IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received October 27, 2020, accepted November 8, 2020, date of publication November 17, 2020, date of current version December 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038559

Timing Predictability in High-Performance Computing With Probabilistic Real-Time

FEDERICO REGHENZANI, (Member, IEEE), GIUSEPPE MASSARI, AND WILLIAM FORNACIARI, (Senior Member, IEEE)
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy

Corresponding author: Federico Reghenzani (federico.reghenzani@polimi.it)

This work was supported by the RECIPE FET-HPC EU H2020 project under Grant 801137.

ABSTRACT Application requirements in High-Performance Computing (HPC) are becoming increasingly exacting, and the demand for computational resources is rising. In parallel, new application domains are emerging, as well as additional requirements, such as meeting real-time constraints. This requirement, typical of embedded systems, is difficult to guarantee when dealing with HPC infrastructures, due to the intrinsic complexity of the system. Traditional embedded systems static analyses to estimate the Worst-Case Execution Time (WCET) are not applicable to HPC, because modeling and analyzing all the system's hardware and software components is not practical. Measurement-based probabilistic analyses for the WCET emerged in the last decade to overcome these issues, but it requires the system to satisfy certain conditions to estimate a correct and safe WCET. In this work, we show the emerging application timing requirements, and we propose to exploit the probabilistic real-time theory to achieve the required time predictability. After a brief recap of the fundamentals of this methodology, we focus on its applicability to HPC systems to check their ability to satisfy such conditions. In particular, we studied the advantages of having heterogeneous processors in HPC nodes and how resource management affects the applicability of the proposed technique.

INDEX TERMS Heterogeneous computing, high performance computing, real-time systems, statistical timing analysis.

I. INTRODUCTION

High-Performance Computing (HPC) aims at providing computing infrastructures capable of fulfilling the increasing performance requirements of modern applications, in both scientific and industrial domains. Figure 1 provides a simplified example of the hardware architecture of an HPC system. The infrastructure is a large distributed system, in which some servers (nodes) are devoted to specific management tasks, such as login, authentication, diagnostic, and entry point services. The core of the infrastructure consists of clusters of thousand of computing and parallel storage nodes interconnected by a network based on technologies such as Ethernet or Infiniband. To achieve high performance, computing nodes typically include multiple processors and multiple cores, because of the barrier of the single-core performance, mainly caused by power and thermal limits. More in detail, by looking at the evolution of the processors, the breakdown of the Dennard scaling [21] in 2005 triggered a change of paradigm, because we can no longer ignore the other side of the coin, i.e., power consumption. Chip design moved from pushing the single-core clock frequency to introducing the parallelism at the CPU-level, through multi-core architectures, instruction-level parallelism, and many other techniques. As a consequence of the end of Dennard scaling, maximizing energy efficiency has become the primary goal of the evolution of high-performance processors. To achieve this goal, processor manufacturers added many advanced features (e.g., pipelines, multi-level caches, vector instructions). In this picture, having both a distributed topology and high-performance multi-core processors, HPC systems offer the possibility of scaling the performance of the applications by leveraging on both *inter-node* and *intra-node* parallelisms. To effectively exploit such parallelisms, proper software frameworks are required [1]. On the one hand, the applications usually rely on well-known programming models (e.g., MPI, OpenMP); on the other hand, the HPC infrastructure needs to run a certain number of services and management frameworks on top of the operating system. These frameworks include at least a job scheduler

The associate editor coordinating the review of this manuscript and approving it for publication was Laxmisha Rai.

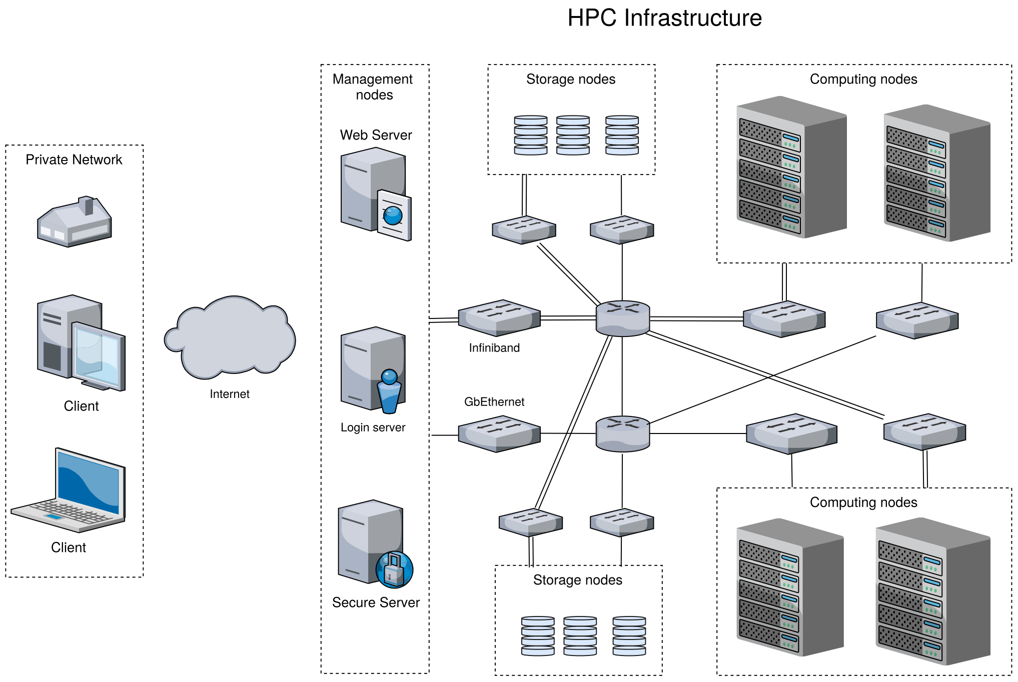


FIGURE 1. Simplified example of the architecture of an HPC system.

(or queue/resource manager) responsible for the management of the execution requests, i.e., it is in charge of assigning scheduling priorities and dispatching the runnable applications/jobs onto a selected set of computing nodes. The job scheduler is a critical component of the software stack since it directly impacts the occupancy of computing resources and the application execution time.

A. 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 needed. Moreover, the exploitation of parallelism is less challenging than having heterogeneous computational resources [16]. Nevertheless, from the 2010s, an increasing number of HPC systems started to include heterogeneous resources, as depicted in Figure 2 that shows the number of clusters in the TOP500 list exploiting heterogeneity. The achievement of exascale capabilities in HPC requires not only adequate computational performance but also the meeting of power and energy budget constraints. Heterogeneous architectures are a natural energy-efficient solution in this scenario [59].

B. PAPER ORGANIZATION

In this article, we propose to exploit a technique from the embedded world – i.e., probabilistic real-time – for addressing emerging challenges in HPC application timing requirements. The next Section II discusses the evolution of such requirements, the motivations behind this work, and the gap of the current state-of-the-art solutions. The necessary background is then provided in Section III, and how to apply the proposed technique to the HPC scenario is presented in Section IV. The subsequent Section V provides a qualitative analysis of the HPC infrastructure in connection with the applicability conditions of probabilistic real-time, followed by a quantitative experimental evaluation of a real HPC platform in Section VI. Finally, from the experimental results, we derive some recommendations to effectively perform resource management in such a scenario. These considerations and the conclusions of the paper are reported in Section VII.

