leads to excessive overestimation since we always consider the worst-case value for all the jobs. The second option instead requires the application of the convolution operator previously described in Section 2.1.2. In this work, we consider the second option, since it allows us to obtain a tight estimation of the real WCEC. Solving the integral of the convolution operator is hard to obtain analytically, especially for GEVD, but easy to obtain using one of the several numerical algorithms available in the literature. Regarding the $\mathcal{E}_{\texttt{IDLE}}$ value, we simply compute the worst-case value at the predefined probability failure p, i.e. $\bar{E}^p_{\texttt{IDLE}} = \text{iccdf}_{\mathcal{E}_{\texttt{IDLE}}}(p)$.

By using the probabilistic model just defined, the previous Lemma 6.1.1 can be rewritten as:

Lemma 6.2.1. *Given the set $\{\mathcal{E}^*_k\}$ of random variables defined as the following convolution ($\circledast$):*

$$\mathcal{E}^*_k = \overset{nr\_jobs}{\circledast} \mathcal{E}_k \qquad \forall k \text{ s.t. } \tau_k \in T \tag{6.2}$$

an energy-feasible and correct HLS, with a violation probability p exists only if:

$$\sum_{k \text{ s.t. } L_k = HI} \bar{E}^p_k < B(0) - \bar{E}^p_{IDLE} \tag{6.3}$$

where $\bar{E}^p_k$ is the WCEC value of all the jobs of the k-th task, with violation probability p, computed by using the convoluted random variables: $\bar{E}^p_k := iccdf_{\mathcal{E}^*_k}(p)$. The term $\bar{E}^p_{IDLE}$ represents instead the WCEC estimation in idle mode. □

If the number of tasks is extremely high, the use of another convolution operator replacing the sum operator in Equation (6.3) should be considered to reduce the over-approximation. For convenience, we indicate the left-hand term of the Equation (6.3), i.e., the WCEC of all the HI-crit tasks, with $\bar{E}_{\texttt{HI}}$.

6.2.2 Energy-aware job admission protocol

In this subsection, we focus on the job admission problem, i.e., to whether a job can enter the scheduling queue or not. In traditional mixed-criticality systems, the admission of HI-crit jobs must be guaranteed in any case. Consequently, we need to define a

policy for the admission of LO-crit tasks, based on the energy required by the tasks and the system energy budget available. We can derive the energy budget for LO-crit tasks from Lemma 6.2.1, as follows:

$$B^{\text{LO}}(t) = B(t) - \bar{E}_{\text{HI}}. \tag{6.4}$$

Accordingly, if there exists an energy-feasible and correct HLS, then it is always $B^{\text{LO}}(t) \geq 0$. Following the same probabilistic approach of HI-crit tasks, we can state the *maximum energy consumption condition* for LO-crit tasks as:

Lemma 6.2.2. *Given a failure probability p and the survival period T^*, LO-crit tasks energy consumption upper bound is:*

$$\sum_{k \text{ s.t. } L_k = LO} \bar{E}_k(p) < B^{LO}(0) - \bar{E}^p_{IDLE} \tag{6.5}$$

We refer to the left-hand part of Equation (6.5) as $\bar{E}_{\text{LO}}$, deriving the overall remaining unused energy budget as follows:

$$B(T^*) = B(0) - \bar{E}_{\text{HI}} - \bar{E}_{\text{LO}} - \bar{E}_{\text{IDLE}} \tag{6.6}$$

The *job admission algorithm* assigns the values to the set $S_{k,j}$, in accordance with the aforementioned conditions. Relying on the previous WCEC estimations, this section aims at proposing a policy that minimizes $B(T^*)$, while maintaining a fair energy distribution among LO-crit tasks. An algorithm carrying out the minimal positive value of $B(T^*)$, among the values carried out by all the algorithms, is said to be *optimal*.

The job admission pseudo-code is shown in Algorithm 6.2. The policy considers the scheduling of HI-crit tasks first, followed by the LO-crit ones. The algorithm sets $S_{i,j} = 1$ for all the jobs of the HI-crit tasks (lines 10-12), so that it complies with the HLS correctness requirement (Definition 6.1.2). The remaining energy budget B is reduced by the WCEC value estimated for the scheduled task with violation probability p (line 13). If $B < 0$, no energy-feasible and correct HLS exists (line 13).

Once all the HI-crit tasks are admitted for scheduling, we can compute the energy budget for LO-crit tasks as in Equation (6.4). In this version of the algorithm, we apply a fair policy by distributing the same budget over all the tasks (line 20). In case a task τ_i does not use the entire budget assigned (lines 21-25), all the jobs are admitted. Conversely, if the budget is not sufficient to execute all the jobs (lines 25-30), then the

maximum number of allowed jobs must be computed (line 26). The convolution operator must then be re-applied to get the new energy consumption estimation (lines 27-28). In both cases, the overall LO-crit budget is decremented by the WCEC estimation of the task, $\bar{E}_i^p$ (line 31), thus preserving the unused energy for other tasks. It is worth mentioning that, as it requires the exploration of a large number of possible convolutions to compute the random variable in Equation (6.2), the function $max\_jobs(\tau_i, B_i)$ is computationally intensive. In particular, since state-of-the-art convolution algorithms have a complexity of $O(n \log(n))$ and the exploration is performed at most for n times, then the overall worst-case complexity of the Algorithm 1 for n tasks is upper-bounded by $O(n \cdot M \cdot \log(M))$, where M is the total number of jobs. Once the number of jobs to schedule is computed, a proper selection policy must be applied to get the $S_{i,j}$ assignment (line 29). This policy depends on the specific scenario and application requirements. A couple of possible trivial approaches are: (1) to select only the first *nr_jobs* tasks or (2) to generate a uniformly distributed assignment. In the subsequent experimental evaluation, we applied the latter approach. Finally, $B^{NULL} = B(T^*)$ is the remaining not yet assigned energy. Sorting the tasks by energy consumption (line 18) guarantees that no dropped job exists with WCEC lower than $B(T^*)$: the algorithm is *optimal*.

Algorithm 6.2 Mixed-Energy Criticality Job Admission Protocol. (Original Source: [213])

Input: τ_i, $\bar{E}_k^p$, $\bar{E}_{\text{IDLE}}^p$, B^*, T^*, p.
Output: $S_{k,j}$ (the HLS).
procedure SCHEDULE
    $B \leftarrow B^* - \bar{E}_{\text{IDLE}}^p$
    $B^{LO} \leftarrow$ ScheduleHITasks(B)
    $B^{NULL} \leftarrow$ ScheduleLOTasks(B^{LO})
end procedure
procedure SCHEDULEHITASKS(B)
    for all k s.t. $L_k = HI$ **do**
        for all j job of task k **do**
            $S_{k,j} \leftarrow 1$
        end for
        $B \leftarrow B - \bar{E}_k(p)$; assert($B > 0$)
    end for
    return B
end procedure
procedure SCHEDULELOTASKS(B)
    $T \leftarrow$ sort $\{\tau_i : L_i = LO\}$ by $\bar{E}_i^p$ ascending order; $n \leftarrow$ size(T)
    for all $\tau_i \in T$ **do**
        $B_i \leftarrow \frac{B}{n}$
        if $\bar{E}_i^p \leq B_i$ **then**
            for all job j in τ_i **do**
                $S_{i,j} \leftarrow 1$
            end for
        else
            nr_jobs $\leftarrow$ max_jobs(τ_i, B_i)
            $\mathcal{E}_i^* \leftarrow \circledast_j^{\text{nr\_jobs}} \mathcal{E}_i$
            $\bar{E}_i^p \leftarrow \text{iccdf}_{\mathcal{E}_i^*}(p)$
            $S_{i,\{1,2,\ldots,m\}} \leftarrow$ selection_policy(τ_i, nr_jobs)
        end if
        $B \leftarrow B - \bar{E}_i^p$; $n \leftarrow n - 1$
    end for
    return B
end procedure

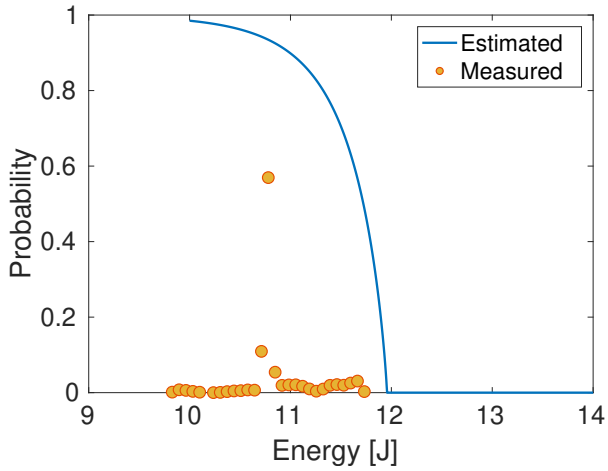


Figure 6.2: *The WCEC estimation validation. The solid line represents our worst-case model, while the dots indicate the frequency of the energy consumption observed probability. (Original Source: [213])*

6.2.3 Experimental validation

For the experimental validation, we performed two classes of tests. The first one aimed at verifying (1) the upper-bound of the energy consumption estimation of the single job execution of each task, and (2) how the convolution operator helps in carrying out tight but still safe estimations. In the second one, instead, we evaluated the job admission algorithm, checking its ability to guarantee the energy budget constraint while keeping a tight estimation of the energy consumption.

Experimental setup. The selected hardware platform was an Odroid XU-3 equipped with a `big.LITTLE` CPU, featuring 4 Cortex-A7 and 4 Cortex-A15 cores. The selection of such a complex architecture was made to show the validity and effectiveness of probabilistic approaches in the black-box estimation of energy consumption. This platform would have been really tough to be analyzed with traditional model-based approaches. The tasks under analysis were pinned onto the `big` cores, while the scheduler and the energy measuring task onto the `LITTLE` ones. The *Dynamic Voltage and Frequency Scaling (DVFS)* was disabled, forcing the frequency of big cores to be 1.8 GHz. This prevents the unwanted thermal throttling effect that would also improve the repro-

ducibility of the experiments. The pWCEC estimation is theoretically agnostic w.r.t. the presence of a DVFS mechanism. However, further dedicated studies are required to assess the validity of the i.i.d. EVT condition when DVFS is enabled. The energy consumption we refer to in this subsection is measured for the `big` cores only, where the tasks under analysis were running, by exploiting on-chip power sensors.

The considered workload was made up of 4 multi-threaded applications from the RODINIA benchmark suite [52] (`lavaMD` (`la`), `streamcluster` (`st`), `leukocyte` (`le`), and `particlefilter` (`pa`)) characterized by the same order of magnitude of execution times. The Linux operating system was equipped with the PREEMPT_RT patch, in order to maximize the software determinism and increase the experiment reproducibility.

Energy estimation upper bounding. The first set of experiments focused on the estimation of the WCEC and the verification of the performance of the convolution operation from Equation (6.2). First, we ran 500 jobs of each task to get the $X_1, X_2, ..., X_n$ measures necessary to perform the EVT estimation. The i.i.d. hypothesis of the samples was verified by using the Ljung-Box statistical test. Then, the distribution estimation was performed. For instance, the pWCEC of the `st` benchmark is the distribution $GEV(11.596025, 0.425034, -1.178425)$, with its cumulative distribution function depicted in Figure 6.2. By setting failure probability value to $p = 10^{-9}$, the corresponding $WCEC$ is $11.9567J$. In other words, the probability of observing a job consuming more than $11.9567J$ is 10^{-9} or lower. To empirically check the estimation's quality, we acquired other $10\,000$ samples of `st` job energy consumption. The observed probability was computed from these samples and depicted in Figure 6.2. The GEVD safely over-estimates the real energy consumption, as expected by the theory.

Once we estimated the *pWCEC* for each task, we applied the convolution operator to the interval $j = [1; 100]$ to estimate the energy consumption of the sequences of jobs of size $1, 2, ..., 100$, considering a violation probability $p = 10^{-9}$. In a real scenario, the violation probability depends on the criticality of the application. This estimation is shown in Figure 6.3 (dashed lines) with respect to the number of jobs j. We verified the results by running 100 jobs of each task and measuring cumulative energy consumption. The real energy consumption is depicted in Figure 6.3 with solid lines. As it might be seen, the convolution operator produced tight estimations (the maximum overestimation value is 7%) and no underestimations. By running the same experiment, while using

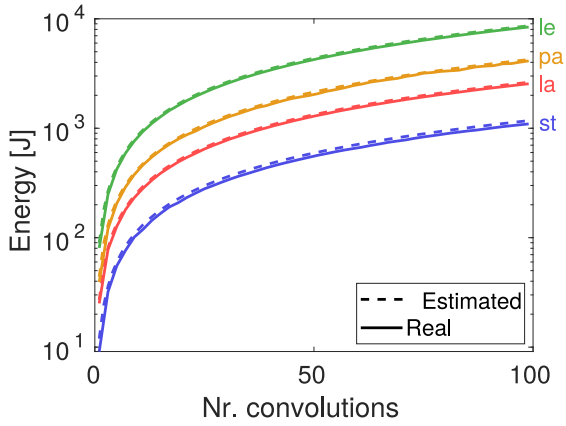


Figure 6.3: *The energy estimated and measured varying the number of convolutions for the four benchmarks considered. (Original Source: [213])*

Scenario	Energy budget B^*	Survival period T^*
(1)	$100\,000\,J$	$10\,h = 36\,000\,s$
(2)	$50\,000\,J$	$7\,h = 25\,200\,s$
(3)	$8\,000\,J$	$1\,h = 3\,600\,s$

Table 6.1: *Experimental setup by considering three scenarios with different energy budget and survival period. (Original Source: [213])*

the mean energy value instead of the worst-case provided by EVT, we found that 24% of the cumulative scenarios of `pa` were underestimated and thus unsafe.

Job scheduling and energy budget. From the previous results on the WCEC per-job estimations, we used the proposed job admission/scheduling algorithm in three different scenarios. The benchmarks `la` and `pa` were considered as `HI`-crit tasks, while `le` and `st` represented two `LO`-crit tasks. The scheduling periods were configured as follows: $T_{\texttt{la}} = 30s$, $T_{\texttt{pa}} = 100s$, $T_{\texttt{le}} = 100s$, $T_{\texttt{st}} = 60s$. The four tasks were scheduled, considering the following three scenarios and a high level of confidence ($p = 10^{-9}$), with the budgets presented in Table 6.1.

The proposed algorithm generated HLSs with the number and percentage of admit-

Scenario	la (HI)	pa (HI)	le (LO)	st (LO)
(1)	1200 (100%)	360 (100%)	290 (81%)	600 (100%)
(2)	840 (100%)	252 (100%)	15 (6%)	110 (26%)
(3)	120 (100%)	36 (100%)	7 (19.4%)	52 (86.7%)

Table 6.2: *Schedulability result for each benchmark and scenario. (Original Source: [213])*

Scenario	Theoretical Energy	Real Energy	Overestim.
(1)	99 982.3 J	96 858.9 J	3.23 %
(2)	49 988.0 J	48 206.8 J	3.70 %
(3)	7 996.3 J	7 739.6 J	3.32 %

Table 6.3: *Energy estimation and results. (Original Source: [213])*

ted jobs indicated in Table 6.2. As expected, the HI-crit jobs are all executed, while some LO-crit jobs are rejected to meet the energy budget requirements. The difference between the percentage of allocated jobs between le and st is due to the energy-fair scheduling: the le energy consumption is much higher than st. We executed these three scenarios using the computed HLS and the energy results presented in Table 6.3 were observed.

In the previous table, the *theoretical energy* is the energy budget subtracted by the unused energy (i.e., $B^* - B(T^*)$) while *real energy* is the energy measured during the whole HLS execution. As expected by the theoretical guarantees, the proposed approach has never under-estimated the real energy in the three considered scenarios. Simultaneously, the estimation was maintained very tight, showing an over-estimation in the $3-4\%$ range compared to $20-30\%$ of the previously cited state-of-the-art tools.

6.2.4 Lesson learned

The computation of the pWCEC with the EVT provides an important tool for task scheduling. This characterization becomes essential when modeling and implementing energy-harvesting systems, as described in the following section. The use of a mixed-

criticality workload increases the interest in this topic for future works, which may also include timing constraints.

6.3 Resource Management for Energy-Harvesting Systems

The use of Internet of Things (IoT) devices is increasing in many domains, including industry, automation, communication, medicine, and transportation. Several applications in these domains require such devices to be small and constrained with respect to the energy and power usage while maintaining sufficient performance levels. These constraints are also because many IoT devices run on batteries, thus having a limited amount of energy and power. This problem is even exacerbated when dealing with energy harvesting systems, i.e., systems relying on an electrical power source self-harvested from the environment, such as a wind turbine or a solar panel. The constraint on the energy budget availability, in these devices, is given by the limited battery capacity and the future quantity of available input power, which cannot be easily predicted. In fact, the external power sources are usually unreliable since they are heavily dependent on weather conditions, yielding to high volatility in the exploitable energy budget. Energy harvesting systems are usually auto-sufficient, i.e., they are designed to guarantee the highest uptime possible with the least human intervention possible.