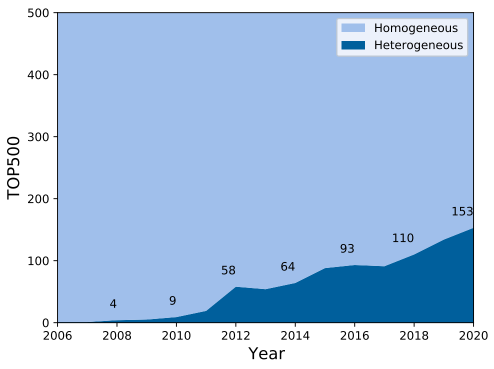


FIGURE 2. The increasing number of heterogeneous platforms in the TOP500 list.[2]

II. THE EVOLUTION OF APPLICATION TIMING REQUIREMENTS IN HPC

The performance of HPC systems is typically measured in terms of *throughput*, i.e., the amount of work completed in

[2]Data source: https://www.top500.org/statistics/list/

a given time unit. 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 community is striving towards *Exascale* compliant solutions [60], taking into account the power consumption envelope and trying to go beyond the scalability limits of current infrastructures.

A. THE CURRENT PERFORMANCE GOALS

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, [29] and [62], provide an overview of the current state-of-the-art strategies to deal with application requirements and resources 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.

When a user issues a request to execute a given application, the time he/she needs to wait until the output is called *turnaround time*. This is mainly composed of the *waiting time* and of the actual *execution time*. In fact, in large HPC infrastructure, the wait time introduces a significant component of the *turnaround time*. To make the HPC center cost-effective, the maximization of its utilization is usually a commercial goal. The waiting time may span from a few seconds to days, and a value of waiting time less than 25% of the execution time is considered acceptable by users [58]. The actual value depends on the availability of resources and on the priority. The priority, in turn, depends on many factors, and it can be dynamic. Usual metrics affecting the priority are the number and type of resources, the user privileges, the waiting time, and the past history of requests.

B. THE EMERGING TIMING REQUIREMENTS

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 [24]. Application requirements in such domains include specific timing constraints, similar to real-time applications running on embedded systems. The concept of timing constraint is well-expressed by the time-utility function proposed by Jensen *et al.* [31]. To each task, a time-utility function representing the satisfaction of the timing requirements is assigned with respect to its completion time and a defined deadline. This characterization is the generalization of soft- or hard- real-time system concepts. Three common examples are depicted in Figure 3. Case (c) is typical of soft real-time systems, while Case (a) is typical of a hard real-time system where deadlines cannot be missed

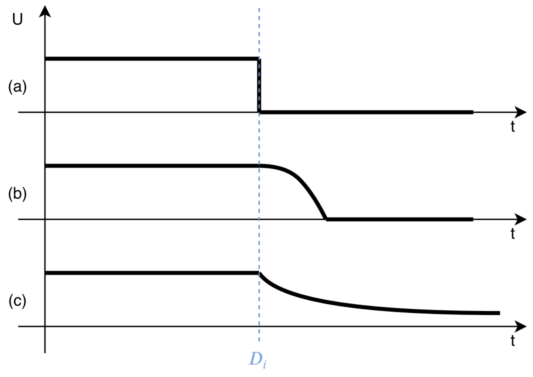


FIGURE 3. Three examples of utility functions with respect to the deadline D_i: (a) hard constraint, the system is failed immediately if the task overruns its deadline; (b) soft constraint, the system is considered functioning for a certain period of time, in degraded mode, for a given time frame after the deadline; (c) soft constraint, the system is considered functioning with a decreasing level of utility.

at all. 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.

EXAMPLES OF REAL WORLD APPLICATIONS

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 [44]. 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.

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 [46], 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, some bridge monitoring systems [66] require 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 human loss.

Currently, several research projects are addressing the challenges introduced by these emerging use cases. It is the case of recent EU H2020 projects, like MANGO [23] and RECIPE [24]. 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, 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.

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 a trade-off against 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. However, 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 [45]. In this regard, it is worth noting that the number of papers dealing with response-time driven resource management is only 15% of the total amount reported in the survey by Singh *et al.* [62]. 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 3c. Unfortunately, the use of the utility function, and in particular the analysis of the case of Figure 3a, is uncommon in HPC research. Even if the *hard* real-time on parallel architectures has been widely studied in recent decades [13], 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 (Commercial-Off-The-Shelf) components to reduce the cost of the clusters. A recent work [17] proposed a real-time scheduler for a COTS $\times 64$ 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 [51]. However, SMIs are just one of the numerous hardware and software mechanisms that make an HPC cluster unpredictable in time.

C. THE CHALLENGES FOR TIME PREDICTABILITY

The increase in complexity of computing architectures discussed in Section I makes the problem of computing the *Worst-Case Execution Time (WCET)* of a job or task, running on an HPC system, a challenging problem. Traditional WCET static analyses are based on control-flow graph and architectural timing specifications to detect the worst-case conditions and compute the WCET value. Unfortunately, these analyses failed to progress sufficiently fast to get reliable and tight WCET estimation in modern processors. They require either an unfeasible computational complexity or produce an overpessimistic WCET estimation, which would make the obtained value useless [36]. However, the computation of WCET for timing-sensitive applications, having the previously described timing requirements, is mandatory if we have to provide any sort of guarantee for mission- and safety-critical systems. Performing such analyses for HPC infrastructures is even more complicated or almost impossible, due to the extensive use of COTS hardware, general-purpose operating systems, and unpredictable inter-node networking infrastructure. These factors affect the predictability of the task execution time [63] and make the exploration of the hardware states too large to be computed in a reasonable time [49]. On the other hand, moving computing resources towards specialized components and custom real-time operating systems would be a possible solution to the predictability problem, allowing the use of traditional WCET analyses. At the same time, this solution in HPC 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.

D. THE PROPOSED SOLUTION AND CONTRIBUTIONS

To guarantee the timing predictability of the applications, without impacting the HPC infrastructure and its overall average performance, we propose relying on an approach that has come out recently in the real-time *Cyber-Physical System (CPS)*[3] world: *Probabilistic Real-Time Computing*. This paradigm is based on the computation of the WCET by using statistical frameworks applied to measurement-based techniques. From the industrial perspective, this is very attractive because it is easy to develop and does not require complex analyses of the application code and the platforms. However, Probabilistic Real-Time Computing is still immature due to a large number of open challenges [32] that have stimulated scientific research in recent years. Most of these challenges are related to the necessity to prove the safety of the approach. 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 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 system. 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 costs while maximizing the utilization of the system.

To summarize, in this manuscript, we:

1) showed, in the current section, the emerging time-critical application requirements in the HPC environment and which are the gaps in the current approaches;
2) propose to use the probabilistic real-time theory on the HPC applications instead of the embedded systems, and we introduce the necessary background of this technique for the reader;
3) identify the barriers for the exploitation of probabilistic real-time in HPC systems, and we discuss how heterogeneous computing may be helpful to solve such issues;
4) analyze, both qualitatively and quantitatively, the satisfaction of the probabilistic real-time hypotheses for different resource management choices, basing the evaluation on experiments performed on a real HPC cluster;
5) discuss the different benefits of this technique in improving not only the meeting of the application requirements, but also the management of the job queue and, in general, the HPC resources.

[3]For the sake of readiness, in this paper, we use the term CPS to refer to the real-time class of embedded systems.

III. BACKGROUND ON PROBABILISTIC REAL-TIME

Probabilistic real-time has been a hot topic since the beginning of the 2000s. The early works in applying EVT to WCET estimation are the paper by Edgar *et al.* [20] published in 2001, and the article by Bernat *et al.* [7] published in 2002. The concept of *Measurement-Based Probabilistic Timing Analysis (MBPTA)* was instead formalized about a decade later, in 2013, by Cucu-Grosjean *et al.* [11]. The MBPTA is a very convenient technique, because it makes the WCET computation possible without much effort, regardless of the hardware complexity. This method estimates a statistical distribution of the WCET directly from the execution time observations of the applications running on the real system. No complex control-flow analyses or details about the hardware configuration are needed, provided that some hypotheses, later presented in Section III-D, are satisfied.

A. PROBABILISTIC-WCET

The goal of the MBPTA estimation method is to compute the so-called *probabilistic-WCET (pWCET)*. The pWCET is not a single scalar value like the outcome of classical WCET analysis tools. Instead, the pWCET is a probability distribution, and it is usually expressed with the following formula:

$$p = P(X > \overline{WCET}) = 1 - F(\overline{WCET})$$

where X is the random variable representing the execution time, p the probability of observing execution times larger than $\overline{WCET}$, and $F(\cdot)$ the cumulative distribution function (cdf). We informally call p the *violation probability*. Depending on the requirements, the developer can set a value for p and compute the related $\overline{WCET}$ or, vice versa, set a value for $\overline{WCET}$ and compute the violation probability p. Provided that it is possible to guarantee that the violation probability is not underestimated, its value can be added to the fault analysis process of a safety-critical system. In embedded safety-critical systems, the required probability of failure can be very low: For instance, the aviation standard DO-178C [56] requires a probability of failure for the most critical systems to be $p = 10^{-9}$. In order to guarantee the safety of probabilistic real-time, the cdf $F(\cdot)$ must be estimated correctly. The simplest way to estimate the $F(\cdot)$ function is to compute the ecdf (Empirical Cumulative Distribution

Function) directly from the time measurements. However, even if all the statistical hypotheses of ecdf are satisfied, to use the pWCET with such low probability values with a high degree of confidence, we need to collect a large number of samples. The confidence is formally defined as the probability of having estimated an unsafe pWCET distribution: $\zeta = P[F(\overline{WCET}) > F^*(\overline{WCET})]$, where F^* is the real (and unknown) cdf. This is not a run-time property but rather the probability of having computed a wrong distribution in the offline EVT analysis of the pWCET distribution. For example, according to the Dvoretzky-Kiefer-Wolfowitz inequality [19], to estimate the ecdf directly from timing samples, with a violation probability $p = 10^{-9}$ and confidence of $\zeta = 10^{-10}$, we need to collect $\approx 10^{20}$ time measurements, which is clearly not practical. For this reason, the statistical theory described in the following paragraph has been created and can be used for our purposes.

B. EXTREME VALUE THEORY

The *Extreme Value Theory (EVT)* has been developed to study the so-called *tails* of the distribution, and it is used for rare events predictions, especially for the forecasting of natural disasters. This statistical theory goes in the opposite direction with respect to the well-known central limit theorem, which instead focuses on the behavior of the distribution around its mean value.

Given a sequence of independent and identically distributed (iid) random variables $X_1, X_2, \ldots, X_n$, the EVT provides the limit distribution at the extremes, i.e. the $\max(X_1, X_2, \ldots, X_n)$ or $\min(X_1, X_2, \ldots, X_n)$. In our scenario, $X_1, X_2, \ldots, X_n$ is a sequence of execution times of a given program or task. Consequently, by excluding the *min* because we are interested in the maximum value (WCET), we can formalize the probability of not incurring in an execution time longer than a certain value x (deadline missing) as follows:

$$\begin{aligned} &P(\max(X_1, X_2, \ldots, X_n) \leq x) \\ &\quad = P(X_1 \leq x, X_2 \leq x, \ldots, X_n \leq x) \\ &\quad \overset{\text{iid}}{=} P(X_1 \leq x)P(X_2 \leq x)\cdots P(X_n \leq x) \\ &\quad = F^n(x) \end{aligned} \tag{1}$$

where $F(x)$ is the cumulative distribution function of X_i. Without going into the statistical details, it is possible to demonstrate that [9]:

$$\exists a_n, b_n \text{ s.t. } \lim_{n\to\infty} F^n(a_n x + b_n) = G(x) \tag{2}$$

where $G(x)$ is the cdf of the so-called *Extreme Value Distribution*. The form of this distribution is well-known and can be estimated from the samples through appropriate and well-assessed methods. According to the Fisher-Tippett-Gnedenko theorem [22], $G(x)$ converges to the *Generalized Extreme Value Distribution (GEVD)* or, asymptotically equivalently, to the *Generalized Pareto Distribution (GPD)*. Their probabilistic distribution functions are respectively:

$$G(x) = \begin{cases} e^{-(1+\xi(\frac{x-\mu}{\sigma}))^{-\frac{1}{\xi}}} & \xi \neq 0 \\ e^{-e^{-\frac{x-\mu}{\sigma}}} & \xi = 0 \end{cases} \quad G{\sim}\text{GEVD}(\mu,\sigma,\xi) \tag{3}$$

$$G(x) = \begin{cases} 1 - (1 + \xi\dfrac{x-\mu}{\sigma})^{-\frac{1}{\xi}} & \xi \neq 0 \\ 1 - e^{-\frac{x-\mu}{\sigma}} & \xi = 0 \end{cases} \quad G{\sim}\text{GPD}(\mu,\sigma,\xi) \tag{4}$$

Both distributions have three parameters[4]: the location μ, the scale σ, and the shape ξ. While the first two parameters are a similar concept to the average and standard deviation, the third parameter has an important role when considering pWCET distributions. Depending on the sign of this parameter, the tails of the distribution significantly change, as shown in Figure 4. The tail has a critical effect on pWCET, as discussed later in detail in Section III-E.

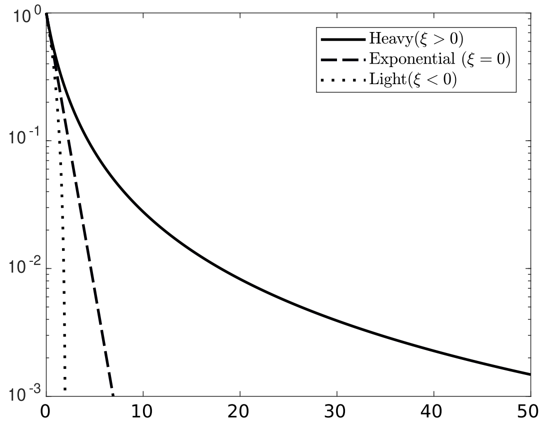


FIGURE 4. The three different tails of the Complementary Cumulative Distribution Function of extreme value distributions.