6.3.1 A motivational use case: a transponder tracking system

The transponder is an on-board device of an aircraft that periodically broadcasts its position together with some flight information, e.g., altitude, direction, etc. The ADS-B standard is widely used by almost all commercial airplanes for airborne collision avoidance or tracking purposes. This system is widely known with the name of its commercial implementation: Traffic Collision and Avoidance System (TCAS). ADS-B signals are also received on the ground by institutional operators (such as airport control towers) and amateurs. However, an ADS-B transponder is usually too expensive and too much power-hungry for gliders and ultralight aircraft. In these cases, a preferable choice is to use less common technologies, enabling a simpler anti-collision system and ground tracking. The tracking of such an aircraft from the ground requires to build a dedicated network of receivers because neither the government nor the amateur network for ADS-B can be exploited. For example, flying clubs face the necessity to track their small aircraft, without having the financial capability to build a network of re-

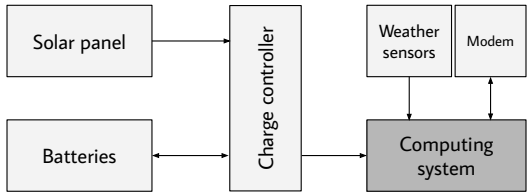


Figure 6.4: *Block diagram of the hardware architecture. (Original Source: [59])*

ceivers. The solution currently under investigation in Lombardy (Italy) is to place a few receivers on some mountain tops to cover the largest possible area with the minimum number of devices. Such devices in these remote areas limit the possibility of using the electrical grid as a power source. Consequently, the devices are equipped with a battery and a solar panel to harvest energy. The energy source is then non-reliable for the already explained reasons. We need a resource management strategy able to guarantee a good trade-off between energy consumption and performance by knowing the weather current information and forecast.

6.3.2 System architecture

In Figure 6.4, we sketched the hardware structure of the embedded platform our work focuses on. The core is represented by a computing system, featuring a networking device (usually a 4G modem) for remote control and data transmission, and a dedicated set of "weather sensors". The energy budget is defined by the power provided by a solar panel and a battery pack. A charge controller is then responsible for recharging the batteries and regulate the power supply for the computing platform. The energy budget is characterized by a certain degree of variability, given by the weather conditions, which, in turn, affect the solar panel output, and by the current charge level of the batteries. On the software hand, the running tasks and applications may reasonably use a variable amount of resources (CPU, memory, etc.) over time, which increases the variability of the overall system's power consumption.

6.3.3 Energy budget prediction and management

To estimate the budget, we used two sources of information: a remote weather forecast and a local weather station composed of a set of sensors (which provide temperature,

humidity, visible light, and barometric pressure). More specifically, we want to explore the interplay of weather forecast information coming from a remote service on the Web and the data provided by weather sensors, locally connected to the system. This is motivated by the fact that the weather forecast services typically provide coarse-grained information, both from the spatial (city) and the temporal perspective (one prediction per hour), although obtained through accurate models and observations. Instead, the weather sensors allow the system to 1) gather data at high rate and 2) capture extremely local conditions, like, for example, transient clouds covering the sky and decreasing the solar irradiance for a few minutes.

When solar panels are used to harvest energy, the problem of modeling the input power as a function of weather forecast is split into two parts (also in accordance with previous works [234]):

1. Finding the model that links weather data to solar irradiance.

2. Finding the model that links solar irradiance to solar panel's output power.

The first model is needed when the information source is the local weather station. We need, in fact, to infer the solar irradiance (SI) from the visible light value (L) by using the following formula:

$$SI\left[\frac{W}{m^2}\right] = 0.0079 \cdot L[lx] \tag{6.7}$$

Instead, the remaining sensors – temperature, humidity, and pressure – allow us to create the *precipitation potential* as model feature, as proposed by a previous work [187]. The features are then fed to a *Support Vector Machine* to train a machine learning model.

The second model is the solar panel's output power, which strongly depends on the Sun's incidence angle on the solar panel. Formulas to compute such value exist, but they are affected by the non-ideality of the sonar panel, by its installation, and other parameters challenging to control *a priori*. The considered features are the day of the year, the current time, and the solar irradiance SI. A *Support Vector Machine* is also used to train this second model.

The details of the machine learning model are omitted in this manuscript, and the reader can refer to the original paper [59]. We reported the experimental evaluation result in Table 6.4, performed by running a real system connected to a solar panel, a

Model	% of SV	SCC
Solar irradiance	18.76	0.968
Power generation	28.64	0.857

Table 6.4: *Confidence results on the Support Vector Machine model. (Original Source: [59])*

local weather station, and a remote weather service. The results are expressed in terms of percentage of support vectors (% of SV) and *Squared Correlation Coefficient* (SCC). It is possible to notice a low number of SV, which shows we are not underfitting the data, and a good high value for SCC, which means the correlation between input and output of the model is good.

6.3.4 A proposed energy budget model for future works

In this subsection, we improve the previous model for the energy budget in energy-harvesting systems, based on pWCEC. The model is currently used in ongoing works and, consequently, no experimental evaluation is available yet. The model is presented in the following paragraphs as a reference for possible future works.

The main goal of this model is to make the system survive in a given time period $[0; T^*]$ by considering both input and output power. For instance, in the previous example of Section 6.3.1, the time period $[0; 24h]$ is a reasonable interval. Let $P^{\text{IN}}(t)$ the input power from the energy-harvesting device (e.g. the solar panel) at a generic time t. The input power is used to charge an energy storage device (e.g., a battery), which has a finite maximum capacity, identified by $\bar{E}^{\text{store-max}}$. The input budget function can be then written as:

$$B(t) = \min\left(\bar{E}^{\text{store-max}}, B(0) + E^{\text{IN}}(t) - E^{\text{OUT}}(t)\right) \tag{6.8}$$

where $B(0) \geq 0$ is the energy already available in the storage at $t = 0$, and $E^{\text{IN}}(t)$ is the total input energy harvested during the period $[0; t]$:

$$E^{\text{IN}}(t) = \int_0^t P^{\text{IN}}(x)dx \tag{6.9}$$

and the consumed energy $E^{\texttt{OUT}}(t)$ is computed in ideal conditions as follows:

$$\begin{aligned} E^{\texttt{OUT}}(t) &= \int_0^t P^{\texttt{OUT}}(x)dx = \int_0^t \left(P_{\texttt{IDLE}} + P^{\text{tasks}}(x)\right) dx \\ &= P_{\texttt{IDLE}} \cdot t + \int_0^t P^{\text{tasks}}(x)dx \end{aligned} \tag{6.10}$$

where $P_{\texttt{IDLE}}$ is the minimum power consumption when the system is idle, and P^{tasks} is the power consumption to execute the set of tasks.

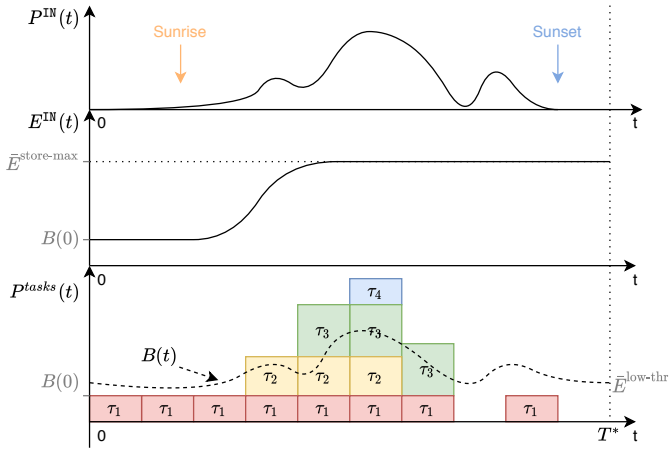
The optimization problem has to goal to maximize the power used by the task, that implicitly, in first approximation, improves the total throughput. It can be written as follows:

$$\begin{aligned} &\max \int_0^{T^*} P^{\text{tasks}}(x)dx \\ &\text{s.t. } B(t) \geq \bar{E}^{\text{low-thr}} \qquad\qquad \forall t \in [0; T^*] \end{aligned} \tag{6.11}$$

where $\bar{E}^{\text{low-thr}} \geq 0$ is the minimum amount of energy that must be maintained in the storage. This value is ideally 0, but in practical applications, a minimum amount of energy must be maintained to keep the system online and run the next allocation by the resource manager. To optimize the problem of Equation (6.11) we need to discretized the integral. This is needed for two reasons: 1) computing an analytical form for $P^{\text{tasks}}(x)$ is difficult for the power modeling issues discussed at the beginning of this section, and 2) we can not perform a continuous allocation of tasks. For these reasons, we consider a time period $T^{alloc} < T^*$ that corresponds to a single allocation decision. This value must be $T^{alloc} > \max_j C_j$, for a reason that will be clear later on. This means that in our target survival timespan T^*, we have a total of $\left\lceil \frac{T^*}{T^{alloc}} \right\rceil$ decisions to take. We can use this interval to approximate and split the maximum problem with a sum:

$$\int_0^t P^{\text{tasks}}(x)dx \leq \sum_{i=1}^{\lceil \frac{t}{T^{alloc}} \rceil} \int_{(i-1)T^{alloc}}^{iT^{alloc}} P^{\text{tasks}}(x)dx \tag{6.12}$$

Then, the integral can be written by exploiting a similar concept of HLS described in Section 6.1.2. We define the symbol S_j^i as the allocation decision for task τ_j in the i-th time period, i.e. if the jobs of task τ_j are admitted or not to schedule in the time period $[(i-1)T^{alloc}; iT^{alloc}]$. With this symbol, the last integral can be replaced with



	S_j^1	S_j^2	S_j^3	S_j^4	S_j^5	S_j^6	S_j^7	S_j^8	S_j^9	S_j^{10}
S_1^i	1	1	1	1	1	1	1	0	1	0
S_2^i	0	0	0	1	1	1	0	0	0	0
S_3^i	0	0	0	0	1	1	1	0	0	0
S_4^i	0	0	0	0	0	1	0	0	0	0

Figure 6.5: *An example of the proposed model. For illustrative purpose only, axes not in scale.*

a function f such that $\int_{(i-1)T^{alloc}}^{iT^{alloc}} P^{\text{tasks}}(x)dx = f(i, T^{alloc})$ defined as:

$$f(i, T^{alloc}) = \sum_{j:S_j^i=1} N_j \cdot F_j^{-1}(\bar{p}) \tag{6.13}$$

where F^{-1} is the icdf of the pWCEC of the task τ_j, N_j is the number of jobs of τ_j in a time period of length T^{alloc}. In this way, the optimization algorithm consists of only selecting S_j^i for each allocation period.

An illustrative and comprehensive example of the symbols defined in this model is depicted in Figure 6.5.

6.3.5 Lesson learned

This proposed model is part of ongoing activity to build a better resource management policy in energy-constrained systems, conceivably with a real-time workload. We are currently studying algorithms to build the allocation matrix S_i^j, by considering mixed-criticality workload similar to the model previously proposed in Section 5.2. The proposed model has been developed in ideal conditions; future works may take into account the non-ideality of the energy storage or the charger to improve the result accuracy.

CHAPTER 7

Developed Tools and Datasets

7.1 The `chronovise` Tool

The software `chronovise` [212][1] has been developed to support the experimental evaluation of this thesis. This framework is open-source and aims at standardizing the flow of the MBPTA process, integrating both estimation and testing phases needed to carry out a valid pWCET.

This is not the first software developed for MBPTA. In 2010, Lu et al. [173] proposed an algorithm called `RapidRT` to estimate via Block-Maxima the pWCET, but it is unclear if the software has been released or not. The authors stated that part of the software is based on a proprietary library, and, consequently, we suspect it is not acces-

[1] Last version could be found in the following git repository: `https://github.com/federeghe/chronovise`

sible. In 2015, Lesage et al. [161] proposed a framework for MBPTA, but it is uncertain if the software has been made available to the public and is usable for generic applications or only to the specific case described in the paper. The two open-source tools we are aware of are `MBPTA-CV` [3] and `diagXtrm` [109][2]. The first software has been developed for the particular algorithm MBPTA-CV, but it suffers from the issue described in Section 3.2, and it provides the estimation part only. Instead, the `diagXtrm` tool is more oriented to a user-friendly visualization tool, without integrating all the features presented in this thesis for the EVT process. It is possible to state that `diagXtrm` is complementary to `chronovise` , and a future integration is possible.

The `chronovise` software has been developed as a library written in C++, which can be linked with any other program. The choice of C++ required to re-implement several statistical libraries already available in other languages (e.g., Python or MATLAB), but it guarantees the maximum speed of the algorithms. This choice has two advantages: (1) the software can be easily converted to HPC programming models (e.g., MPI) if a considerable computational power is needed to carry out the solution; (2) it enables the possibility to run the algorithms online, i.e., during the execution of the system as a sort of monitoring activity. This monitoring of the tasks is one of the possible future works, as explained in Section 8.2.

7.1.1 Components and features

The software can be compiled with a C++14 compliant compiler and by using the *CMake* tool. Even if tested only on Linux, `chronovise` works potentially on any operating system because it does not use any specific function of Linux. Optionally, a test suite could be run by exploiting the Google Test tool. The test suite's availability and the continuous integration guarantee that the results remain correct even when new versions of the software are pushed in the repository. The codebase is divided into five different areas:

1. The top-level code which implements the EVT process flow.

2. The EVT components, i.e., the BM, PoT, and MBPTA-CV algorithms, together with the GEVD and GPD distributions.

[2]The original repository of `diagXtrm` was at this url: `https://forge.onera.fr/projects/diagxtrm/`, however it is currently not accessible at the time of writing.

3. The statistical tools, in particular the estimators (MLE, PWM) and a set of statistical tests (AD, BDS, CvM, KPSS, KS, Ljung-Box, RS).

4. Input generators, to generate input values to be provided to the program according to a given distribution.

5. Some utility functions, e.g. to read or write to a `.csv` file.

The framework is built as a C++ static library that can be linked to the user implementation. The user implementation must include a derivative class that implements the abstract class methods provided by the framework to configure and adapt the framework to the user-specific requirements.

In order to simplify some common uses of the framework, two applications based on the `chronovise` library has been developed and distributed with the framework: the algorithm to compute the PPI (described in Section 4.1) and a pWCET estimation process based on BM (thus producing a GEVD).

The `chronovise` tool was used to carry out the results of most of the analyses presented in this thesis.

7.2 Statistical Power Estimation and Dataset

The statistical power of the GoF tests is a critical parameter to estimate the result's uncertainty of MBPTA methods, as we explained in Section 3.3 and Section 3.4. The analytical calculation of the statistical power is unfortunately not easy: for most of the GoF tests, a closed-form expression does not even exist. In Section 3.3.3, we already described why this estimation is necessary to properly select the sample size for testing procedures. We created a novel dataset for statistical power estimation for GoF tests in extreme distribution, which can be advantageous for several fields, not only for probabilistic real-time; in fact, the selection of the sample size is often performed with empirical procedures and where the results are often interpreted in a too optimistic view [165].

Among the previous works on statistical power estimation, Heo et al. [126] estimated the critical values and statistical power for AD and MAD tests by using a Montecarlo approach for GoF tests of EVT distributions. The critical values were computed for a scenario where the model parameters to be tested were estimated from the same data used for the test. This scenario is commonly referred to as *case 3*, i.e., the assumed

distribution parameters are unknown. Instead, the *case 0* scenario (also called *external validation*) is when the samples used to perform the test are a different set with respect to the samples used to estimate the reference distribution. In particular, to the best of our knowledge, quantitative information of only *case 3* scenarios is available in the literature because the traditional applications of EVT have little data to deal with, while probabilistic real-time can gather thousands of samples with a low effort. However, to the best of our knowledge, no *case 0* power analysis for such distribution classes is available in the literature. Generally, statistical power estimations for *case 0* are not representative of *case 3* and vice versa. The dataset proposed in this section wants exactly to fill this knowledge gap. The *case 0* is also more powerful that *case 3*, leading to the most stringent and unbiased test [102].

7.2.1 Estimation method

This work presents the estimated statistical power of three Goodness-of-Fit (GoF) tests: Kolmogorov-Smirnov (KS), Anderson-Darling (AD), and Modified Anderson Darling (MAD) described in Section 3.3.2. Other common tests have been excluded, for example, the Chi-Squared (CS) and Cramer-von Mises (CvM) tests, because state-of-the-art works already showed that they have a lower statistical power compared to KS or AD [261, 10].