Provided that some hypotheses are satisfied (presented in Section III-D), the EVT ensures that the maxima (or the minima) values of the observed phenomenon, no matter what the original distribution is, are distributed according to the GEVD or GPD distribution. This is a key property of our purposes: since we are interested in the WCET values, the pWCET distribution we are searching for is distributed according to GEVD or GPD. The fundamental advantage, since we do not need to know the original distribution of the execution time, is that we do not need to know the timing model of the platform. This simplifies the problem of computing the pWCET for a given task or application compared to traditional static analyses. In fact, to statically estimate the WCET, it would have been necessary to know a detailed timing model of the system's hardware and software.

[4]For completeness, the GPD distribution is sometimes presented with two parameters only, assuming $\mu = 0$. This difference has no relevance in this work.

C. THE ESTIMATION PROCESS

Two possible estimation methods exist to fit the GEVD or GPD. The first is called *Block-Maxima (BM)*: the following *filter* is applied to the sequence of observations X_1, $X_2, \ldots, X_n$:

$$Y_i = max(X_{B\cdot(i-1)+1}, X_{B\cdot(i-1)+2}, \ldots, X_{B\cdot i}) \tag{5}$$

where B is a parameter called *block size*. In other words, the set of observations is divided into blocks of fixed size B, and for each block, the maximum value is taken. The sequence $Y_1, Y_2, \ldots, Y_{\lceil n/B \rceil}$ represents the maxima of the blocks. The second approach is called *Peak-over-Threshold (PoT)*, and it is a simple threshold filter:

$$Y = \{X_i > u\} \tag{6}$$

where u is a predefined threshold. The sequence Y_1, $Y_2, \ldots, Y_m$ represents the measurements that are greater than the threshold u, discarding all the values under this threshold. Then, any traditional parameter estimation algorithm can be used to fit the distribution, e.g., the Maximum Likelihood Estimator (MLE) or Probabilistic Weighted Moments (PWM). When BM is used, the resulting distribution is the GEVD. Vice versa, when PoT is used, the resulting distribution is the GPD.

The whole pWCET estimation process can be summarized in the following steps:

1) acquire the time measurements $X_1, X_2, \ldots, X_n$;
2) check the EVT hypotheses on inputs (see Section III-D);
3) apply BM or PoT filtering to the time measurements to obtain a new set of execution times $Y_1, Y_2, \ldots, Y_n$ representing the WCET behaviors;
4) run the estimator (e.g. MLE or PWM);
5) compare the obtained distribution with the original samples thanks to a Goodness-of-Fit (GoF) test. This is to verify the last EVT hypothesis (see Section III-D);
6) compute the WCET given the desired violation probability by using the cdf of the estimated pWCET.

By following these steps, it is possible to estimate the pWCET distribution with a certain degree of confidence. Steps 2 and 5 are performed with proper statistical tests. To be effective in the detection of a violation, these tests require a sufficient number of samples, i.e., a minimum value for n. The estimator is able to run with a very low number of samples but, depending on the required safety level, the statistical tests usually require a larger number of samples to achieve sufficient confidence. Two recent works [54], [55] already addressed the problem of computing the minimum amount of time measurements to capture in order to obtain reliable results from the tests and how to deal with estimation errors.

D. APPLICABILITY

The possible applications of the probabilistic theory, shown in the previous section, rely on the fact that the pWCET is correctly computed. The EVT produces reliable results only if three assumptions are satisfied: the input representativity, the independent and identically distributed condition, and the maximum domain of attraction hypotheses [53]. Such conditions are analyzed in detail in the following paragraphs.

INPUT REPRESENTATIVITY

In general, the execution time of the jobs also depends on the input data. In any measurement-based approach, the observed execution time must be generated by an input set representative of the real-world scenario. This hypothesis usually requires all the paths in the control-flow-graph of the program to be covered, leaving the probabilistic theory to deal with the uncertainty related to the hardware state. If covering all the paths is too expensive, at least a representative subset of them should be selected and covered. The verification of this hypothesis depends mainly on the application code, and making general statements on its validity is not possible.

INDEPENDENT AND IDENTICALLY DISTRIBUTED CONDITION

The input sequence of EVT, i.e., the time trace in our scenario, must be independent and identically distributed (iid). In CPS, the hardware effects, such as the presence of a cache memory, are the main causes of violation of the iid condition, making the execution times dependent on the subsequent job executions. This condition can be split into stationarity, short-range, and long-range dependence hypotheses [57]. The stationarity sub-hypothesis requires the execution times to be generated by the same statistical process. In the computer science scenario, this hypothesis is guaranteed if the job has no abrupt behavior change (in timing sense). The short-range requires that the elements of the sequence of execution times do not present a statistical dependency with the near elements. The long-range dependence, instead, requires that the job execution time does not present *seasonality*, i.e., they have no long periodicity.

MAXIMUM DOMAIN OF ATTRACTION HYPOTHESIS

The last hypothesis is related to a more statistical detail: the distribution of the execution times must be in the *Maximum Domain of Attraction (MDA)* of an extreme value distribution. This condition is almost always true if the time measurements are considered as continuous samples, while in the discrete case (e.g., number of clocks), this cannot be assumed true and must be verified. In any case, to check the validity of the MDA hypothesis, a goodness-of-fit statistical test is usually performed [53] a posteriori of the pWCET analysis. This goodness-of-fit test also helps to identify any error in the estimation phase due to its ability to detect the discrepancies between the estimated pWCET distribution and the real measured data.

E. THE CRITICAL PARAMETER: ξ

Both distributions GEVD and GPD include the so-called *shape* parameter ξ. This parameter defines which subclass of GEVD or GPD the distribution actually is. The sub-classes

TABLE 1. Extreme value theory distributions.

Class	ξ	Distribution
GEVD(μ, σ, ξ)	<0	Weibull
	=0	Gumbel
	>0	Fréchet
GPD(μ, σ, ξ)	<0	-
	=0	Exponential
	>0	Pareto if $\mu = \sigma/\xi$

for the GEVD case and the GPD case are summarized in Table 1 and their tails are depicted in Figure 4.

There are important differences from the real-time computing perspective: if $\xi < 0$, it means that a finite WCET value exists because the cdf is finite, and it has an upper-bound: $\exists x : F(x) = 1$. In other cases ($\xi \geq 0$), the WCET value is infinite. However, for the $\xi = 0$ case, the cdf goes to 1 extremely fast, so we can easily consider a finite WCET with an extremely high level of confidence.

On the contrary, the $\xi > 0$ is a critical case: the cdf of the pWCET curve is slowly going to one and it does not have a finite maximum, i.e. $\nexists x < \infty : F(x) = 1$. Even worse, if $\xi \geq 1$, the mean value of the distribution is unbounded. This means that we may be in one of the following cases: (1) the architecture or the job has this silly behavior or (2) the estimation procedure has been incorrectly performed. The first case is quite uncommon but not impossible. For example, some architectures may generate a distribution with $\xi > 0$ because they cannot provide an upper-bound for memory latencies [35], theoretically making memory access to require an unlimited time, even if this is highly improbable.