The analytical computation of the statistical power, and consequently the selection of the appropriate sample size for testing, is usually not possible due to the frequent lack of the *effect size knowledge*, i.e., the real characterization of the population's distribution from which the samples are collected. To estimate it, Muthén and Muthén [191] studied the usage of Monte Carlo methods to select the sample size and determine the testing power. We follow the same approach to estimate the statistical power of our scenarios, identified by a set of tuples representing the test conditions. In particular, the Monte Carlo sampling is executed for every tuple $(D, n, \alpha, \mathcal{G}_1, \mathcal{G}_2)$, where D is the statistic of the test under analysis, n is the sample size, α the level of significance, $\mathcal{G}_1, \mathcal{G}_2$ are respectively the reference distribution with cumulative distribution function $F(x)$ and the empirical distribution with cumulative distribution function $F_n(x)$. The statistics and critical values for KS, AD, and MAD tests are computed by using the formulas of Appendix B.

The pseudo-code of the analysis is presented in Algorithm 7.3. For each scenario, the critical value is computed (line 4), and a large number of explorations N is per-

Algorithm 7.3 Power estimation algorithm with Monte Carlo simulations. (Original Source: [217])

1: **Input**: N(number iterations), D (test statistic), n (sample size), α (significance level), $\mathcal{G}_0, \mathcal{G}_1$ (null and alternative distributions)
2: **Output**: W (test power)
3: reject = not_reject = 0
4: critical_value = get_critical_value(D, $\mathcal{G}_0$, n, α)
5: **for** $i \in [1; N]$ **do**
6: X = collect_sample($\mathcal{G}_1$, n)
7: **if** $D(F_{\mathcal{G}_0}(\cdot), X) >$ critical_value **then**
8: reject++
9: **else**
10: not_reject++
11: **end if**
12: **end for**
13: $W =$ reject / (reject + not_reject)

formed (lines 5-12). Each time, we draw a sample from the reference distribution (line 6), and we check if the statistic D of the ecdf matches or not with the drawn sample, comparing it with the critical value (line 7). If the statistic value is higher than the critical value, then the sample is rejected (line 8), otherwise not (line 10). Finally, the ratio rejection over total samples provides us the statistical power W (line 13). If the test is able to detect the differences between $\mathscr{G}_1$ and $\mathscr{G}_2$, we expect to get a value near 1 for this ratio. In this specific Monte Carlo simulation, the standard error of W is computed as [262]:

$$\sqrt{\frac{R(N-R)}{N^3}} \tag{7.1}$$

where $R \leq N$ is the number of rejects (the accumulation variable of line 12).

The selected values for parameters of each Monte Carlo estimation are:

- $N = 10^9$: number of Monte Carlo iterations;
- D: the test statistics previously described;

- n: the sample size. Exploring all the possible values would have increased in a non-sustainable way the computational effort required by the Monte Carlo simulations. Since the power test function is a non-decreasing function of n, we explored them easily selecting the following values: $n = (50, 100, 150, 200, 300, 400, 500, 750, 1\,000, 2\,500)$;
- α: the significance level. We studied the traditional values of 0.05 and 0.01.

The simulations ran on 4 nodes of the CINECA supercomputing facility (GALILEO-A1 cluster, 2 x Intel Xeon E5-2697v4@2.3GHz per node) for a total of 144 CPU cores. It took $\approx$ 13h for KS tests, $\approx$ 17.5h for AD test, $\approx$ 16h for the MAD test.

7.2.2 Results

In the following pages, Table 7.1, Table 7.2, and Table 7.3 present the results of the Monte Carlo estimation, by showing the statistical power obtained for each scenario considered. These tables can be very useful as a reference for future works on probabilistic real-time, but also in other scientific fields that use the EVT and the related tests. Each table is composed of three macro rows indicating the reference distribution ($\mathscr{G}_1$) and the actual distribution from where the samples have been generated ($\mathscr{G}_2$). The results in Table 7.1 show that the KS test is not able to reach the target probability ($1 - 10^{-9}$) even while using $1\,000$ samples, while with $2\,500$ samples the test is always correct. AD and MAD are much more sensible as shown in Table 7.2 and Table 7.3: both of them needs only 750 samples to reach the target probability. No significant difference between AD and MAD has been observed.

$\mathscr{G}_1$	$\mathscr{G}_2$	α	Sample size (n)									
			50	100	150	200	300	400	500	750	1000	2500
$GEV(0,1,0)$	$N(0,1)$	0.05	0.433100925	0.883347765	0.991603951	0.999615010	0.999999874	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.173221448	0.643452223	0.926426259	0.990668281	0.999969746	0.999999977	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.402221446	0.827773062	0.975704624	0.997320621	0.999988383	0.999999963	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.164124225	0.581214439	0.872639669	0.972233002	0.999500289	0.999996110	0.999999985	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.286787074	0.754349990	0.962845802	0.996346524	0.999991250	0.999999992	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.092007617	0.442246778	0.782790629	0.944778594	0.999292443	0.999997245	0.999999994	1.000000000	1.000000000	1.000000000
	$GEV(0,1,-0.5)$	0.05	0.061924052	0.847865820	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.005506233	0.173621553	0.914148452	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0.5)$	0.05	0.109873147	0.293153608	0.549437525	0.740308878	0.943703112	0.991959149	0.999253784	0.999999586	1.000000000	1.000000000
		0.01	0.029933482	0.121708898	0.280459902	0.438755733	0.781235667	0.939418508	0.988237286	0.999932671	0.999999921	1.000000000
$GEV(0,1,0.5)$	$N(0,1)$	0.05	0.869488165	0.999999315	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.454885837	0.998873700	0.999999998	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.766801312	0.999765759	0.999999991	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.359259013	0.983640502	0.999988015	0.999999997	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.367774702	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.139806008	0.744566527	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,-0.5)$	0.05	0.987657414	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.632639500	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0)$	0.05	0.032299787	0.231060818	0.576355557	0.852314165	0.995633829	0.999963685	0.999999914	1.000000000	1.000000000	1.000000000
		0.01	0.003184289	0.034069354	0.173820625	0.443183696	0.892633450	0.993613818	0.999882386	1.000000000	1.000000000	1.000000000
$GEV(0,1,-0.5)$	$N(0,1)$	0.05	0.284370451	0.629365918	0.862809163	0.953616572	0.996003172	0.999709102	0.999984209	0.999999990	1.000000000	1.000000000
		0.01	0.115952998	0.409541419	0.678458448	0.854765234	0.979383833	0.997758518	0.999791031	0.999999723	1.000000000	1.000000000
	$t(10)$	0.05	0.283091343	0.616658436	0.853660417	0.948713438	0.995402308	0.999673102	0.999984703	0.999999999	1.000000000	1.000000000
		0.01	0.116120515	0.399945876	0.664716864	0.844156028	0.976087763	0.997213016	0.999726448	0.999999688	0.999999999	1.000000000
	$U(-2,3)$	0.05	0.826074656	0.998836102	0.999998678	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.564242941	0.981799855	0.999913887	0.999999780	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0.5)$	0.05	0.726868690	0.993646689	0.999951322	0.999999872	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.466562747	0.954538684	0.999232900	0.999988813	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0)$	0.05	0.200916378	0.658118197	0.907265288	0.983782467	0.999851447	0.999999511	0.999999999	1.000000000	1.000000000	1.000000000
		0.01	0.062618618	0.341107470	0.726574304	0.899661698	0.997334290	0.999960032	0.999999740	1.000000000	1.000000000	1.000000000

Table 7.1: *Statistical powers of Kolmogorov-Smirnov (KS) test. (Original Source: [217])*

$\mathscr{G}_1$	$\mathscr{G}_2$	α	Sample size (n)									
			50	100	150	200	300	400	500	750	1000	2500
$GEV(0,1,0)$	$N(0,1)$	0.05	0.898883879	0.997852804	0.999984066	0.999999942	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.718376305	0.980296191	0.999486072	0.999993816	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.900608464	0.996296207	0.999925665	0.999999076	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.748572022	0.977738108	0.998990518	0.999971307	0.999999992	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.879039216	0.996647326	0.999962734	0.999999807	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.637742192	0.966816380	0.998871526	0.999980674	0.999999999	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,-0.5)$	0.05	0.506193505	0.988842459	0.999999819	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.156587927	0.642128743	0.986360566	0.999993819	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0.5)$	0.05	0.670656084	0.922479505	0.988032793	0.998722304	0.999993686	0.999999989	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.459281936	0.781070668	0.936797739	0.986617002	0.999711112	0.999997335	0.999999992	1.000000000	1.000000000	1.000000000
$GEV(0,1,0.5)$	$N(0,1)$	0.05	0.999748036	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.995080617	0.999999963	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.999535626	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.996014553	0.999999766	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.998987729	0.999999982	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.994605043	0.999997776	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,-0.5)$	0.05	0.999824564	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.950973931	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0)$	0.05	0.496070245	0.864390928	0.985947394	0.999396436	0.999999874	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.268355771	0.570821100	0.843710853	0.968703009	0.999817934	0.999999875	1.000000000	1.000000000	1.000000000	1.000000000
$GEV(0,1,-0.5)$	$N(0,1)$	0.05	0.883425766	0.990317893	0.999349700	0.999962533	0.999999898	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.806435001	0.975341606	0.997569908	0.999802265	0.999999137	0.999999998	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.953012799	0.998489273	0.999961459	0.999999177	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.916712052	0.995729267	0.999835355	0.999994869	0.999999994	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.999998030	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.999994727	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0.5)$	0.05	0.999997678	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.999996758	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0)$	0.05	0.998992782	0.999998975	0.999999999	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.998880701	0.999998793	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000

Table 7.2: *Statistical powers of Anderson-Darling (AD) test. (Original Source: [217])*

$\mathscr{G}_1$	$\mathscr{G}_2$	α	Sample size (n)									
			50	100	150	200	300	400	500	750	1000	2500
$GEV(0,1,0)$	$N(0,1)$	0.05	0.867320643	0.997898948	0.999990214	0.999999977	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.588673674	0.973245243	0.999478267	0.999995867	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.781941641	0.985645303	0.999484369	0.999987393	0.999999994	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.491534388	0.919403834	0.993532836	0.999678517	0.999999668	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.435062367	0.842316752	0.981577615	0.999015458	0.999999594	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.165284527	0.485703819	0.811172107	0.962024230	0.999674018	0.999999555	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,-0.5)$	0.05	0.645973895	0.999995654	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.109046892	0.906790997	0.999997798	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0.5)$	0.05	0.791838371	0.961675361	0.994629272	0.999381894	0.999994651	0.999999961	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.631781428	0.893099447	0.976268016	0.995786651	0.999913894	0.999998846	0.999999990	1.000000000	1.000000000	1.000000000
$GEV(0,1,0.5)$	$N(0,1)$	0.05	0.999736838	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.965085315	0.999999991	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.996004168	0.999999998	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.914097219	0.999992894	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.559739086	0.999999974	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.085883847	0.848460957	0.999999946	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,-0.5)$	0.05	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.948498429	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0)$	0.05	0.269985358	0.803816020	0.984632476	0.999564876	0.999999972	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.045445362	0.315514515	0.745273839	0.958954776	0.999873523	0.999999965	1.000000000	1.000000000	1.000000000	1.000000000
$GEV(0,1,-0.5)$	$N(0,1)$	0.05	0.861794911	0.987006580	0.999023454	0.999937794	0.999999803	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.774748242	0.966673086	0.996200975	0.999643427	0.999997890	0.999999994	1.000000000	1.000000000	1.000000000	1.000000000
	$t(10)$	0.05	0.938597184	0.997569575	0.999925765	0.999998185	0.999999999	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.895361187	0.993189832	0.999670695	0.999987242	0.999999992	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$U(-2,3)$	0.05	0.999996144	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.999991056	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0.5)$	0.05	0.999997506	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.999996636	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
	$GEV(0,1,0)$	0.05	0.999028457	0.999998993	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000
		0.01	0.998893433	0.999998740	0.999999995	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000	1.000000000

Table 7.3: *Statistical powers of Modified Anderson-Darling (MAD) test. (Original Source: [217])*

7.3 Benchmarks

In many scientific works, researchers use benchmark suites to carry out the experimental results by running them on real platforms or via simulation. Regarding the WCET, the following three free and open-source suites (written in C) have been extensively used in research articles regarding real-time systems in embedded scenario:

- **Mälardalen WCET** [118]: Probably the most famous benchmark suite for WCET applications, it is a collection of research and small industrial use cases. Most of the included programs are simple algorithms, such as array ordering, basic math operations, or synthetic programming structure to test specific behaviors.

- **TACLeBench** [88]: Similarly to the previous, the TACLeBench is a collection of benchmarks, which have been rewritten to match a defined coding style and to remove the dependencies on external libraries and operating system. The suite is composed of 53 benchmarks, including synthetic and pseudo-applications.

- **MiBench** [119]: Inspired to a commercial benchmark suite, it is a mix of different algorithms, organized in the following application categories: Industrial and Control, Consumer, Office, Network, Security, and Telecommunication.

These three benchmark suites are publicly available, stable, and widely used in research. However, they have some issues, explained in the following paragraph, which make them unsuitable for testing measurement-based methods and, in particular, the MBPTA methods.

7.3.1 Current limitations and solutions

Some of the programs, especially of the Mälardalen WCET suite, come from previously existent software, often written with deprecated C standards. In addition, many benchmarks contain debug code, OS-specific libraries, different programming techniques, and non-configurable constants. All these factors increase the necessary effort for the scientist to use such benchmark suites, increasing the possibility of committing mistakes, and reducing the time spent in research activities. Another unwanted effect we noticed is due to the compiler optimizations. Many modern compilers perform smart optimizations that reduce the executed code by computing as much as possible at com-

pile time. In some corner cases, we observed the compiler even replacing an entire complex algorithm with the solution, if all the input data is known at compile-time.

All the programs of the previous benchmark suites use constant values for inputs, mainly because of the necessity to remain independent from external software. This can be acceptable for static analyses, but not for measurement-based, for the compiler optimization problem just described and the necessity to observe different execution behavior at run-time. For this reason, we created a new benchmark suite by taking the programs from the aforementioned suites and applying the following modifications:

- Getting rid of the deprecated C syntax (e.g., as the K&R constructs) and porting it to C99 standard.
- Code cleaning to uniform the code style, remove the unnecessary constants and primitives used for debugging.
- Adding an external, user-provided, way to measure the execution time, to allow the developer to use external tool instead of the C library `<time.h>`.
- Removing the dependency from external libraries and architecture-specific instructions; when possible, the dependency on the C library was removed.
- Replacing the hard-coded inputs with randomly generated inputs by a PNRG, according to the uniform distribution or, for some cases, to a distribution selected by the user.

The detailed description of the modifications and the preliminary results are explained in the recent master's thesis of Confalonieri [55], which is the exploratory report of the currently ongoing scientific work. A preliminary experimental evaluation showed how the original benchmarks meet the EVT hypotheses, according to the PPI index described in Section 4.1, for the 50% of the benchmarks, while with the introduced modifications, this percentage increases to 81%.

7.3.2 The necessity of input-independent software

The obtained result of the satisfaction of the PPI hypotheses for 81% of the benchmarks does not represent a realistic situation. To show this, we implemented two complex algorithms used in real applications: the TCAS[3] and an audio compression benchmark.

[3]The Traffic Collision Avoidance System is a system used on-board of aircraft to detect imminent collisions and suggest the best collision avoidance manoeuvres to pilots

Both of them obtained excellent results when fed by random inputs, obtaining the satisfaction of the hypotheses for all the considered time traces. However, when we tried to use real data – coming from, respectively, a flight simulator and real audio files – the PPI values and the EVT hypotheses' satisfaction drastically dropped, making the obtained pWCET unsafe and possibly underestimated. This result perfectly exemplifies the problem of representativity we discussed in Section 4.2. All these preliminary results are available in the previously mentioned master's thesis [55].

Currently, we are developing a solution to this problem based on randomized algorithms. This solution, already described in Section 4.2.4, makes the algorithm timing independent from the input, and consequently compliant with EVT hypotheses. For example, we obtained preliminary results with the bubble sort algorithm, showing that sorting an array and its flipped version in a row and removing some optimizations make the execution time independent from the values in the array itself. Clearly, the average execution time got worse (up to 2.5x), but it enables the possibility to compute the pWCET. Similar approaches are available from the cryptography and security worlds: to avoid side-channel attacks, some cryptographic algorithms are built to exhibit an execution time independent from the input data and the key, to avoid any leakage of information. Such techniques and novel techniques can be studied in future works to generalize input-independent algorithms and remove this problematic aspect of representativity.

CHAPTER 8

Future Works and Conclusions

As the last chapter of this thesis, we present the possible future works and research trends related to timing-sensitive systems and, particularly, on probabilistic real-time computing, which is the main subject of this manuscript. Finally, the conclusion section summarizes the achievements and the research advances obtained by the works presented in this thesis.