If the distribution passes all the statistical tests, it means that it adheres to the original execution time samples; thus, we have to consider it valid, even if it may turn out to be unusable from the scheduling and resource management perspective. For this reason, one of the goals of the experimental evaluation of this work is to assess if different computing configurations can affect the value of the parameter ξ and, consequently, the tightness of the estimated pWCET.

IV. 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.

A. 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, in its simplest form, by a period or minimal interarrival time T_i, a relative deadline D_i, and a WCET C_i: $\tau_i = (T_i, D_i, C_i)$. 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 $f_{i,j}$. A schedule is said to be feasible if $f_{i,j} \leq a_{i,j} + D_i$ for any job. In a non-preemptive system, since C_i is the WCET, the condition becomes $a_{i,j} + D_i \geq s_{i,j} + C_i$, where $s_{i,j}$ is the starting time of the j-th job spawned by the i-th task. The difference $s_{i,j} - a_{i,j}$ represents the waiting time for the job.

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 previously defined in Section II-A. 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,[5] 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, 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 paper, we focus on the probabilistic estimation of the WCET C_i only.

A recent work by Fusi *et al.* [25] studied the applicability of probabilistic real-time techniques for a geophysical exploration HPC application. In particular, they employed

[5]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.

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 [25] 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 has no effect because no cache data is maintained across job executions, as detailed later in Section V-B.

B. PROBABILISTIC REAL-TIME IN CPS

Research in probabilistic real-time has focused on different aspects that can be categorized into two classes: methodology studies and architectural enablers. Two recent surveys, [10] and [14], provided a general overview of probabilistic real-time in CPS. The methodology papers focused on the application of the EVT to the pWCET problem. Applying the statistical theory to real-world scenarios is, in fact, not trivial and requires methods and procedures designed to fit the specific problem domain. In a previous work [52], we investigated the adherence of such hypotheses for embedded systems by running WCET benchmarks on different platforms, from a simple microprocessor to a complex embedded Linux system. A detailed description of the estimation process and statistical tests is available in the paper by Santinelli *et al.* [57] and by Lima *et al.* [41]. Instead, the research on the architectural aspects is mostly focused on building computing architectures able to satisfy the iid requirement of the EVT theory, such as the randomized architectures [48]. The idea behind these architectures is to make the execution time independent across different executions by randomizing the hit/miss cache behavior. The problem has been faced with both hardware [27], [37], [64] and software solutions [38], [39]. Other randomized hardware components were proposed, for instance, memory buses [30]. However, the use of randomized architectures falls outside the scope of this work: similar to the last paragraph of Section IV-A, the dependency on the overall execution time of a traditional HPC application is not influenced by the initial state of the caches, making the use of randomized architecture at node-level less relevant for achieving EVT compliance in HPC. Instead, we will show that a randomized behavior can be instead advantageous if applied at a larger scale VII-A.

Research on probabilistic real-time theory for CPS is still ongoing and very active in both categories. The scientific community has not reached a consensus yet on the practical exploitation of this theory in real safety-critical systems and particular skepticism has been expressed as to whether these results can be compliant for the certification processes.

C. ON THE ADVANTAGES OF USING pWCET IN HPC

The novel idea proposed is to apply the probabilistic WCET analyses for CPS to HPC environments, 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: (1) obtaining real-time guarantees on the worst-case rather than the usual QoS average-case requirements for HPC applications; (2) improving the resource management policies; (3) optimizing non-functional metrics (e.g., energy consumption).

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, 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 [68] – 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 [50]. 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 with respect to non-time-sensitive applications.

V. 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 VI, we show experimental results collected by executing benchmark applications on a real HPC infrastructure.

A. GENERAL CONSIDERATIONS

As discussed in Section I, 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 [40], [47]. 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, as previously defined in Section IV-A. 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 degree 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 general-purpose operating systems and other background services are a further source of variability, even if the real-time task is set at the maximum priority. This effect is also observed even applying the real-time PREEMPT_RT patch of Linux [15], [51]. Third, modern COTS motherboards and CPUs may run unpredictable, and non-maskable interrupt routines, to manage the occurrence of low-level hardware events [34]. 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.

B. EVT HYPOTHESES

In order to apply the pWCET and EVT process to the HPC scenario, the hypotheses described in Section III-D 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.

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. This hypothesis does not depend on the HPC infrastructure itself, but on the user ability to stress all the possible application behaviors.

THE IID HYPOTHESIS

The second hypothesis is iid. As we have already explained, it 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 a 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. 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. The inability to exploit the caches across the job executions (but only inside each job) removes the necessity of custom architectures implementing randomized caches: the overall job's execution time is independent on processor caches. 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.* [65], 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 [12], EVT can correctly estimate pWCET distribution even in the presence of moderate dependency.

THE MDA HYPOTHESIS

Finally, we consider 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 [61]. 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.

C. HETEROGENEOUS HARDWARE AND PREDICTABILITY

Despite the increasing complexity of the HPC infrastructures, discussed in II-C, 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.

The use of GPUs and hardware accelerators has become an opportunity to increase the total throughput by leveraging the device's hardware architecture, which is specifically designed for parallel computing. A dedicated memory hierarchy and a minimal and efficient interconnect intrinsically minimize the resource contention side-effects. However, since they mainly focus on throughput, the overall latency is not often considered in GPU hardware design [3]. On the software side, GPU kernels are usually smaller and simpler than their CPU counterparts to exploit the parallel programming models effectively. Predictable GPU scheduling algorithms based on priority and isolation are available [33], and custom accelerators may not even require a scheduler. These unbalanced hardware and software effects require proper investigations to assess the overall predictability of the heterogeneous hardware.

The research on static WCET analyses for GPUs has been recently very active. While the well-defined parallel model and the simple kernel software make attractive the use of traditional WCET analysis [42], recent studies showed how GPUs hide many details that can negatively affect the execution time [2] and that it is necessary to develop dedicated WCET analyses [28]. In the opposite view, other works noticed a substantial improvement in the real-time capability of the heterogeneous solution [67], [69]. A hybrid static and measurement-based solution to WCET estimation for GPU was proposed by Betts *et al.* [8] in 2013, while the first pWCET approach was presented in 2014 by Berezovskyi *et al.* [6] and its extension [5] in 2016. The authors mainly analyzed the iid hypothesis and computed the pWCET estimation, obtaining good results in the satisfaction of the iid hypothesis and a satisfactory pWCET estimation. However, no indications were provided regarding the MDA hypothesis, and the considered workload was a rather simple custom benchmark. None of the previous works considered the interaction of CPU–GPU, which is an essential component of our analysis for HPC because it is a source of unpredictable overheads [43].

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.

VI. EXPERIMENTAL EVALUATION

The previous section focused on a qualitative empirical analysis of the applicability of EVT to the HPC domain. The goal of this section 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 tightness and usability of the estimated pWCET distributions. This section is structured as follows: the first two subsections describe the experimental setup and the analyzed metrics; the last two subsections describe the results and discuss the applicability of probabilistic real-time techniques to HPC.