8.1 Trends in Real-Time Systems

As the industry is moving towards more complex architectures due to the end of Dennard's scaling and the imminent end of Moore's law [189], traditional approaches to estimate the WCET are becoming difficult to be effectively used. Probabilistic real-time solutions are promising for future approaches, as later discussed in Section 8.2. The

current inability to safely estimate a tight WCET has led the researchers to focus on the MC framework (described in Section 5.2.1) in the last decade. MC scheduling is still a very hot topic in real-time research. A good candidate in this regard for future works is exploiting probabilistic real-time information to improve the schedulability, or other non-functional metrics, in MC context. In this thesis, we proposed an example (Section 5.2), which used probabilistic information to optimize energy consumption. Many derivative works are possible, from the moving towards a dynamic-priority scheduler to multi-criticality levels. The interest in energy optimization is also critical for energy-constrained embedded systems, as we described in Section 6.3. In addition to energy, future research can pursue other goals:

- Using the probabilistic information (pWCEC or pET) to improve the schedulability of `LO`-criticality tasks, while maintaining the formal guarantees for `HI`-criticality tasks. More than two criticality levels can also be considered to exploit the probabilistic information more thoroughly.
- Optimizing other metrics, such as reliability and power consumption, which interest is rapidly increasing in modern architectures due to components' miniaturization.
- Dealing with fault-tolerance requirements by allowing tasks to re-execute if a failure occurs. Probabilistic information can be exploited to verify the probability of transient faults to happen and perform probabilistic scheduling on the task re-execution.

The last possible future research is becoming remarkably important in some safety-critical applications, such as embedded systems for aviation and space. The reason is the miniaturization of the components, which makes the electronic devices also more susceptible to transient faults. The development of software techniques for such scenarios can improve reliability while not requiring the usual triple redundancy setup or a heavy radiation shield, which would increase the costs, space, and weight of the final system.

8.2 The Future of Probabilistic Real-Time

Probabilistic real-time and, in particular, MBPTA are a controversial topic in the real-time community. They are attractive from an industrial standpoint, thanks to the sim-

plicity of the black-box approach, which hides the software and hardware complexity. Whether such approaches can be certified in the future is an open question, due to the numerous open challenges [139]. The most crucial challenge for the success of probabilistic real-time is, most probably, the representativity hypothesis. In Section 4.2.4, we have already identified the major obstacles and the possible future research directions regarding this condition. Further theoretical studies are required, supported by the development of novel architectures compliant with the representativity hypothesis.

EVT process choices (e.g., block size selection), uncertainties of estimators, and statistical power of tests are other sources of errors that must be quantified to obtain a safe pWCET. This thesis presented some theoretical tools to deal with and estimate such uncertainties. Further investigations are needed to close the circle on this uncertainty quantification. For example, the choice of the block-size for BM (or the threshold value for PoT) were only partially and empirically investigated by previous work, and a more theoretical approach is needed.

Even if the pWCET cannot be considered safe for WCET estimation, it could be useful as a monitoring tool. The estimated distribution and/or the statistical test outputs can be used to monitor the system's health from the timing standpoint and recognize any unexpected change before an extreme event occurs. For example, a run-time monitor may continuously check the goodness-of-fit of the pWCET distribution with the timing samples, detecting, in this way, changes to the timing behavior. The monitoring can be exploited in different manners, e.g., reacting to maintain soft real-time requirements or identifying security threats. Preliminary studies, not presented in this thesis, suggest that pWCET is a promising solution as a change detection algorithm for the timing behaviors of applications.

The first results in using pWCET estimation techniques for non-embedded systems, such as HPC, as we showed in Section 5.3, are encouraging, especially when dealing with heterogeneous architectures. Future works are needed to assess the role of each subcomponent (e.g., network, SAN, etc.) in the satisfaction or not of the i.i.d. hypotheses and, consequently, in the accuracy of the pWCET estimation. The use of the Linux PREEMPT_RT patch described in Section 5.1 can help in pursing the accuracy goals. How to use the pWCET information not only for timing requirements but also to improve the job and resource management in such systems is another interesting topic to study in-depth.

Other approaches exploiting pWCET not directly to compute the WCET can be

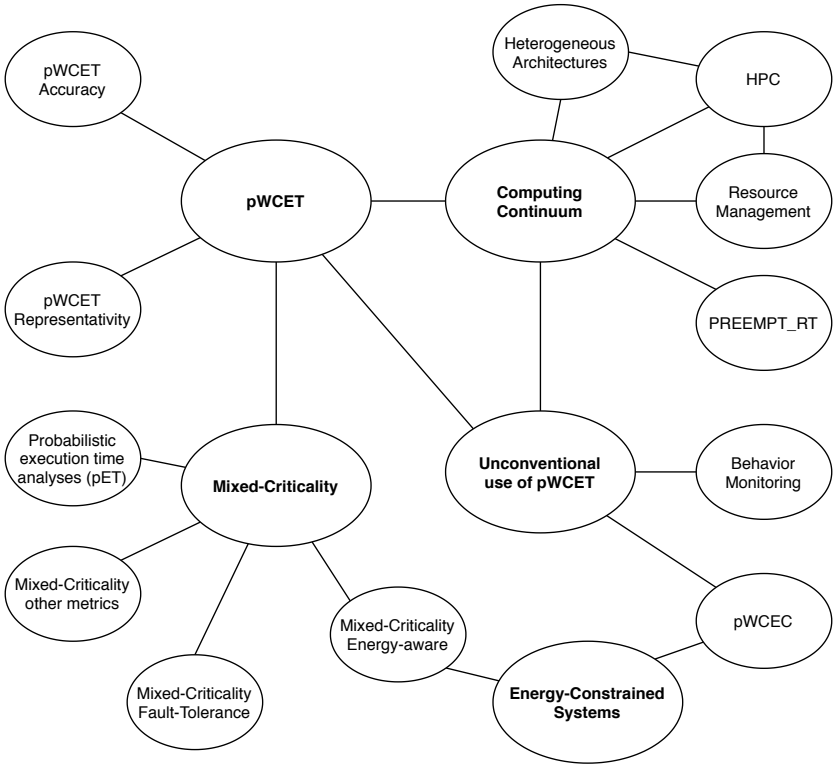


Figure 8.1: *Conceptual map of the possible topic for future works.*

the subject of future studies. The model for energy-constrained systems described in Section 6.3.4, together with the pWCEC concept of Section 6.2, can trigger many future works. Conversely to time requirements, the energy requirement is rarely required to be formally proved, making the probabilistic theory very interesting. Different quantities can also be explored, such as maximum power consumption, temperatures of the cores, reliability, etc.

All the keywords related to these proposed research directions for future works are summarized in the conceptual map of Figure 8.1.

8.3 Conclusions

This manuscript reported the scientific works published during the Ph.D. course of the author. It began by explaining the difficulties in estimating the WCET on modern architectures with traditional static methods, which push the research to find alternative solutions. One of the state-of-the-art solutions is probabilistic real-time computing, born about 20 years ago and developed in the last decade, which is the central topic of this thesis. The main advances in terms of improvement of the scientific knowledge are outlined with the following list:

1. Uncertainties analysis for the pWCET estimation process, by focusing on different aspects of it. We proved a common misconception in previous works regarding the distribution upper-bounding (Section 3.2), which may lead to optimistic and unsafe pWCET estimations; the concept of statistical power and its impact on the pWCET uncertainties (Section 3.3) were explained; we developed a mathematical tool, called Region of Acceptance, to explore the area of uncertainties on the GEVD parameters (Section 3.4).

2. The most crucial hypothesis, i.e., representativity – which is currently the main barrier to the use of probabilistic real-time for industrial systems – was discussed and possible solutions proposed (Section 4.2).

3. Evaluation of the EVT hypotheses satisfaction on real platforms. To perform such an analysis, we developed a new statistical test called PPI (Section 4.1), which was exploited to test embedded platforms (Section 4.1.2), including a Linux PREEMPT_RT and a complex HPC platform (Section 5.3). The latter scenario was also studied with heterogeneous architectures, identifying the benefits of heterogeneity in probabilistic real-time hypotheses satisfaction.

4. The probabilistic theory was applied to the energy problem to estimate the pWCEC instead of the pWCET. The pWCEC was then experimentally validated on an embedded platform (Section 6.2). This metric can also be used in the context of energy-constrained and energy-harvesting systems (Section 6.3), for which a task and system models were proposed for future works.

5. On the same energy aspect, we proposed a probabilistic approach to energy-aware mixed-criticality scheduling (Section 5.2), such that the hard real-time constraints

are met in any condition by using a static WCET, while the energy-optimization is based on the pWCET evaluation.

6. To perform the analyses of the previous points, we developed an open-source software called `chronovise` (Section 7.1), we estimated the statistical power with a high degree of confidence (Section 7.2), and, finally, we started to develop a custom WCET benchmark suite targeted to probabilistic approaches (Section 7.3), which will be the subject of a forthcoming future work.

In summary, thanks to the research activity of this thesis, we improved the knowledge of the uncertainties affecting the pWCET and how to quantify and deal with them. We explored the major obstacles to certification by using both theoretical reasoning and experimental evaluations. Possible future solutions to these obstacles were discussed, including their potentials and limits. In addition, unconventional uses of the pWCET were proposed and tested.

The scientific community is far from reaching a consensus on the future of probabilistic techniques for WCET estimation in safety-critical systems. Many challenges are still open, creating significant interest in the research community in the last decade. During these recent years, the research articles, including the works presented in this thesis, improved the knowledge on the pWCET estimation. However, the theoretical setup is still unfledged and requires consistent effort to make it mature. For this reason, many future works arise, as shown in the previous Section 8.2, forecasting a great ferment on this topic for the next years.

APPENDIX A

List of Publications

In this appendix we provide the list of scientific articles resulting from the research conducted during the PhD and presented in this thesis. Each paper is summarized with the following structure:

Title			
Authors			
Publication venue (Journal/Proceedings/etc.)			
Year	Bibliographic info	DOI (if any)	Cit.
Reference to thesis chapters or sections			

Section A.1 lists the main papers related to this thesis and presented in the previous chapters. Unless otherwise stated, the author of this thesis is the main contributor to these papers for both the theoretical and experimental parts. Section A.2, instead, presents the articles related to the works which are subject of this thesis, but present only in minimal part in this thesis. The author of this thesis provided a significant contribution to these manuscripts. Finally, Section A.3 lists all the manuscripts currently under review.

A.1 Main papers

Timing Predictability in High-Performance Computing with Probabilistic Real-Time			
F. Reghenzani, G. Massari, W. Fornaciari			
IEEE Access			
2020	vol. 8	10.1109/ACCESS.2020.3038559	[221]
Section 5.3			

Optimizing Energy in Non-preemptive Mixed-Criticality Scheduling by Exploiting Probabilistic Information			
A.A. Bhuiyan, F. Reghenzani*, W. Fornaciari, Z. Guo (*equal contribution)*			
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems			
2020	vol. 9 no. 11	10.1109/TCAD.2020.3012231	[29]
Section 5.2			

Dealing with Uncertainty in pWCET Estimations			
F. Reghenzani, L. Santinelli, W. Fornaciari			
ACM Transactions on Embedded Computing Systems			
2020	vol. 19 no. 5	10.1145/3396234	[222]
Section 3.4			

Probabilistic-WCET Reliability: Statistical Testing of EVT hypotheses			
F. Reghenzani, G. Massari, W. Fornaciari			
Microprocessors and Microsystems, Elsevier			
2020	vol. 77 p. 103135	10.1016/j.micpro.2020.103135	[220]
Section 3.3.1, Section 4.1			

Why statistical power matters for probabilistic real-time: work-in-progress			
F. Reghenzani, L. Santinelli, W. Fornaciari			
ACM Proceedings of the International Conference on Embedded Software Companion (EMSOFT'19)			
2019	no. 3 p. 1-2	10.1145/3349568.3351555	[218]
Section 3.3			

Statistical power estimation dataset for external validation GoF tests on EVT distribution			
F. Reghenzani, G. Massari, L. Santinelli, W. Fornaciari			
Data in brief, Elsevier			
2019	vol. 25 no. 104071	10.1016/j.dib.2019.104071	[217]
Section 7.2, Section 3.3			

A Probabilistic Approach to Energy-Constrained Mixed-Criticality Systems			
F. Reghenzani, G. Massari, W. Fornaciari			
IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)			
2019	pp. 1-6	10.1109/ISLPED.2019.8824991	[213]
Section 6.2			

Appendix A. List of Publications

Probabilistic-WCET Reliability: On the experimental validation of EVT hypotheses			
F. Reghenzani, G. Massari, W. Fornaciari, A. Galimberti			
ACM Proceedings of the International Conference on Omni-Layer Intelligent Systems			
2019	pp. 229-234	10.1145/3312614.3312660	[216]
Section 3.3.1, Section 4.1			

The Real-Time Linux Kernel: A Survey on PREEMPT_RT			
F. Reghenzani, G. Massari, W. Fornaciari			
ACM Computing Surveys (CSUR)			
2019	vol. 52 no. 1 pp. 1-36	10.1145/3297714	[214]
Section 5.1			

The Misconception of Exponential Tail Upper-Bounding in Probabilistic Real-Time			
F. Reghenzani, G. Massari, W. Fornaciari			
IEEE Embedded Systems Letters			
2019	vol. 11 no. 3 pp. 77-80	10.1109/LES.2018.2889114	[215]
Section 3.2			

chronovise: Measurement-based probabilistic timing analysis framework			
F. Reghenzani, G. Massari, W. Fornaciari			
Journal of Open Source Software			
2018	vol. 3 no. 28 p. 711	10.21105/joss.00711	[212]
Section 7.1			

A.2 Secondary papers

Predictive Reliability and Fault Management in Exascale Systems: State of the Art and Perspectives			
R. Canal, C. Hernandez, R. Tornero, A. Cilardo, G. Massari, F. Reghenzani, W. Fornaciari, M. Zapater, D. Atienza, A. Oleksiak, W. PiĄtek, and J. Abella			
ACM Computing Surveys (CSUR)			
2020	vol. 53 no. 5 pp. 1-32	10.1145/3403956	[42]
Section 1			

A Game Theory Approach to Heterogeneous Resource Management			
L. Premi, F. Reghenzani, G. Massari, W. Fornaciari			
ACM Proceedings of the International Conference on Embedded Software Companion (EMSOFT'20)			
2020	In publishing	DOI not available.	[205]
Chapter 1			

Predictive Resource Management in Energy-constrained Embedded Systems			
S. Crippa, G. Massari, F. Reghenzani, M. Zanella, W. Fornaciari			
IEEE Proceedings of Euromicro Conference on Digital System Design (DSD), 2020			
2020	In publishing	DOI not available.	[59]
Chapter 6.3			

Appendix A. List of Publications

Reliable power and time-constraints-aware predictive management of heterogeneous exascale systems			
W. Fornaciari, G. Agosta, D. Atienza, C. Brandolese, L. Cammoun, L. Cremona, A. Cilardo, A. Farres, J. Flich, C. Hernandez, M. Kulchewski, S. Libutti, J. M. Martínez, G. Massari, A. Oleksiak, A. Pupykina, F. Reghenzani, R. Tornero, M. Zanella, M. Zapater, D. Zoni			
ACM Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS '18)			
2018	pp. 187-194	10.1145/3229631.3239368	[95]
Section 1.3, Section 1.4.2, Section 5.3			

Managing Heterogeneous Resources in HPC Systems			
G. Agosta, W. Fornaciari, G. Massari, A. Pupykina, F. Reghenzani, M. Zanella			
ACM Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM'18)			
2018	pp. 7-12	10.1145/3183767.3183769	[8]
Chapter 5.3			

Mixed time-criticality process interferences characterization on a multicore Linux system			
F. Reghenzani, G. Massari, W. Fornaciari			
IEEE Proceedings of Euromicro Conference on Digital System Design (DSD), 2017			
2017	pp. 427-434	10.1109/DSD.2017.18	[211]
Chapter 5.1			

A.3 Under review

Representativity in Measurement-Based WCET			
F. Reghenzani, W. Fornaciari			
-			
2020	Under Review	DOI not available.	-
Section 4.2			

APPENDIX B

Computation of the Statistics of Tests

This appendix provides the state-of-the-art formula to compute the statistical tests used in this thesis.