A. BENCHMARKS AND PLATFORM

For the tests, we used the NAS Parallel Benchmarks (NPB) suite, which is a very common choice for HPC performance estimation [4]. For each benchmark application different implementations are available: MPI, OpenMP and CUDA [18]. Among these, we selected the three pseudo-applications[6]: `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 time* via the Time Stamp Counter register, with a theoretical resolution of 1ns.[7]

We ran the benchmarks on the computing nodes of the GALILEO HPC cluster located at the CINECA

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

[7]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.

TABLE 2. Execution conditions of the benchmarks running during the experimental evaluation.

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

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 2. We refer to the C class of NPB benchmarks with the term *Extended Data Size*.

For each execution condition, each benchmark was executed 500 times by using the default resource manager SLURM [68], 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. According to [54], the statistical power of the KS test and AD test is, respectively, greater than $1-10^{-3}$ and greater than $1-10^{-8}$. This guarantees that the GoF test performed after the pWCET estimation has a low false-negative probability.

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.

B. 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.

GENERAL STATISTICAL INFORMATION

The first set of computed metrics is the traditional statistical measures: average and standard deviation. Instead of providing these direct absolute values that would have had little significant informative content, we computed the *coefficient of variation* (CV) in order to make it possible to compare different benchmarks and different execution conditions. 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 measurable 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.

COMPLIANCE WITH EVT HYPOTHESES

The second set is composed of four metrics representing three 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. Finally, the Kolmogorov-Smirnov (KS) and the Anderson-Darling (AD) tests were used to check the MDA hypothesis. The choice of these tests was made based on previous state-of-the-art works [53].

TIGHTNESS OF THE pWCET DISTRIBUTION

The last set of metrics contains the three parameters of the estimated GEV distribution. 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 III-E. 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: $\mathrm{INC} = 100\frac{\mathrm{WCET}-\mathrm{WCOT}}{\mathrm{WCOT}}$. 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.

C. 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

TABLE 3. Result of the statistical tests used to check the iid EVT hypotheses, for each benchmark and for each execution condition of Table 2. A statistical test fails when the value of its statistic is greater than the critical value (cells in red).

	Stationarity KPSS			Short-range dep. BDS			Long-range dep. R/S		
Crit. Value	0.46149			1.9600			1.7470		
Scenario	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

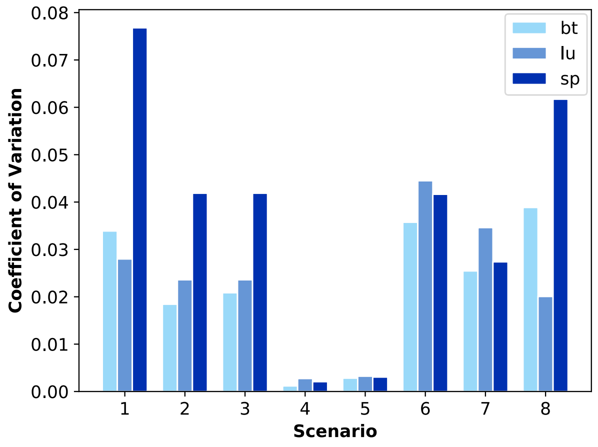


FIGURE 5. The Coefficient of Variation for the three NAS benchmarks running with the different execution conditions of Table 2. These data are not affected by the absolute value of execution times.

TABLE 4. Result of the statistical tests used to check the MDA hypothesis, for each benchmark and for each execution condition of Table 2. A statistical test fails when the value of its statistic is greater than the critical value (cells in red).

	KS			AD		
Crit. Value	0.2716			variable (≈2.50)		
Scenario	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. The estimated pWCET for each benchmark and for each execution condition of Table 2. 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.

	ξ			WCET/WCOT ratio		
Scenario	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%

data of such experiments is made available online for any further analyses.[8]

GENERAL STATISTICAL INFORMATION

Figure 5 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, as we hypothesize in V-C. 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`.

Finding 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 3 and Table 4. 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, as discussed in Section V-B. Previous works on cyber-physical systems showed similar results [26].

Finding 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 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

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

TABLE 6. Summary of the qualitative and quantitative analysis performed related to the satisfaction of EVT hypotheses.

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 makes this hypothesis valid, 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.

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 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 3, are definitely compliant with the dependency hypothesis.

Finding 4: The two dependency properties are affected by the data locality of the SAN.

Finding 5: If data locality is removed, the GPU scenarios pass the dependency tests 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 goodness-of-fit tests in Table 4. 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 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. 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 no. 8 benchmark `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 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.

D. 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 system 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 6 summarizes, for each hypothesis, the qualitative discussions, the experimental method and the results obtained on the real platforms.

VII. FUTURE WORKS AND CONCLUSION

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.

A. 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; a randomized behavior, in a similar fashion of CPS cache randomization, can also be considered for reducing the SAN data locality;
- 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.

B. 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, as shown in Section II.

In this paper, we proposed to exploit the probabilistic real-time theory, in particular the Measurement-Based Probabilistic Timing Analysis (MBPTA), by applying it to 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. This paper opens several possible future works. Possible improvements in hardware and software architectures can be considered with the goal of fulfilling the EVT hypotheses. In addition, 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. Finally, the effect of applications themselves can be the subject of possible future studies: how to guarantee the representativity hypothesis and how the interaction between different processes of the same job affects the pWCET analysis?

REFERENCES

[1] G. Agosta, W. Fornaciari, G. Massari, A. Pupykina, F. Reghenzani, and M. Zanella, "Managing heterogeneous resources in HPC systems," in *Proc. 9th Workshop 7th Workshop Parallel Program. RunTime Manage. Techn. Manycore Archit. Design Tools Archit. Multicore Embedded Comput. Platforms PARMA-DITAM*, 2018, pp. 7–12.

[2] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "GPU scheduling on the NVIDIA TX2: Hidden details revealed," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2017, pp. 104–115.

[3] M. Andersch, J. Lucas, M. A. LvLvarez-Mesa, and B. Juurlink, "On latency in GPU throughput microarchitectures," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2015, pp. 169–170.

[4] D. Bailey, T. Harris, W. Saphir, R. V. D. Wijngaart, A. Woo, and M. Yarrow, "The NAS parallel benchmarks 2.0," NASA Ames Research Center, Washington, DC, USA, Tech. Rep. NAS-95-020, 1995.

[5] K. Berezovskyi, F. Guet, L. Santinelli, K. Bletsas, and E. Tovar, "Measurement-based probabilistic timing analysis for graphics processor units," in *Architecture of Computing Systems*, F. Hannig, J. M. P. Cardoso, T. Pionteck, D. Fey, W. Schröder-Preikschat, and J. Teich, Eds. Cham, Switzerland: Springer, 2016, pp. 223–236.

[6] K. Berezovskyi, L. Santinelli, K. Bletsas, and E. Tovar, "WCET measurement-based and extreme value theory characterisation of CUDA kernels," in *Proc. 22nd Int. Conf. Real-Time Netw. Syst. - RTNS*, 2014, pp. 279–288.

[7] G. Bernat, A. Colin, and S. M. Petters, "WCET analysis of probabilistic hard real-time systems," in *Proc. 23rd IEEE Real-Time Syst. Symp. RTSS*, Dec. 2002, pp. 279–288.