B.1 KPSS

Omitting the mathematical proofs available in the original paper [154], the test statistic D_{KPSS} can be computed as:

$$\eta(X) = \left(\sum_{i=1}^{n} (X_i - \bar{X}) \right)^2 - n^2$$

$$D_{\text{KPSS}}(X) = \frac{\eta(X)}{\sigma_{X,l}}$$

where σ_X is the consistent estimate of the error variance computed for lags $1, ..., l$. The value of l can be computed with the following well-known formula [232]:

$$l = 12\sqrt[4]{\frac{n}{100}}$$

B.2 BDS

The test statistic can be computed with the following formula:

$$D_{\text{BDS}} = \sqrt{n-m+1}\frac{c_{m,n} - c_{1,m-n+1}^{m}}{\sigma_{m,n}} \tag{B.1}$$

where n is the sample size, m is the *embedding dimension*, $\sigma_{m,n}$ the consistent variance estimator and $c_{a,b}$ is defined as follow [141]:

$$c_{a,b} = 2\frac{1}{(b-a+1)(b-a)}\sum_{s=a}^{b}\sum_{t=s+1}^{b}\prod_{j=0}^{a-1} I(X_{s-j}, X_{t-j})$$

where

$$I(X_{s-j}, X_{t-j}) = \begin{cases} 1 & \text{if } |X_{s-j} - X_{t-j}| < \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

for some $\varepsilon > 0$. We do not provide here a detailed explanation of the above formulas, leaving the reader to examine them in detail in the cited statistical articles.

Under independence conditions, the D_{BDS} is normally distributed, therefore the critical region is obtained using the well-known t-student inverse-cdf.

B.3 R/S - Hurst

Given the time series $X = \{X_1, X_2, ..., X_n\}$ and its mean value $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ the function *cumulative sum* is defined as:

$$Z_c(X) = \sum_{i=1}^{c}(X_i - \bar{X})$$

The test statistic is the defined as:

$$D_{R/S}(X) = \frac{1}{\sqrt{n}} \frac{\max_c(Z_c(X)) - \min_c(Z_c(X))}{\sigma_X}$$

where σ_X is the sample standard deviation. If the values are uncorrelated the statistic follows the distribution having the following cdf:

$$F(v) = 1 + 2\sum_{i=1}^{\infty}(1 - 4k^2v^2) \cdot e^{-2(kv)^2}$$

from which the critical values can be computed by numerical methods.

B.4 KS

Given the cdf $F(x)$ of a reference distribution and an ecdf $\hat{F}_n(x)$ computed on a set of measurements of size n by Equation (2.1), the statistic of the Kolmogorov-Smirnov test is [142]:

$$D_{KS} = \sup_x |F_n(x) - F(x)|$$

The critical value is computed with the following closed form – valid for $n > 30$ – for KS test [227]:

$$CV_{KS} = \frac{\sqrt{-\frac{1}{2}\log\frac{\alpha}{2}}}{\sqrt{n}}$$

B.5 AD and MAD

Given the cdf $F(x)$ of a reference distribution and an ecdf $F_n(x)$ computed of a set of measurements of size n, the statistic of the Anderson-Darling test is [12]:

$$D_{A^2} = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\log(F(x_i)) - \frac{1}{n}\sum_{i=1}^{n}(2n-2i+1)\log(F(1-x_i))$$

while for modified Anderson-Darling test is [240]:

$$D_{AU^2} = \frac{n}{2} - 2\sum_{i=1}^{n}F(x_i) - \sum_{i=1}^{n}\frac{2n-2i+1}{n}\log(F(1-x_i))$$

For (M)AD test no critical value closed form is available because the critical value computation procedure strongly depends on the real (unknown) distribution. To obtain them, a dedicated Monte Carlo estimation, similar to the method used by Heo et al. [126], can be used to compute $CV_{(M)AD}$.

Bibliography

[1] J. Abella, D. Hardy, I. Puaut, E. Quiñones, and F. J. Cazorla. On the Comparison of Deterministic and Probabilistic WCET Estimation Techniques. In *2014 26th Euromicro Conference on Real-Time Systems*, pages 266–275, 2014. doi: 10.1109/ECRTS.2014.16.

[2] J. Abella, E. Quiñones, F. Wartel, T. Vardanega, and F. J. Cazorla. Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA. In *26th Euromicro Conference on Real-Time Systems*, pages 255–265, July 2014. doi: 10.1109/ECRTS.2014.33.

[3] Jaume Abella. MBPTA-CV, nov 2017. Zenodo. Software. doi: 10.5281/zenodo.1065776.

[4] Jaume Abella, Maria Padilla, Joan Del Castillo, and Francisco J. Cazorla. Measurement-Based Worst-Case Execution Time Estimation Using the Coefficient of Variation. *ACM Trans. Des. Autom. Electron. Syst.*, 22(4):72:1–72:29, June 2017. ISSN 1084-4309. doi: 10.1145/3065924.

[5] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*, pages 4–13, 1998. doi: 10.1109/REAL.1998.739726.

[6] Luca Abeni, Nicola Manica, and Luigi Palopoli. Reservation-based scheduling for IRQ threads. In *Proceedings of the 11th OSADL Real-Time Linux Workshop*, pages 179–186, Dresden, DEU, 2009. TU Dresden Faculty of Computer Science.

[7] I. Agirre, F. J. Cazorla, J. Abella, C. Hernandez, E. Mezzetti, M. Azkarate-askatsua, and T. Vardanega. Fitting Software Execution-Time Exceedance into a Residual Random Fault in ISO-26262. *IEEE Transactions on Reliability*, pages 1–14, 2018. ISSN 0018-9529. doi: 10.1109/TR.2018.2828222.

[8] Giovanni Agosta, William Fornaciari, Giuseppe Massari, Anna Pupykina, Federico Reghenzani, and Michele Zanella. Managing Heterogeneous Resources in HPC Systems. In *Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms*, PARMA-DITAM '18, page 7–12, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450364447. doi: 10.1145/3183767.3183769.

[9] Ankit Agrawal, Gerhard Fohler, Johannes Freitag, Jan Nowotsch, Sascha Uhrig, and Michael Paulitsch. Contention-Aware Dynamic Memory Bandwidth Isolation with Predictability in COTS Multicores: An Avionics Case Study. In Marko Bertogna, editor, *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, volume 76 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:22, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-037-8. doi: 10.4230/LIPIcs.ECRTS.2017.2.

[10] R Alpini and L Fattorini. Empirical performance of some goodness–of–fit tests for the weibull and type i extreme value distributions. *Statistica Applicata*, 5, 1993.

[11] T. W. Anderson and D. A. Darling. Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23(2):193–212, 06 1952. doi: 10.1214/aoms/1177729437.

[12] T. W. Anderson and D. A. Darling. A test of goodness of fit. *Journal of the American Statistical Association*, 49(268):765–769, 1954. doi: 10.1080/01621459.1954.10501232.

[13] Maya Arbel and Adam Morrison. Predicate RCU: An RCU for Scalable Concurrent Updates. In *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP 2015, pages 21–30, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3205-7. doi: 10.1145/2688500.2688518.

[14] Luís Fernando Arcaro, Karila Palma Silva, and Rômulo Silva De Oliveira. On the Reliability and Tightness of GP and Exponential Models for Probabilistic WCET Estimation. *ACM Trans. Des. Autom. Electron. Syst.*, 23(3):39:1–39:27, March 2018. ISSN 1084-4309. doi: 10.1145/3185154.

[15] Siro Arthur, Carsten Emde, and Nicholas Mc Guire. Assessment of the realtime preemption patches (RT-Preempt) and their impact on the general purpose performance of the system. In *Proceedings of the 9th OSADL Real-Time Linux Workshop*, Linz, AUT, 2007. Institute for Measurement Technology.

[16] David Bailey, Tim Harris, William Saphir, Rob Van Der Wijngaart, Alex Woo, and Maurice Yarrow. The NAS parallel benchmarks 2.0. Technical report, Technical Report NAS-95-020, NASA Ames Research Center, 1995.

[17] A. A. Balkema and L. de Haan. Residual life time at great age. *Ann. Probab.*, 2(5): 792–804, 10 1974. doi: 10.1214/aop/1176996548.

[18] S. K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *RTSS*. IEEE, 2011. doi: 10.1109/RTSS.2011.12.

[19] Sanjoy Baruah and Zhishan Guo. Mixed-criticality scheduling upon varying-speed processors. In *RTSS*. IEEE, 2013.

[20] Sanjoy Baruah and Zhishan Guo. Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor. In *RTSS*. IEEE, 2014.

[21] Sanjoy Baruah, Vincenzo Bonifaci, Gianlorenzo D'angelo, Haohan Li, Alberto Marchetti-Spaccamela, Suzanne Van Der Ster, and Leen Stougie. Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems. *Journal of the ACM (JACM)*, 62(2):1–33, 2015.

[22] Sanjoy Baruah, Arvind Easwaran, and Zhishan Guo. MC-Fluid: simplified and optimally quantified. In *RTSS*. IEEE, 2015.

[23] Sanjoy Baruah, Alan Burns, and Zhishan Guo. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors. In *ECRTS*. IEEE, 2016.

[24] R. Bender and S. Lange. Adjusting for multiple testing - when and how? *Journal of Clinical Epidemiology*, 54(4):343–349, April 2001. ISSN 0895-4356.

[25] Kostiantyn Berezovskyi, Luca Santinelli, Konstantinos Bletsas, and Eduardo Tovar. WCET measurement-based and extreme value theory characterisation of CUDA kernels.

In *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*, RTNS '14, page 279–288, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327275. doi: 10.1145/2659787.2659827.

[26] Kostiantyn Berezovskyi, Fabrice Guet, Luca Santinelli, Konstantinos Bletsas, and Eduardo Tovar. Measurement-based probabilistic timing analysis for graphics processor units. In *Architecture of Computing Systems - ARCS 2016 - 29th International Conference, Nuremberg, Germany, April 4-7, 2016, Proceedings*, pages 223–236. Springer International Publishing, 2016.

[27] G. Bernat, A. Colin, and S. M. Petters. WCET analysis of probabilistic hard real-time systems. In *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.*, pages 279–288. IEEE, Dec 2002. doi: 10.1109/REAL.2002.1181582.

[28] Wolfgang Betz, Marco Cereia, and Ivan Cibrario Bertolotti. Experimental evaluation of the Linux RT Patch for real-time applications. In *Emerging Technologies & Factory Automation (ETFA). IEEE Conference on*, pages 1–4. IEEE, Sept 2009. doi: 10.1109/ETFA.2009.5347056.

[29] A. Bhuiyan, F. Reghenzani, W. Fornaciari, and Z. Guo. Optimizing Energy in Non-preemptive Mixed-Criticality Scheduling by Exploiting Probabilistic Information. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–12, 2020. doi: 10.1109/TCAD.2020.3012231. In Publishing.

[30] Ashikahmed Bhuiyan, Zhishan Guo, Abusayeed Saifullah, Nan Guan, and Haoyi Xiong. Energy-efficient real-time scheduling of DAG tasks. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(5):84, 2018.

[31] Ashikahmed Bhuiyan, Sai Sruti, Zhishan Guo, and Kecheng Yang. Precise scheduling of mixed-criticality tasks by varying processor speed. In *RTNS*, 2019.

[32] Ashikahmed Bhuiyan, Kecheng Yang, Samsil Arefin, Abusayeed Saifullah, Nan Guan, and Zhishan Guo. Mixed-criticality multicore scheduling of real-time gang task systems. In *RTSS*. IEEE, 2019.

[33] Ashikahmed Bhuiyan, Di Liu, Aamir Khan, Abusayeed Saifullah, Nan Guan, and Zhishan Guo. Energy-efficient parallel real-time scheduling on clustered multi-core. *IEEE Transactions on Parallel and Distributed Systems*, 31(9):2097–2111, 2020.

[34] Enrico Bini, Giorgio Buttazzo, and Giuseppe Lipari. Minimizing cpu energy in real-time systems with discrete speed management. *ACM Transactions on Embedded Computing Systems (TECS)*, 8(4):1–23, 2009.

[35] Konstantinos Bletsas and Björn Andersson. Notional processors: An approach for multiprocessor scheduling. In *15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 3–12, April 2009. doi: 10.1109/RTAS.2009.25.

[36] C.E. Bonferroni. *Teoria statistica delle classi e calcolo delle probabilità*, volume 8. Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze, 1936.

[37] Pedro Braga, Luís Henriques, Bruno Carvalho, Philippe Chevalley, and Marco Zulianello. xluna-d emonstrator on esa mars rover. In *DASIA 2008 Data Systems In Aerospace*, volume 665, 2008.

[38] Björn B Brandenburg and Andrea Bastoni. The case for migratory priority inheritance in linux: Bounded priority inversions on multiprocessors. In *Proceedings of the 14th OSADL Real-Time Linux Workshop*, Chapel Hill, USA, 2012. Department of Computer Science, University of North Carolina.

[39] W. A. Broock, J. A. Scheinkman, W. D. Dechert, and B. LeBaron. A test for independence based on the correlation dimension. *Econometric Reviews*, 15(3):197–235, 1996. doi: 10.1080/07474939608800353.

[40] Jeremy H Brown and Brad Martin. How fast is fast enough? choosing between Xenomai and Linux for real-time applications. In *Proceedings of the 12th OSADL Real-Time Linux Workshop*, Nairobi, KEN, 2010. Strathmore University.

[41] Alan Burns and Robert I Davis. A survey of research into mixed criticality systems. *ACM Computing Surveys (CSUR)*, 50(6), 2017.

[42] Ramon Canal, Carles Hernandez, Rafa Tornero, Alessandro Cilardo, Giuseppe Massari, Federico Reghenzani, William Fornaciari, Marina Zapater, David Atienza, Ariel Oleksiak, Wojciech Piundefinedtek, and Jaume Abella. Predictive Reliability and Fault Management in Exascale Systems: State of the Art and Perspectives. *ACM Comput. Surv.*, 53(5), September 2020. ISSN 0360-0300. doi: 10.1145/3403956.

[43] F. P. Cantelli. Sulla determinazione empirica delle leggi di probabilità. *Giorn. Ist. Ital. Attuari*, 4:421–424, 1933.

[44] Andreu Carminati. Implementation and Evaluation of the Synchronization Protocol Immediate Priority Ceiling in PREEMPT-RT Linux. *Journal of Software*, 7(3), Mar 2012. doi: 10.4304/jsw.7.3.516-525.

[45] Enrique Castillo. *Extreme Value Theory in Engineering*. Statistical Modeling and Decision Science. Elsevier, 2012. ISBN 9780080917252.

[46] Enrique Castillo, Ali S Hadi, Narayanaswamy Balakrishnan, and José-Mariá Sarabia. *Extreme value and related models with applications in engineering and science*. Wiley Hoboken, NJ, nov 2005. ISBN 978-0-471-67172-5.

[47] Francisco J. Cazorla, Eduardo Quiñones, Tullio Vardanega, Liliana Cucu, Benoit Triquet, Guillem Bernat, Emery Berger, Jaume Abella, Franck Wartel, Michael Houston, Luca Santinelli, Leonidas Kosmidis, Code Lo, and Dorin Maxim. Proartis: Probabilistically analyzable real-time systems. *ACM Trans. Embed. Comput. Syst.*, 12(2s), May 2013. ISSN 1539-9087. doi: 10.1145/2465787.2465796.

[48] Francisco J. Cazorla, Leonidas Kosmidis, Enrico Mezzetti, Carles Hernandez, Jaume Abella, and Tullio Vardanega. Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. *ACM Comput. Surv.*, 52(1):14:1–14:35, February 2019. ISSN 0360-0300. doi: 10.1145/3301283.

[49] Felipe Cerqueira and Björn Brandenburg. A Comparison of Scheduling Latency in Linux, PREEMPT-RT, and LITMUS RT. In *Proceedings of International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*, pages 19–29. SYSGO AG, 2013.

[50] Konstantinos Chalas. Evaluation of real-time operating systems for fgc controls. Technical Report CERN-STUDENTS-Note-2015-201, CERN, 2015.

[51] Robert N Charette. This car runs on code. *IEEE spectrum*, 46(3):3, 2009.

[52] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, Sang-Ha Lee, and Kevin Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *Int. Symp. on Workload Characterization*, pages 44–54. IEEE, 2009.

[53] Jian-Jia Chen, Andreas Schranzhofer, and Lothar Thiele. Energy minimization for periodic real-time tasks on heterogeneous processing units. In *ISPA*. IEEE, 2009.

[54] X. Civit, J. del Castillo, and J. Abella. A reliable statistical analysis of the best-fit distribution for high execution times. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 727–734, 2018.

[55] Riccardo Confalonieri. Development and Evaluation of WCET Benchmarks for Probabilistic Real-Time Approaches. Master's thesis, Politecnico di Milano, Oct 2020.

[56] Jonathan Corbet. Priority inheritance in the kernel. *LWN.net*, Apr 2006. URL https://lwn.net/Articles/178253/.

[57] Jonathan Corbet. Full tickless operation in 3.10. *LWN.net*, May 2013. URL https://lwn.net/Articles/549580/.

[58] Michel Couillard and Matt Davison. A comment on measuring the hurst exponent of financial time series. *Physica A: Statistical Mechanics and its Applications*, 348:404 – 418, 2005. ISSN 0378-4371. doi: 10.1016/j.physa.2004.09.035.

[59] S. Crippa, G. Massari, F. Reghenzani, M. Zanella, and W. Fornaciari. Predictive Resource Management in Energy-constrained Embedded Systems. In *2017 Euromicro Conference on Digital System Design (DSD)*, Aug 2020.

[60] 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 *24th Euromicro Conference on Real-Time Systems*, July 2012. doi: 10.1109/ECRTS.2012.31.

[61] 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 *2012 24th Euromicro Conference on Real-Time Systems*, pages 91–101, 2012.

[62] Corentin Damman, Gregory Edison, Fabrice Guet, Eric Noulard, Luca Santinelli, and Jerome Hugues. Architectural Performance Analysis of FPGA Synthesized LEON Processors. In *Proceedings of the 27th International Symposium on Rapid System Prototyping: Shortening the Path from Specification to Prototype*, pages 33–40, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4535-4. doi: 10.1145/2990299.2990306.

[63] Neil T Dantam, Daniel M Lofaro, Ayonga Hereid, Paul Y Oh, Aaron D Ames, and Mike Stilman. The ach library: a new framework for real-time communication. *IEEE Robotics Automation Magazine*, 22(1):76–85, March 2015. ISSN 1070-9932. doi: 10.1109/MRA.2014.2356937.

[64] D. A. Darling. The kolmogorov-smirnov, cramer-von mises tests. *The Annals of Mathematical Statistics*, 28(4):823–838, 1957. ISSN 00034851.

[65] D. Dasari, B. Akesson, V. Nélis, M. A. Awan, and S. M. Petters. Identifying the sources of unpredictability in COTS-based multicore. In *Int. Symp. on Industrial Embedded Systems*, pages 39–48. IEEE, 2013. doi: 10.1109/SIES.2013.6601469.

[66] Robert Davis and Liliana Cucu-Grosjean. A Survey of Probabilistic Timing Analysis Techniques for Real-Time Systems. *Leibniz Transactions on Embedded Systems*, 6(1): 03–1–03:60, 2019. ISSN 2199-2002. doi: 10.4230/LITES-v006-i001-a003.

[67] Robert I. Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35:1–35:44, October 2011. ISSN 0360-0300. doi: 10.1145/1978802.1978814.

[68] Robert I. Davis, Alan Burns, and David Griffin. On the Meaning of pWCET Distributions and their use in Schedulability Analysis, 2017. URL https://www-users.cs.york.ac.uk/~robdavis/papers/RTSOPS2017pWCET.pdf.

[69] Robert Ian Davis and Liliana Cucu-Grosjean. A Survey of Probabilistic Schedulability Analysis Techniques for Real-Time Systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–53, May 2019. doi: 10.4230/LITES-v006-i001-a004.

[70] R. H. Dennard, F. H. Gaensslen, H. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[71] Mathieu Desnoyers, Paul E McKenney, Alan S Stern, Michel R Dagenais, and Jonathan Walpole. User-level implementations of read-copy update. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):375–382, Feb 2012. ISSN 1045-9219. doi: 10.1109/TPDS.2011.159.

[72] Kiril Dichev and Alexey Lastovetsky. Optimization of collective communication for heterogeneous HPC platforms. *High-Performance Computing on Complex Environments*, pages 95–114, 2014.

[73] Jean Diebolt, Armelle Guillou, Philippe Naveau, and Pierre Ribereau. Improving probability-weighted moment methods for the generalized extreme value distribution. *REVSTAT-Statistical Journal*, 6(1):33–50, 2008.

[74] E. W. Dijkstra. Structured programming. In *NATO Software Engineering Conference 1969*, Rome, Italy, Oct 1969.

[75] Peter Dinda, Xiaoyang Wang, Jinghang Wang, Chris Beauchene, and Conor Hetland. Hard real-time scheduling for parallel run-time systems. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '18, pages 14–26, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5785-2. doi: 10.1145/3208040.3208052.

[76] Jörg Dümmler. The nas parallel benchmarks 2.0. Technical report, Technical University Chemnitz, 2013. URL https://www.nas.nasa.gov/assets/pdf/techreports/1995/nas-95-020.pdf.

[77] Julien Durand, Youcef Bouchebaba, and Luca Santinelli. Statistical analysis for shared resources effects with multi-core real-time systems. In *13th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip, MCSoC 2019, Singapore, Singapore, October 1-4, 2019*, pages 362–371. IEEE, 2019. doi: 10.1109/MCSoC.2019.00058.

[78] A. Dvoretzky, J. Kiefer, and J. Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *Annals of Mathematical Statistics*, 27(3):642–669, 1956. doi: 10.1214/aoms/1177728174.

[79] S. Edgar and A. Burns. Statistical analysis of WCET for scheduling. In *Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001) (Cat. No.01PR1420)*, pages 215–224, 2001.

[80] Jake Edge. Moving interrupts to threads. *LWN.net*, Oct 2008. URL https://lwn.net/Articles/302043/.

[81] Pontus Ekberg and Wang Yi. Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. *Real-time systems*, 50(1):48–86, 2014.

[82] Glenn A Elliott and James H Anderson. The Limitations of Fixed-Priority Interrupt Handling in PREEMPT RT and Alternative Approaches. In *Proceedings of the 14th OSADL Real-Time Linux Workshop*, pages 149–155, Chapel Hill, USA, 2012. Department of Computer Science, University of North Carolina.

[83] Carsten Emde. Long-term monitoring of apparent latency in PREEMPT RT Linux realtime systems. In *Proceedings of the 12th OSADL Real-Time Linux Workshop*, Nairobi, KEN, 2010. Strathmore University.

[84] BS Emmanuel. Microcontroller-based intelligent power management system (IPDMS) for satellite application. *ARPN Journal of Engineering and Applied Sciences*, 7(3):377–384, 2012.

[85] Rolf Ernst and Marco Di Natale. Mixed criticality systems: A history of misconceptions? *IEEE Design & Test*, 33(5):65–74, 2016.

[86] Alexandre Esper, Geoffrey Nelissen, Vincent Nélis, and Eduardo Tovar. How realistic is the mixed-criticality real-time system model? In *RTNS*, 2015.

[87] Dario Faggioli, Fabio Checconi, Michael Trimarchi, and Claudio Scordino. An EDF scheduling class for the linux kernel. In *Proceedings of the 11th OSADL Real-Time Linux Workshop*, pages 197–204, Dresden, DEU, 2009. Technische Universität Dresden.

[88] Heiko Falk, Sebastian Altmeyer, Peter Hellinckx, Björn Lisper, Wolfgang Puffitsch, Christine Rochange, Martin Schoeberl, Rasmus Bo Sorensen, Peter Wägemann, and Simon Wegener. TACLeBench: A Benchmark Collection to Support Worst-Case Execution Time Research. In Martin Schoeberl, editor, *16th International Workshop on Worst-Case Execution Time Analysis (WCET 2016)*, volume 55 of *OpenAccess Series in Informatics (OASIcs)*, pages 2:1–2:10, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-025-5. doi: 10.4230/OASIcs.WCET.2016.2.

[89] Hasan Fayyad-Kazan, Luc Perneel, and Martin Timmerman. Linux PREEMPT-RT v2. 6.33 versus v3. 6.6: better or worse for real-time applications? *ACM SIGBED Review*, 11 (1):26–31, February 2014. ISSN 1551-3688. doi: 10.1145/2597457.2597460.

[90] Irina Fedotova, Bernd Krause, and Eduard Siemens. Upper Bounds Prediction of the Execution Time of Programs Running on ARM Cortex-A Systems. In Luis M. Camarinha-Matos, Mafalda Parreira-Rocha, and Javaneh Ramezani, editors, *Technological Innovation for Smart Systems*, pages 220–229, Cham, 2017. Springer International Publishing. ISBN 978-3-319-56077-9.

[91] M. Fernandez, D. Morales, L. Kosmidis, A. Bardizbanyan, I. Broster, C. Hernandez, E. Quinones, J. Abella, F. Cazorla, P. Machado, and L. Fossati. Probabilistic timing analysis on time-randomized platforms for the space domain. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 738–739, March 2017. doi: 10.23919/DATE.2017.7927087.

[92] T. B. Ferreira, M. A. Fernandes, and R. Matias Jr. A comprehensive complexity analysis of user-level memory allocator algorithms. In *2012 Brazilian Symposium on Computing System Engineering*, pages 99–104, Nov 2012. doi: 10.1109/SBESC.2012.27.

[93] 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. doi: 10.1017/S0305004100015681.

[94] J. Flich, G. Agosta, P. Ampletzer, D. A. Alonso, C. Brandolese, E. Cappe, A. Cilardo, L. Dragić, A. Dray, A. Duspara, W. Fornaciari, G. Guillaume, Y. Hoornenborg, A. Iranfar, M. Kovač, S. Libutti, B. Maitre, J. M. Martínez, G. Massari, H. Mlinarić, E. Papastefanakis, T. Picornell, I. Piljić, A. Pupykina, F. Reghenzani, I. Staub, R. Tornero, M. Zapater, and D. Zoni. MANGO: Exploring Manycore Architectures for Next-GeneratiOn HPC Systems. In *2017 Euromicro Conference on Digital System Design (DSD)*, pages 478–485, Aug 2017. doi: 10.1109/DSD.2017.51.

[95] William Fornaciari, Giovanni Agosta, David Atienza, Carlo Brandolese, Leila Cammoun, Luca Cremona, Alessandro Cilardo, Albert Farres, José Flich, Carles Hernandez, Michal Kulchewski, Simone Libutti, José Maria Martínez, Giuseppe Massari, Ariel Oleksiak, Anna Pupykina, Federico Reghenzani, Rafael Tornero, Michele Zanella, Marina Zapater, and Davide Zoni. Reliable Power and Time-constraints-aware Predictive Management of Heterogeneous Exascale Systems. In *Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, SAMOS '18, pages 187–194, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6494-2. doi: 10.1145/3229631.3239368.

[96] M. J. Frank. *Convolutions for Dependent Random Variables*, pages 75–93. Springer Netherlands, Dordrecht, 1991. ISBN 978-94-011-3466-8. doi: 10.1007/978-94-011-3466-8_4.

[97] Karl Pearson F.R.S. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900. doi: 10.1080/14786440009463897.

[98] Matteo Fusi, Fabio Mazzocchetti, Albert Farres, Leonidas Kosmidis, Ramon Canal, Francisco J. Cazorla, and Jaume Abella. On the use of probabilistic worst-case execution time estimation for parallel applications in high performance systems. *Mathematics*, 8(3):314, Mar 2020. ISSN 2227-7390. doi: 10.3390/math8030314.

[99] B. R. Gaines. Stochastic computing systems. In Julius T. Tou, editor, *Advances in Information Systems Science: Volume 2*, pages 37–172. Springer US, Boston, MA, 1969. ISBN 978-1-4899-5841-9. doi: 10.1007/978-1-4899-5841-9_2.

[100] Mark K. Gardner and Jane W. S. Liu. Analyzing stochastic fixed-priority real-time systems. In W. Rance Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 44–58, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49059-3.

[101] Philippe Gerum. Xenomai-Implementing a RTOS emulation framework on GNU/Linux. *White Paper*, 2004. URL https://xenomai.org/documentation/xenomai-2.0/pdf/xenomai.pdf.

[102] Rosa Arboretti Giancristofaro and Luigi Salmaso. Model performance analysis and model validation in logistic regression. *Statistica*, 63(2):375–396, 2007.

[103] V. Glivenko. Sulla determinazione empirica delle leggi di probabilità. *Giorn. Ist. Ital. Attuari*, 4:92–99, 1993.

[104] B. Gnedenko. Sur La Distribution Limite Du Terme Maximum D'Une Serie Aleatoire. *Annals of Mathematics*, 44(3):423–453, 1943. doi: 10.2307/1968974.

[105] Luis Claudio R Gonçalves and Arnaldo Carvalho de Melo. Application testing under realtime linux. In *Proceedings of the Linux Symposium*, pages 143–150, Ottawa, CAN, 2008.

[106] David Griffin and Alan Burns. Realism in Statistical Analysis of Worst Case Execution Times. In Björn Lisper, editor, *10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010)*, volume 15 of *OpenAccess Series in Informatics (OASIcs)*, pages 44–53, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-21-7. doi: 10.4230/OASIcs.WCET.2010.44.

[107] Tomás Grimm, Djones Lettnin, and Michael Hübner. A Survey on Formal Verification Techniques for Safety-Critical Systems-on-Chip. *Electronics*, 7(6):81, May 2018. ISSN 2079-9292. doi: 10.3390/electronics7060081.

[108] Austin Joint Working Group. Ieee standard for information technology-portable operating system interface (posix)-part 1: System application program interface (api)- amendment d: Additional real time extensions [c language]. *IEEE Std 1003.1d-1999*, pages 1–114, 1999. doi: 10.1109/IEEESTD.1999.91515.

[109] Fabrice Guet, L. Santinelli, and Jérôme Morio. On the Reliability of Probabilistic Worst-Case Execution Time Estimates. In *8th European Congress on Embedded Real Time Software and Systems (ERTS^2)*, Tolouse, France, 01 2016.

[110] Fabrice Guet, Luca Santinelli, and Jérôme Morio. On the Reliability of the Probabilistic Worst-Case Execution Time Estimates. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, TOULOUSE, France, January 2016.

[111] Fabrice Guet, Luca Santinelli, and Jerome Morio. On the Representativity of Execution Time Measurements: Studying Dependence and Multi-Mode Tasks. In Jan Reineke, editor, *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*, volume 57 of *OpenAccess Series in Informatics (OASIcs)*, pages 3:1–3:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. ISBN 978-3-95977-057-6. doi: 10.4230/OASIcs.WCET.2017.3.

[112] Fabrice Guet, Luca Santinelli, and Jerome Morio. On the Representativity of Execution Time Measurements: Studying Dependence and Multi-Mode Tasks. In Jan Reineke, editor, *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*, volume 57 of *OpenAccess Series in Informatics (OASIcs)*, pages 3:1–3:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-057-6. doi: 10.4230/OASIcs.WCET.2017.3.

[113] Dinakar Guniguntala, Paul E McKenney, Josh Triplett, and Jonathan Walpole. The read-copy-update mechanism for supporting real-time applications on shared-memory multiprocessor systems with linux. *IBM Systems Journal*, 47(2):221–236, 2008. ISSN 0018-8670. doi: 10.1147/sj.472.0221.

[114] Zhishan Guo and Sanjoy Baruah. Mixed-criticality scheduling upon varying-speed multiprocessors. In *DASC*. IEEE, 2014.

[115] Zhishan Guo, Luca Santinelli, and Kecheng Yang. EDF schedulability analysis on mixed-criticality systems with permitted failure probability. In *RTCSA*. IEEE, 2015.

[116] Zhishan Guo, Ashikahmed Bhuiyan, Abusayeed Saifullah, Nan Guan, and Haoyi Xiong. Energy-efficient multi-core scheduling for real-time dag tasks. In *ECRTS 2017*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[117] Zhishan Guo, Ashikahmed Bhuiyan, Di Liu, Aamir Khan, Abusayeed Saifullah, and Nan Guan. Energy-efficient real-time scheduling of DAGs on clustered multi-core platforms. In *RTAS*. IEEE, 2019.

[118] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper. The Mälardalen WCET Benchmarks: Past, Present And Future. In Björn Lisper, editor, *10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010)*, volume 15 of *OpenAccess Series in Informatics (OASIcs)*, pages 136–146, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-21-7. doi: 10.4230/OASIcs.WCET.2010.136.

[119] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, pages 3–14, 2001.

[120] Szabolcs Hajdu and Sándor-Tihamér Brassai. Implementation of Embedded Linux Systems on FPGA Based Circuits for Real Time Process Control. *MACRo 2015, Proceedings of the 5th International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics*, 1(1):145–154, 2015. ISSN 2247-0948.

[121] Jeffery Hansen, Scott Hissam, and Gabriel A. Moreno. Statistical-Based WCET Estimation and Validation. In Niklas Holsti, editor, *9th International Workshop on Worst-Case Execution Time Analysis*, volume 10, pages 1–11. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2009. ISBN 978-3-939897-14-9. doi: 10.4230/OASIcs.WCET.2009.2291.

[122] H. Hanson, S. W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, and J. Rubio. Thermal response to dvfs: analysis with an intel pentium m. In *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED '07)*, pages 219–224, Aug 2007. doi: 10.1145/1283780.1283827.