[8] A. Betts and A. Donaldson, "Estimating the WCET of GPU-accelerated applications using hybrid analysis," in *Proc. 25th Euromicro Conf. Real-Time Syst.*, Jul. 2013, pp. 193–202.

[9] E. Castillo, A. S. Hadi, N. Balakrishnan, and J.-M. Sarabia, *Extreme Value and Related Models With Applications in Engineering and Science*. Hoboken, NJ, USA: Wiley, 2005.

[10] F. J. Cazorla, L. Kosmidis, E. Mezzetti, C. Hernandez, J. Abella, and T. Vardanega, "Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–35, Feb. 2019.

[11] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *Proc. 24th Euromicro Conf. Real-Time Syst.*, Jul. 2012, pp. 91–101.

[12] R. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for real-time systems," *Leibniz Trans. Embedded Syst.*, vol. 6, no. 1, p. 3, 2019.

[13] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 1–44, Oct. 2011.

[14] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic schedulability analysis techniques for real-time systems," *LITES, Leibniz Trans. Embedded Syst.*, vol. 6, no. 1, pp. 1–53, May 2019.

[15] D. B. D. Oliveira, D. Casini, R. S. D. Oliveira, and T. Cucinotta, "Demystifying the real-time linux scheduling latency," in *Proc. 32nd Euromicro Conf. Real-Time Syst. (ECRTS 2020)*, vol. 165, M. Völp, Ed. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 20200, pp. 9:1–9:23.

[16] K. Dichev and A. Lastovetsky, "Optimization of collective communication for heterogeneous HPC platforms," in *High-Performance Computing on Complex Environments*, May 2014, pp. 95–114, doi: 10.1002/9781118711897.

[17] P. Dinda, X. Wang, J. Wang, C. Beauchene, and C. Hetland, "Hard real-time scheduling for parallel run-time systems," in *Proc. 27th Int. Symp. High-Perform. Parallel Distrib. Comput.*, Jun. 2018, pp. 14–26.

[18] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow, "The NAS parallel benchmarks 2.0," NASA Ames Res. Center, Moffett Field, CA, USA, NAS Tech. Rep. NAS-95-020, Dec. 1995.

[19] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, "Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator," *Ann. Math. Statist.*, vol. 27, no. 3, pp. 642–669, Sep. 1956.

[20] S. Edgar and A. Burns, "Statistical analysis of WCET for scheduling," in *Proc. 22nd IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2001, pp. 215–224.

[21] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. 38th Annu. Int. Symp. Comput. Archit. - ISCA*, 2011, pp. 365–376.

[22] R. A. Fisher and L. H. C. Tippett, "Limiting forms of the frequency distribution of the largest or smallest member of a sample," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 24. Cambridge, U.K.: Cambridge Univ. Press, 1928, pp. 180–190.

[23] J. Flich *et al.*, "MANGO: Exploring manycore architectures for next-GeneratiOn HPC systems," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2017, pp. 478–485.

[24] W. Fornaciari *et al.*, "Reliable power and time-constraints-aware predictive management of heterogeneous exascale systems," in *Proc. 18th Int. Conf. Embedded Comput. Syst. Archit., Modeling, Simulation - SAMOS*, 2018, pp. 187–194.

[25] M. Fusi, F. Mazzocchetti, A. Farres, L. Kosmidis, R. Canal, F. J. Cazorla, and J. Abella, "On the use of probabilistic worst-case execution time estimation for parallel applications in high performance systems," *Mathematics*, vol. 8, no. 3, p. 314, Mar. 2020.

[26] F. Guet, L. Santinelli, and J. Morio, "On the representativity of execution time measurements: Studying dependence and multi-mode tasks," in *Proc. 17th Int. Workshop Worst-Case Execution Time Anal. (WCET)*, vol. 57, J. Reineke, Ed. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2017, pp. 3:1–3:13.

[27] 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 *Proc. 53rd Annu. Design Autom. Conf. - DAC*, 2016, pp. 1–6.

[28] Y. Huangfu and W. Zhang, "Static WCET analysis of GPUs with predictable warp scheduling," in *Proc. IEEE 20th Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2017, pp. 101–108.

[29] H. Hussain, S. U. R. Malik, A. Hameed, S. U. Khan, G. Bickler, N. Min-Allah, M. B. Qureshi, L. Zhang, W. Yongji, N. Ghani, and J. Kolodziej, "A survey on resource allocation in high performance distributed computing systems," *Parallel Comput.*, vol. 39, no. 11, pp. 709–736, 2013.

[30] J. Jalle, L. Kosmidis, J. Abella, E. Quinones, and F. J. Cazorla, "Bus designs for time-probabilistic multicore processors," in *Proc. Design, Autom. Test Eur. Conf. Exhibi. (DATE)*, 2014, pp. 1–6.

[31] E. D. Jensen, C. D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time operating systems," in *Proc. RTSS*, vol. 85, Dec. 1985, pp. 112–122.

[32] S. Jimenez 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 Syst. Lett.*, vol. 9, no. 3, pp. 69–72, Sep. 2017.

[33] S. Kato, K. Lakshmanan, R. Rajkumar, and Y. Ishikawa, "Timegraph: Gpu scheduling for real-time multi-tasking environments," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2011, p. 2.

[34] W. Kau, J. H. Cornish, Q. A. Qureshi, and S. A. Wichman, "Method and apparatus for handling system management interrupts (SMI) as well as, ordinary interrupts of peripherals such as pcmcia cards," U.S. Patent 6 112 273, Aug. 29, 2000.

[35] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar, "Bounding memory interference delay in COTS-based multi-core systems," in *Proc. IEEE 19th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2014, pp. 145–154.

[36] R. Kirner and P. Puschner, "Obstacles in worst-case execution time analysis," in *Proc. 11th IEEE Int. Symp. Object Compon.-Oriented Real-Time Distrib. Comput. (ISORC)*, May 2008, pp. 333–339.

[37] L. Kosmidis, J. Abella, E. Quinones, and F. J. Cazorla, "A cache design for probabilistically analysable real-time systems," in *Proc. Design, Autom. Test Eur. Conf. Exhibi. (DATE)*, Mar. 2013, pp. 513–518.

[38] L. Kosmidis, C. Curtsinger, E. Quinones, J. Abella, E. Berger, and F. J. Cazorla, "Probabilistic timing analysis on conventional cache designs," in *Proc. Design, Autom. Test Eur. Conf. Exhibi. (DATE)*, Mar. 2013, pp. 603–606.

[39] 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 *Proc. 51st Annu. Design Autom. Conf. Design Autom. Conf. - DAC*, 2014, pp. 1–6.

[40] T. Leng, R. Ali, J. Hsieh, V. Mashayekhi, and R. Rooholamini, "An empirical study of hyper-threading in high performance computing clusters," *Linux HPC Revolution*, vol. 45, 2002.