[123] Damien Hardy, Benjamin Rouxel, and Isabelle Puaut. The Heptane Static Worst-Case Execution Time Estimation Tool. In Jan Reineke, editor, *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*, volume 57 of *OpenAccess Series in Informatics (OASIcs)*, pages 8:1–8:12, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-057-6. doi: 10.4230/OASIcs.WCET.2017.8.

[124] Joachim Henkel. Selective revealing in open innovation processes: The case of embedded linux. *Research Policy*, 35(7):953 – 969, 2006. ISSN 0048-7333. doi: 10.1016/j.respol.2006.04.010.

[125] Luis Henriques. Threaded IRQs on Linux PREEMPT-RT. In *Proceedings of International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*, pages 23–32, Dublin, IRL, 2009. Euromicro.

[126] J.H. Heo, H. Shin, W. Nam, J. Om, and C. Jeong. Approximation of modified Anderson–Darling test statistics for extreme value distributions with unknown shape parameter. *Journal of Hydrology*, 499:41–49, 2013.

[127] Joseph Herkert, Jason Borenstein, and Keith Miller. The boeing 737 max: Lessons for engineering ethics. *Science and Engineering Ethics*, Jul 2020. ISSN 1471-5546. doi: 10.1007/s11948-020-00252-y.

[128] C. Hernandez, J. Abella, A. Gianarro, J. Andersson, and F. J. Cazorla. Random modulo: A new processor cache design for real-time critical systems. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016.

[129] M. T. Higuera-Toledano and V. Issarny. Analyzing the performance of memory management in rtsj. In *Proceedings Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. ISIRC 2002*, pages 26–33, 2002. doi: 10.1109/ISORC.2002.1003657.

[130] Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, and Lothar Thiele. Energy efficient DVFS scheduling for mixed-criticality systems. In *EMSOFT*. IEEE, 2014.

[131] Stephen Hunter and Tom Basciano. Advantages and challenges of using cots industrial networking technology on the international space station. *The Automation Conference 2015*, May 2015.

[132] H. E. Hurst. Long term storage capacity of reservoirs. *ASCE Transactions*, 116(776): 770–808, 1951.

[133] Hameed Hussain, Saif Ur Rehman Malik, Abdul Hameed, Samee Ullah Khan, Gage Bickler, Nasro Min-Allah, Muhammad Bilal Qureshi, Limin Zhang, Wang Yongji, Nasir Ghani, Joanna Kolodziej, Albert Y. Zomaya, Cheng-Zhong Xu, Pavan Balaji, Abhinav Vishnu, Fredric Pinel, Johnatan E. Pecero, Dzmitry Kliazovich, Pascal Bouvry, Hongxiang Li, Lizhe Wang, Dan Chen, and Ammar Rayes. A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709 – 736, 2013. ISSN 0167-8191. doi: 10.1016/j.parco.2013.09.009.

[134] International Electrotechnical Commission. IEC 61508 - Functional safety of electrical/-electronic/programmable electronic safety-related systems, Apr 2010.

[135] International Organization for Standardization. ISO 26262 – Road vehicles – Functional safety, Dec 2018.

[136] J. Jalle, L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. Bus designs for time-probabilistic multicore processors. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2014.

[137] R. Jayaseelan, T. Mitra, and Xianfeng Li. Estimating the worst-case energy consumption of embedded software. In *12th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)*, pages 81–90, 2006. doi: 10.1109/RTAS.2006.17.

[138] E Douglas Jensen, C Douglas Locke, and Hideyuki Tokuda. A time-driven scheduling model for real-time operating systems. In *IEEE Real-Time Systems Symposium (RTSS)*, volume 85, pages 112–122, San Diego, CA, USA, 1985.

[139] 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. *IEEE Embedded Systems Letters*, 9(3):69–72, 2017.

[140] Matthew Jones, Hans Fabricius Hansen, Allan Rod Zeeberg, David Randell, and Philip Jonathan. Uncertainty quantification in estimation of extreme environments. *Coastal Engineering*, 141:36 – 51, 2018. ISSN 0378-3839. doi: 10.1016/j.coastaleng.2018.07.002.

[141] Belaire-Franch Jorge and Contreras Dulce. How to compute the bds test: a software comparison. *Journal of Applied Econometrics*, 17(6):691–699, 2002. doi: 10.1002/jae.679.

[142] Frank J. Massey Jr. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. doi: 10.1080/01621459.1951.10500769.

[143] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1 (2):139–159, Jun 2009. ISSN 1866-9964. doi: 10.1007/s12559-009-9009-8.

[144] Dongwook Kang, Woojoong Lee, and Chanik Park. Kernel thread scheduling in real-time linux for wearable computers. *ETRI journal*, 29(3):270–280, 2007. ISSN 2233-7326. doi: 10.4218/etrij.07.0506.0019.

[145] Daniel Kästner and Christian Ferdinand. Safety standards and WCET analysis tools. In *Embedded Real Time Software and Systems (ERTS2012)*, Tolouse, France, Feb 2012. URL https://hal.archives-ouvertes.fr/hal-02192406.

[146] Weiyuen Kau, John H Cornish, Qadeer A Qureshi, and Shannon A Wichman. Method and apparatus for handling system management interrupts (SMI) as well as, ordinary interrupts of peripherals such as PCMCIA cards, Aug 2000. Holder Texas Instruments Inc., US Patent 6,112,273.

[147] Michalis Kokologiannakis and Konstantinos Sagonas. Stateless Model Checking of the Linux Kernel's Hierarchical Read-copy-update (Tree RCU). In *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, SPIN 2017, pages 172–181, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5077-8. doi: 10.1145/3092282.3092287.

[148] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. A cache design for probabilistically analysable real-time systems. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 513–518, 2013.

[149] L. Kosmidis, C. Curtsinger, E. Quiñones, J. Abella, E. Berger, and F. J. Cazorla. Probabilistic timing analysis on conventional cache designs. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 603–606, 2013.

[150] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. Efficient cache designs for probabilistically analysable real-time systems. *IEEE Transactions on Computers*, 63(12):2998–3011, 2014.

[151] 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 *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2014.

[152] L. Kosmidis, E. Quiñones, J. Abella, T. Vardanega, I. Broster, and F. J. Cazorla. Measurement-based probabilistic timing analysis and its impact on processor architecture. In *2014 17th Euromicro Conference on Digital System Design*, pages 401–410, Aug 2014. doi: 10.1109/DSD.2014.50.

[153] Kotz Samuel and Nadarajah Saralees. *Extreme Value Distributions: Theory and Applications*. World Scientific, 2000. ISBN 9781860944024.

[154] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159 – 178, 1992. ISSN 0304-4076. doi: 10.1016/0304-4076(92)90104-Y.

[155] M. R. Leadbetter and Holger Rootzen. Extremal theory for stochastic processes. *Ann. Probab.*, 16(2):431–478, 04 1988.

[156] Anthony W. Ledford and Jonathan A. Tawn. Diagnostics for Dependence within Time Series Extremes. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 65(2):521–543, 2003. ISSN 13697412, 14679868.

[157] Jaewoo Lee, Kieu-My Phan, Xiaozhe Gu, Jiyeon Lee, Arvind Easwaran, Insik Shin, and Insup Lee. MC-Fluid: Fluid model-based mixed-criticality scheduling on multiprocessors. In *RTSS*. IEEE, 2014.

[158] Jaewoo Lee, Hoon Sung Chwa, Linh TX Phan, Insik Shin, and Insup Lee. MC-ADAPT: Adaptive task dropping in mixed-criticality scheduling. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):1–21, 2017.

[159] Tau Leng, Rizwan Ali, Jenwei Hsieh, Victor Mashayekhi, and Reza Rooholamini. An empirical study of hyper-threading in high performance computing clusters. *Linux HPC Revolution*, 45, 2002.

[160] H. Leppinen. Current use of linux in spacecraft flight software. *IEEE Aerospace and Electronic Systems Magazine*, 32(10):4–13, October 2017. ISSN 0885-8985. doi: 10.1109/MAES.2017.160182.

[161] Benjamin Lesage, David Griffin, Frank Soboczenski, Iain Bate, and Robert I. Davis. A Framework for the Evaluation of Measurement-Based Timing Analyses. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, RTNS '15, page 35–44, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450335911. doi: 10.1145/2834848.2834858.

[162] Nancy G Leveson and Peter R Harvey. Software fault tree analysis. *Journal of Systems and Software*, 3(2):173–181, 1983.

[163] Jing Li, David Ferry, Shaurya Ahuja, Kunal Agrawal, Christopher Gill, and Chenyang Lu. Mixed-criticality federated scheduling for parallel real-time tasks. *Real-time systems*, 53 (5):760–811, 2017.

[164] L. Liang, P. E. McKenney, D. Kroening, and T. Melham. Verification of tree-based hierarchical read-copy update in the linux kernel. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 61–66, March 2018. doi: 10.23919/DATE.2018.8341980.

[165] Richard L Lieber. Statistical significance and statistical power in hypothesis testing. *Journal of Orthopaedic Research*, 8(2):304–309, 1990.

[166] G. Lima and I. Bate. Valid application of EVT in timing analysis by randomising execution time measurements. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 187–198, April 2017. doi: 10.1109/RTAS.2017.17.

[167] G. Lima, D. Dias, and E. Barros. Extreme Value Theory for Estimating Task Execution Time Bounds: A Careful Look. In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 200–211, July 2016. doi: 10.1109/ECRTS.2016.20.

[168] L.R. Lima and B. Neri. A test for strict stationarity. In Van-Nam Huynh, Vladik Kreinovich, Songsak Sriboonchitta, and Komsan Suriya, editors, *Uncertainty Analysis in Econometrics with Applications*, pages 17–30. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-35443-4.

[169] Di Liu, Jelena Spasic, Nan Guan, Gang Chen, Songran Liu, Todor Stefanov, and Wang Yi. EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees. In *RTSS*. IEEE, 2016.

[170] Andrew W Lo. Long-term memory in stock market prices. Technical report, National Bureau of Economic Research, 1989. doi: 10.3386/w2984.

[171] Robert Love and Andrew Morton. *Linux kernel development*, volume 1. Sams, 2004.

[172] Rapita Systems Ltd. Automating WCET analysis for DO-178B/C. https://www.rapitasystems.com/wcet-analysis-do-178-whitepaper, Unknown Year. Accessed Jun 2020.

[173] Y. Lu, T. Nolte, J. Kraft, and C. Norstrom. A Statistical Approach to Response-Time Analysis of Complex Embedded Real-Time Systems. In *2010 IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 153–160, 2010.

[174] Yue Lu, Thomas Nolte, Iain Bate, and Liliana Cucu-Grosjean. A new way about using statistical analysis of worst-case execution times. *SIGBED Rev.*, 8(3):11–14, September 2011. doi: 10.1145/2038617.2038619.

[175] Nicola Manica, Luca Abeni, Luigi Palopoli, Dario Faggioli, and Claudio Scordino. Schedulable device drivers: Implementation and experimental results. *Proceedings of International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*, pages 53–62, 2010. URL http://www.artist-embedded.org/docs/Events/2010/OSPERT/OSPERT2010-Proceedings.pdf.

[176] Paolo Mantegazza, Lorenzo Dozio, and S Papacharalambous. RTAI: Real time application interface. *Linux Journal*, 2000(72es):10, 2000. ISSN 1075-3583.

[177] Margarita Martínez-Díaz and Francesc Soriguera. Autonomous vehicles: theoretical and practical challenges. *Transportation Research Procedia*, 33:275 – 282, 2018. ISSN 2352-1465. doi: 10.1016/j.trpro.2018.10.103. XIII Conference on Transport Engineering, CIT2018.

[178] Miguel Masmano, Ismael Ripoll, Alfons Crespo, and Jorge Real. Tlsf: A new dynamic memory allocator for real-time systems. In *IEEE Proceedings of 16th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 79–88, June 2004. doi: 10.1109/EMRTS.2004.1311009.

[179] Giuseppe Massari, Federico Terraneo, Michele Zanella, and Davide Zoni. Towards fine-grained DVFS in embedded multi-core cpus. In *ARCS*. Springer, 2018.

[180] Alexander Maxiaguine, Simon Kunzli, and Lothar Thiele. Workload characterization model for tasks with variable execution demand. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, volume 2, pages 1040–1045. IEEE, 2004.

[181] C. Maxim, A. Gogonel, I. Asavoae, M. Asavoae, and L. Cucu-Grosjean. Reproducibility and Representativity: Mandatory Properties for the Compositionality of Measurement-Based WCET Estimation Approaches. *SIGBED Rev.*, 14(3):24–31, November 2017. doi: 10.1145/3166227.3166230.

[182] Daniel McFadden. Modeling the Choice of Residential Location. *Transportation Research Record*, 673:72–77, 1978.

[183] Paul E McKenney. Motivating lazy rcu callbacks under out-of-memory conditions, March 3 2015. US Patent 8,972,801.

[184] Paul E McKenney, Jonathan Appavoo, Andi Kleen, Orran Krieger, Rusty Russell, Dipankar Sarma, and Maneesh Soni. Read-copy update. In *AUUG Conference Proceedings*, pages 175–184, Kensington AUS, Sep 2001. AUUG Inc.

[185] Paul E McKenney, Silas Boyd-Wickizer, and Jonathan Walpole. RCU usage in the linux kernel: One decade later. Technical report, Linux Technology Center, IBM, Sep 2013. URL https://web.cecs.pdx.edu/~walpole/papers/rcu-usage2012.pdf.

[186] 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 *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–6, 2017.

[187] Mohd Adib Mohammad Razi, Wardah Tahir, Noratina Alias, Lokman Hakim Ismail, and Junaidah Ariffin. Development of rainfall model using meteorological data for hydrological use. *International Journal of Integrated Engineering*, 5(1), Nov. 2013.

[188] Amir Sabbagh Molahosseini, Leonel Seabra De Sousa, and Chip-Hong Chang. *Embedded systems design with special arithmetic and number systems*. Springer, 2017. doi: 10.1007/978-3-319-49742-6.

[189] Samuel K Moore. Another step toward the end of moore's law: Samsung and tsmc move to 5-nanometer manufacturing-[news]. *IEEE Spectrum*, 56(6):9–10, 2019.

[190] Jeremy Morse, Steve Kerrison, and Kerstin Eder. On the limitations of analyzing worst-case dynamic energy of processing. *ACM Trans. Embed. Comput. Syst.*, 17(3):59:1–59:22, February 2018. ISSN 1539-9087. doi: 10.1145/3173042.

[191] Linda K. Muthén and Bengt O. Muthén. How to use a monte carlo study to decide on sample size and determine power. *Structural Equation Modeling: A Multidisciplinary Journal*, 9(4):599–620, 2002. doi: 10.1207/S15328007SEM0904_8.

[192] G. Márton, I. Szekeres, and Z. Porkoláb. High-level c++ implementation of the read-copy-update pattern. In *2017 IEEE 14th International Scientific Conference on Informatics*, pages 243–248, Nov 2017. doi: 10.1109/INFORMATICS.2017.8327254.

[193] Shinichi Nakagawa. A farewell to Bonferroni: the problems of low statistical power and publication bias. *Behavioral Ecology*, 15(6):1044–1045, 11 2004. ISSN 1045-2249. doi: 10.1093/beheco/arh107.

[194] Sujay Narayana, Pengcheng Huang, Georgia Giannopoulou, Lothar Thiele, and R Venkatesha Prasad. Exploring energy saving for mixed-criticality systems on multi-cores. In *RTAS*. IEEE, 2016.

[195] Thomas Nolte, Meng Liu, and Bjorn Lisper. Challenges with probabilities in response-time analysis of real-time systems. In *5th Real-Time Scheduling Open Problems Seminar, RTSOPS, Spain*, pages 3–4, 2014.

[196] T. Ogasawara. An algorithm with constant execution time for dynamic storage allocation. In *Proceedings Second International Workshop on Real-Time Computing Systems and Applications*, pages 21–25, Oct 1995. doi: 10.1109/RTCSA.1995.528746.

[197] Edgar Elias Osuna. Tail-restricted stochastic dominance. *IMA Journal of Management Mathematics*, 24(1):21–44, 2013. doi: 10.1093/imaman/dpr023.

[198] Chandandeep Singh Pabla. Completely fair scheduler. *Linux Journalpabla*, 2009(184), August 2009. ISSN 1075-3583.

[199] James Pallister, Steve Kerrison, Jeremy Morse, and Kerstin Eder. Data dependent energy modeling for worst case energy consumption analysis. In *Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems*, pages 51–59, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5039-6. doi: 10.1145/3078659.3078666.

[200] Antonio Paolillo, Joël Goossens, Pradeep M Hettiarachchi, and Nathan Fisher. Power minimization for parallel real-time systems with malleable jobs and homogeneous frequencies. In *RTCSA*. IEEE, 2014.