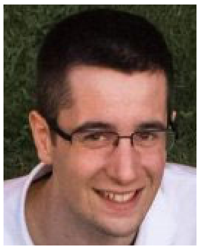
[41] G. Lima, D. Dias, and E. Barros, "Extreme value theory for estimating task execution time bounds: A careful look," in *Proc. 28th Euromicro Conf. Real-Time Syst. (ECRTS)*, Jul. 2016, pp. 200–211.

[42] B. Lisper, "Towards parallel programming models for predictability," in *Proc. 12th Int. Workshop Worst-Case Execution Time Anal.*, vol. 23, T. Vardanega, Ed. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 48–58. .

[43] D. Lustig and M. Martonosi, "Reducing GPU offload latency via fine-grained CPU-GPU synchronization," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 354–365.

[44] M. Martínez-Díaz and F. Soriguera, "Autonomous vehicles: Theoretical and practical challenges," *Transp. Res. Procedia*, vol. 33, pp. 275–282, Jan. 2018.

[45] G. Massari, A. Pupykina, G. Agosta, and W. Fornaciari, "Predictive resource management for next-generation high-performance computing heterogeneous platforms," in *Proc. Embedded Comput. Syst., Archit., Modeling, Simulation*, D. N. Pnevmatikatos, M. Pelcat, M. Jung, Eds. Cham, Switzerland: Springer, 2019, pp. 470–483.
[46] S.-M. Park and M. Humphrey, "Predictable high-performance computing using feedback control and admission control," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 396–411, Mar. 2011.
[47] L. Porter, M. A. Laurenzano, A. Tiwari, A. Jundt, W. A. Ward, Jr., R. Campbell, and L. Carrington, "Making the most of SMT in HPC: System- and application-level perspectives," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 4, pp. 1–26, Jan. 2015.
[48] E. Quiñones, E. D. Berger, G. Bernat, and F. J. Cazorla, "Using randomized caches in probabilistic real-time systems," in *Proc. 21st Euromicro Conf. Real-Time Syst.*, Jul. 2009, pp. 129–138.
[49] P. Radojković, S. Girbal, A. Grasset, E. Quiñones, S. Yehia, and F. J. Cazorla, "On the evaluation of the impact of shared resources in multithreaded COTS processors in time-critical environments," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 4, pp. 1–25, Jan. 2012.
[50] F. Reghenzani, G. Massari, and W. Fornaciari, "A probabilistic approach to energy-constrained mixed-criticality systems," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.
[51] F. Reghenzani, G. Massari, and W. Fornaciari, "The real-time linux kernel: A survey on PREEMPT_RT," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 18:1–18:36, Feb. 2019.
[52] F. Reghenzani, G. Massari, and W. Fornaciari, "Probabilistic-WCET reliability: Statistical testing of EVT hypotheses," *Microprocessors Microsyst.*, vol. 77, Sep. 2020, Art. no. 103135.
[53] F. Reghenzani, G. Massari, W. Fornaciari, and A. Galimberti, "Probabilistic-WCET reliability: On the experimental validation of EVT hypotheses," in *Proc. Int. Conf. Omni-Layer Intell. Syst.*, May 2019, pp. 229–234.
[54] F. Reghenzani, L. Santinelli, and W. Fornaciari, "Why statistical power matters for probabilistic real-time: Work-in-progress," in *Proc. Int. Conf. Embedded Softw. Companion - EMSOFT*, 2019, p. 2.
[55] F. Reghenzani, L. Santinelli, and W. Fornaciari, "Dealing with uncertainty in pWCET estimations," *ACM Trans. Embedded Comput. Syst.*, vol. 19, no. 5, pp. 1–23, Oct. 2020.
[56] *Software Considerations in Airborne Systems and Equipment Certification*, Standard RTCA/EUROCAE, DO-178C, Jan. 2012.
[57] L. Santinelli, F. Guet, and J. Morio, "Revising measurement-based probabilistic timing analysis," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2017, pp. 199–208.
[58] S. Schlagkamp and J. Renker, "Acceptance of waiting times in high performance computing," in *Proc. HCI Int. Posters' Extended Abstr.*, C. Stephanidis, Ed. Cham, Switzerland: Springer, 2015, pp. 709–714.
[59] 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*, vol. 35, no. 4, pp. 26–36, Jul. 2015.
[60] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *High Performance Computing for Computational Science—VECPAR*, J. M. L. M. Palma, M. Daydé, O. Marques, and J. C. Lopes, Eds. Berlin, Germany: Springer, 2011, pp. 1–25.
[61] T. Shimura, "Discretization of distributions in the maximum domain of attraction," *Extremes*, vol. 15, no. 3, pp. 299–317, Sep. 2012.
[62] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," *J. Grid Comput.*, vol. 14, no. 2, pp. 217–264, Jun. 2016.
[63] D. Skinner and W. Kramer, "Understanding the causes of performance variability in HPC workloads," in *Proc. IEEE Int. IEEE Workload Characterization Symp.*, Oct. 2005, pp. 137–149.
[64] M. Slijepcevic, L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla, "Time-analysable non-partitioned shared caches for real-time multicore systems," in *Proc. The 51st Annu. Design Autom. Conf. Design Autom. Conf. - DAC*, 2014, pp. 1–6.
[65] D. Unat *et al.*, "Trends in data locality abstractions for HPC systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 3007–3020, Oct. 2017.
[66] Y.-Q. Wang, C.-R. Zhai, Y.-H. Zhang, and W. Gao, "A new GPS deformation monitoring algorithm applied to donghai bridge," *J. Shanghai Jiaotong Univ. (Sci.)*, vol. 13, no. 2, pp. 216–220, Apr. 2008.
[67] Y. Xiang and H. Kim, "Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2019, pp. 392–405.
[68] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Proc. Workshop Job Scheduling Strategies Parallel Process.* Berlin, Germany: Springer, 2003, pp. 44–60.
[69] H. Zhou, S. Bateni, and C. Liu, "S^{3}DNN: Supervised streaming and scheduling for GPU-accelerated real-time DNN workloads," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2018, pp. 190–201.



FEDERICO REGHENZANI (Member, IEEE) received the master's degree in September 2016. He is currently pursuing the Ph.D. degree with the Politecnico di Milano. He is also working on HPC and embedded computing. His research interests include resource management on heterogeneous platforms, embedded Linux systems, and real-time analyses, in particular on probabilistic WCET estimation and mixed-criticality systems.

GIUSEPPE MASSARI is currently a Fellow Researcher with the Politecnico di Milano, with over 50 publications. His research interests include run-time resource management techniques for embedded and HPC computing platforms, based on heterogeneous processing units (multi-core, many-core processors, GPUs, FPGA, and HW accelerators). He is also the Head Developer of the open-source project BOSP: The Barbeque Runtime Resource Manager.

WILLIAM FORNACIARI (Senior Member, IEEE) is currently with the Politecnico di Milano, Italy. He has published six books and over 300 articles on journals and conferences, collecting six best paper awards, one certificate of appreciation from IEEE. He holds three international patents on low power design. He has been involved in 19 EU- funded international projects. He won the 2016 HiPEAC Technology Transfer Award. In 2019, he won the Switch-to-Product competition. His research interests include multi/many core architectures, embedded systems, low power design of SW and HW, run time resource management, mixed-criticality systems, and thermal management.