[201] S. Park and M. Humphrey. Predictable high-performance computing using feedback control and admission control. *IEEE Transactions on Parallel and Distributed Systems*, 22 (3):396–411, March 2011. ISSN 1045-9219. doi: 10.1109/TPDS.2010.100.

[202] James Pickands III et al. Statistical inference using extreme order statistics. *the Annals of Statistics*, 3(1):119–131, 1975.

[203] Leo Porter, Michael A. Laurenzano, Ananta Tiwari, Adam Jundt, William A. Ward, Jr., Roy Campbell, and Laura Carrington. Making the Most of SMT in HPC: System- and Application-Level Perspectives. *ACM Trans. Archit. Code Optim.*, 11(4):59:1–59:26, January 2015. ISSN 1544-3566. doi: 10.1145/2687651.

[204] Aravinda Prasad and K Gopinath. A frugal approach to reduce rcu grace period overhead. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, pages 41:1–41:15, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5584-1. doi: 10.1145/3190508.3190522.

[205] L. Premi, F. Reghenzani, G. Massari, and W. Fornaciari. A Game Theory Approach to Heterogeneous Resource Management. In *2020 International Conference on Embedded Software*, 2020. In publishing.

[206] I. Puaut. Real-time performance of dynamic memory allocation algorithms. In *Proceedings 14th Euromicro Conference on Real-Time Systems. Euromicro RTS 2002*, pages 41–49, 2002. doi: 10.1109/EMRTS.2002.1019184.

[207] Bo Qian and Khaled Rasheed. Hurst exponent and financial market predictability. In *Proceedings of the Second IASTED International Conference on Financial Engineering and Applications*, 01 2004.

[208] James P. Quirk and Rubin Saposnik. Admissibility and measurable utility functions. *The Review of Economic Studies*, 29(2):140–146, 1962. doi: 10.2307/2295819.

[209] E. Quiñones, E. D. Berger, G. Bernat, and F. J. Cazorla. Using Randomized Caches in Probabilistic Real-Time Systems. In *2009 21st Euromicro Conference on Real-Time Systems*, pages 129–138, 2009.

[210] Pichai Raghavan, Amol Lad, and Sriram Neelakandan. *Embedded Linux system design and development*. CRC Press, Dec 2005. ISBN 9780849340581.

[211] F. Reghenzani, G. Massari, and W. Fornaciari. Mixed Time-Criticality Process Interferences Characterization on a Multicore Linux System. In *2017 Euromicro Conference on Digital System Design (DSD)*, pages 427–434, 2017. doi: 10.1109/DSD.2017.18.

[212] F. Reghenzani, G. Massari, and W. Fornaciari. chronovise: Measurement-Based Probabilistic Timing Analysis framework. *Journal of Open Source Software*, 3(28):711, 2018. doi: 10.21105/joss.00711.

[213] F. Reghenzani, G. Massari, and W. Fornaciari. A Probabilistic Approach to Energy-Constrained Mixed-Criticality Systems. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2019.

[214] F. Reghenzani, G. Massari, and W. Fornaciari. The Real-Time Linux Kernel: A Survey on PREEMPT_RT. *ACM Comput. Surv.*, 52(1):18:1–18:36, February 2019. ISSN 0360-0300. doi: 10.1145/3297714.

[215] F. Reghenzani, G. Massari, and W. Fornaciari. The Misconception of Exponential Tail Upper-Bounding in Probabilistic Real Time. *IEEE Embedded Systems Letters*, 11(3):77–80, 2019.

[216] F. Reghenzani, G. Massari, W. Fornaciari, and A. Galimberti. Probabilistic-WCET Reliability: On the Experimental Validation of EVT Hypotheses. In *Int. Conf. on Omni-Layer Intelligent Systems*, COINS, pages 229–234. ACM, 2019. ISBN 978-1-4503-6640-3. doi: 10.1145/3312614.3312660.

[217] F. Reghenzani, G. Massari, L. Santinelli, and W. Fornaciari. Statistical power estimation dataset for external validation GoF tests on EVT distribution. *Data in Brief*, 25:104071, 2019. ISSN 2352-3409. doi: 10.1016/j.dib.2019.104071.

[218] F. Reghenzani, L. Santinelli, and W. Fornaciari. Why Statistical Power Matters for Probabilistic Real-Time: Work-in-Progress. In *Proceedings of the International Conference on Embedded Software Companion*, EMSOFT '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369244. doi: 10.1145/3349568.3351555.

[219] F. Reghenzani, G. Massari, and W. Fornaciari. Probabilistic Predictability Index MATLAB Script, January 2020. Zenodo. doi: 10.5281/zenodo.3596957.

[220] F. Reghenzani, G. Massari, and W. Fornaciari. Probabilistic-WCET reliability: Statistical testing of EVT hypotheses. *Microprocessors and Microsystems*, 77:103135, 2020. ISSN 0141-9331. doi: 10.1016/j.micpro.2020.103135.

[221] F. Reghenzani, G. Massari, and W. Fornaciari. Timing predictability in high-performance computing with probabilistic real-time. *IEEE Access*, 8:208566–208582, 2020. doi: 10.1109/ACCESS.2020.3038559.

[222] F. Reghenzani, L. Santinelli, and W. Fornaciari. Dealing with Uncertainty in pWCET Estimations. *ACM Trans. Embed. Comput. Syst. (TECS)*, 19(5), 2020. ISSN 1539-9087. doi: 10.1145/3396234.

[223] R.D. Reiss and M. Thomas. *Statistical Analysis of Extreme Values: with Applications to Insurance, Finance, Hydrology and Other Fields*. Birkhäuser Basel, 2007. ISBN 9783764373993.

[224] Steven Rostedt and Darren V Hart. Internals of the rt patch. In *Proceedings of the Linux Symposium*, volume 2, pages 161–172, Ottawa, CAN, 2007.

[225] RTCA/EUROCAE. DO-178B - Software Considerations in Airborne Systems and Equipment Certification, Jan 1992.

[226] RTCA/EUROCAE. DO-178C - Software Considerations in Airborne Systems and Equipment Certification, Jan 2012.

[227] Lothar Sachs. *Angewandte Statistik*. Springer-Verlag Berlin Heidelberg, 1997. doi: 10.1007/978-3-662-05746-9.

[228] L. Santinelli, F. Guet, and J. Morio. Revising Measurement-Based Probabilistic Timing Analysis. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 199–208, April 2017. doi: 10.1109/RTAS.2017.16.

[229] Luca Santinelli and Zhishan Guo. On the Criticality of Probabilistic Worst-Case Execution Time Models. In Kim Guldstrand Larsen, Oleg Sokolsky, and Ji Wang, editors, *Dependable Software Engineering. Theories, Tools, and Applications*, pages 59–74. Springer, 2017. ISBN 978-3-319-69483-2.

[230] Luca Santinelli, Jérôme Morio, Guillaume Dufour, and Damien Jacquemart. On the Sustainability of the Extreme Value Theory for WCET Estimation. In Heiko Falk, editor, *14th International Workshop on Worst-Case Execution Time Analysis*, volume 39 of *OpenAccess Series in Informatics (OASIcs)*, pages 21–30, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-69-9. doi: 10.4230/OASIcs.WCET.2014.21.

[231] M. J. Schulte, M. Ignatowski, G. H. Loh, B. M. Beckmann, W. C. Brantley, S. Gurumurthi, N. Jayasena, I. Paul, S. K. Reinhardt, and G. Rodgers. Achieving exascale capabilities through heterogeneous computing. *IEEE Micro*, 35(4):26–36, 2015.

[232] G. William Schwert. Tests for unit roots: A monte carlo investigation. *Journal of Business & Economic Statistics*, 7(2), 1989. doi: 10.1198/073500102753410354.

[233] John Shalf, Sudip Dosanjh, and John Morrison. Exascale computing technology challenges. In José M. Laginha M. Palma, Michel Daydé, Osni Marques, and João Correia Lopes, editors, *High Performance Computing for Computational Science – VECPAR*

2010, pages 1–25, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-19328-6.

[234] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy. Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems. In *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9, June 2010. doi: 10.1109/SECON.2010.5508260.

[235] Jianping Shen, Michael Hamal, and Sven Ganzenmüller. Dynamic memory allocation on real-time linux. In *Proceedings of the 13th OSADL Real-Time Linux Workshop*, pages 187–193, Prague, CZE, 2011. Faculty of Electrical Engineering, Czech Technical University.

[236] Takaaki Shimura. Discretization of distributions in the maximum domain of attraction. *Extremes*, 15(3):299–317, Sep 2012. ISSN 1572-915X. doi: 10.1007/s10687-011-0137-7.

[237] Suresh Siddha, Venkatesh Pallipadi, and AVD Ven. Getting maximum mileage out of tickless. In *Proceedings of the Linux Symposium*, volume 2, pages 201–207, Ottawa, CAN, 2007.

[238] K. P. Silva, L. F. Arcaro, and R. S. d. Oliveira. On using GEV or Gumbel models when applying EVT for probabilistic WCET estimation. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 220–230. IEEE, Dec 2017. doi: 10.1109/RTSS.2017.00028.

[239] K. P. Silva, L. F. Arcaro, D. B. de Oliveira, and R. S. de Oliveira. An empirical study on the adequacy of MBPTA for tasks executed on a complex computer architecture with linux. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 321–328, Sep. 2018. doi: 10.1109/ETFA.2018.8502513.

[240] C.D. Sinclair, B.D. Spurr, and M.I. Ahmad. Modified anderson darling test. *Communications in Statistics - Theory and Methods*, 19(10):3677–3686, 1990. doi: 10.1080/03610929008830405.

[241] Sukhpal Singh and Inderveer Chana. A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of Grid Computing*, 14(2):217–264, Jun 2016. ISSN 1572-9184. doi: 10.1007/s10723-015-9359-2.

[242] M. Slijepcevic, L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. Time-analysable non-partitioned shared caches for real-time multicore systems. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2014.

[243] Paulo Baltarejo Sousa, Nuno Pereira, and Eduardo Tovar. Enhancing the real-time capabilities of the linux kernel. *SIGBED Rev.*, 9(4):45–48, November 2012. ISSN 1551-3688. doi: 10.1145/2452537.2452546.

[244] P Stakem. Flightlinux: A new option for spacecraft embedded computers. In *2001 Earth Science Technology Conference*, Collage Park, MD, USA, 2001. University of Maryland Conference Center.

[245] Nassim Nicholas Taleb and Pasquale Cirillo. Branching epistemic uncertainty and thickness of tails, 2019.

[246] Vadim Teverovsky, Murad S Taqqu, and Walter Willinger. A critical look at lo's modified r/s statistic. *Journal of Statistical Planning and Inference*, 80(1):211 – 227, 1999. ISSN 0378-3758. doi: 10.1016/S0378-3758(98)00250-X.

[247] T. . Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L. . Wu, and J. W. . Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In *Proceedings Real-Time Technology and Applications Symposium*, pages 164–173, 1995.

[248] D. Unat, A. Dubey, T. Hoefler, J. Shalf, M. Abraham, M. Bianco, B. L. Chamberlain, R. Cledat, H. C. Edwards, H. Finkel, K. Fuerlinger, F. Hannig, E. Jeannot, A. Kamil, J. Keasler, P. H. J. Kelly, V. Leung, H. Ltaief, N. Maruyama, C. J. Newburn, and M. Pericás. Trends in Data Locality Abstractions for HPC Systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):3007–3020, Oct 2017. doi: 10.1109/TPDS.2017.2703149.

[249] Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *RTSS*. IEEE, 2007.

[250] M. Völp, M. Hähnel, and A. Lackorzynski. Has energy surpassed timeliness? scheduling energy-constrained mixed-criticality systems. In *IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 275–284, April 2014. doi: 10.1109/RTAS.2014.6926009.

[251] P. Wägemann, T. Distler, T. Hönig, H. Janker, R. Kapitza, and W. Schröder-Preikschat. Worst-case energy consumption analysis for energy-constrained embedded systems. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 105–114, July 2015. doi: 10.1109/ECRTS.2015.17.

[252] Peter Wägemann, Christian Dietrich, Tobias Distler, Peter Ulbrich, and Wolfgang Schröder-Preikschat. Whole-System Worst-Case Energy-Consumption Analysis for

Energy-Constrained Real-Time Systems. In *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, volume 106, pages 1–25, 2018. ISBN 978-3-95977-075-0. doi: 10.4230/LIPIcs.ECRTS.2018.24.

[253] Ganghuai Wang, J. H. Lambert, and Y. Y. Haimes. Stochastic ordering of extreme value distributions. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(6):696–701, Nov 1999. ISSN 1083-4427. doi: 10.1109/3468.798077.

[254] Yong-quan Wang, Changxin Zhai, Yan-Hua Zhang, and Wang Gao. A new gps deformation monitoring algorithm applied to donghai bridge. *Journal of Shanghai Jiaotong University (Science)*, 13:216–220, 04 2008. doi: 10.1007/s12204-008-0216-3.

[255] F. Wartel, L. Kosmidis, C. Lo, B. Triquet, E. Quiñones, J. Abella, A. Gogonel, A. Baldovin, E. Mezzetti, L. Cucu, T. Vardanega, and F. J. Cazorla. Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 241–248, June 2013. doi: 10.1109/SIES.2013.6601497.

[256] F. Wartel, L. Kosmidis, A. Gogonel, A. Baldovino, Z. Stephenson, B. Triquet, E. Quiñones, C. Lo, E. Mezzetta, 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 *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 397–402, March 2015. doi: 10.7873/DATE.2015.0189.

[257] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. The Worst-Case Execution-Time Problem—Overview of Methods and Survey of Tools. *ACM Trans. Embed. Comput. Syst.*, 7(3), May 2008. ISSN 1539-9087. doi: 10.1145/1347375.1347389.

[258] Z. P. Wu, Y. Krish, and R. Pellizzoni. Worst Case Analysis of DRAM Latency in Multi-requestor Systems. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 372–383, Dec 2013. doi: 10.1109/RTSS.2013.44.

[259] Victor J Yodaiken. The rtlinux manifesto. Technical report, Department of Computer Science, New Mexico Institute of Technology, Socorro, NM, USA, 1999.

[260] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

[261] András Zempléni. Goodness-of-fit test in extreme value applications. Technical report, Ludwig-Maximilians-Universität München, 2004.

[262] Zhiyong Zhang. Monte carlo based statistical power analysis for mediation models: methods and software. *Behavior Research Methods*, 46(4):1184–1198, Dec 2014. ISSN 1554-3528. doi: 10.3758/s13428-013-0424-0.

[263] Fangfang Zhu, Yucong Chen, Jianqiang Wang, Gaofeng Zhang, and Qingguo Zhou. Experimental validation and exploration of a new kind of synchronization in linux. In *System and Software Reliability (ISSSR), International Symposium on*, pages 91–96. IEEE, Oct 2016. doi: 10.1109/ISSSR.2016.023.

Acronyms

BM Block Maxima.

ccdf Complementary Cumulative Distribution Function.

cdf Cumulative Distribution Function.

CFG Control-Flow Graph.

COTS Commercial-Off-The-Shelf.

DVFS Dynamic Voltage and Frequency Scaling.

ecdf Empirical Cumulative Distribution Function.

EVT Extreme Value Theory.

FLOPS Floating-Point Operations per Second.

GEVD Generalized Extreme Value Distribution.

GoF Goodness-of-Fit test.

GP2D Generalized Pareto Distribution (2-parameters).

GP3D Generalized Pareto Distribution (3-parameters).

GPD Generalized Pareto Distribution (generic).

Acronyms

GPOS General-Purpose Operating System.

HLS High-Level Schedule.

HPC High-Performance Computing.

i.i.d. Independent and Identically Distributed.

iccdf Inverse Complementary Cumulative Distribution Function.

icdf Inverse Cumulative Distribution Function.

IoT Internet of Things.

MBDTA Measurement-Based Deterministic Timing Analyses.

MBPTA Measurement-Based Probabilistic Timing Analyses.

MC Mixed-Criticality.

MDA Maximum Domain of Attraction.

pdf Probability Distribution Function.

pET Probabilistic Execution Time.

pmf Probability Mass Function.

PoT Peak-over-Threshold.

PPI Probabilistic Predictability Index.

PRNG Pseudo-Random Number Generator.

pWCEC Probabilistic Worst-Case Energy Consumption.

pWCET Probabilistic Worst-Case Execution Time.

Qos Quality-of-Service.

RCU Read-Copy-Update.

SAN Storage Area Network.

SDTA Static Deterministic Timing Analyses.

SLA Service Level Agreement.

SMI System Management Interrupts.

SPTA Static Probabilistic Timing Analyses.

WCEC Worst-Case Energy Consumption.

WCET Worst-Case Execution Time.

WCOT Worst-Case Observed Time.

Index

“Somewhere, something incredible is waiting to be known.”

(Misattributed to Carl Sagan, probable author Sharon Begley)