HAL
open science

Statistical Analysis for Shared Resources Effects with Multi-Core Real-Time Systems

Julien Durand, Youcef Bouchebaba, Luca Santinelli

To cite this version:

Julien Durand, Youcef Bouchebaba, Luca Santinelli. Statistical Analysis for Shared Resources Effects with Multi-Core Real-Time Systems. 13th IEEE International Symposium on Embedded Multicore/Many-Core Systems-on-Chip, MCSoC 2019, Oct 2019, SINGAPOUR, Singapore. 10.1109/MCSoC.2019.00058 . hal-02904408

HAL Id: hal-02904408

https://hal.science/hal-02904408v1

Submitted on 22 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Statistical analysis for shared resources effects with multi-core real-time systems

Julien Durand
P E&DS SE FR AET
CPT
Toulouse, France
julien.durand@continental-corporation.com

Youcef Bouchebaba, Luca Santinelli
DTIS
ONERA
Toulouse, France
{*youcef.bouchebaba,luca.santinelli*}*@onera.fr*

***Abstract*—Today's multi-core and many-core COTS platforms make available a large amount of computational resource for real-time applications. As they aim at increasing performance for real-time, their challenges are the guarantees for timing constraints. Real-time modeling and analysis are thus facing shared resources, optimization mechanisms, and sophisticated functionalities which all combine into complex system dynamics that are extremely costly to characterize. This paper proposes a measurement-based approach and a statistical analysis applied to define average and worst-case models to task executions under different possible execution conditions. The framework is formalized and then used to investigate different families of shared resources interference effects occurring on multi-core platforms; such effects are quantified with statistical metrics applied to measurements of tasks execution times. The focus of the work is on effects due to shared memories within the NXP T4240 multi-core platform and the PikeOS hypervisor. A set of experiments is conducted to validate the framework proposed.**

I. Introduction and motivation

The continuous quest for high performance is pushing real-time embedded systems for implementations with multi-core and many-core COTS platforms. Such platforms embed shared memories and cache hierarchies, communication networks and DDR controllers, etc. which certainly improves performance, but also makes more difficult guaranteeing predictability.

Defining complete models of today's real-time systems is extremely complex or overly costly [1], [2]. Task execution times are largely affected by the variety of execution conditions which can happen at runtime. The numerous interference on shared resources may end up into difficult to predict variations of tasks execution time, depending on the actual execution condition and when the interference takes place [3], [4]. Modeling interference effects and guaranteeing such epistemic variability [5] is the main challenge when applying multi-core into real-time systems.

Figure 1 shows an example of a real-time application executing on a multi-core platform. The maximum measured execution time of one of the application tasks is plotted under different execution conditions, CaseXs. The conditions are obtained varying the memory requirements of the whole application, thus the effects of interference from shared memory are considered. By increasing the memory requirements, the interference increases and the maximum measured execution time changes significantly e.g., caseF-caseB where execution time increases 5 times for memory requirement increase of 3.5 times: *the impact of shared resources on system performance and determinism cannot be neglected*.

Measurement-Based Probabilistic Timing Analysis (MBPTA) [6], [7], [8], [9] is a probabilistic timing analysis approach emerging as an alternative to the more classical deterministic static timing analysis. MBPTA proposes to estimate Worst-Case Execution Time (WCET) models from measurements of execution time. For it, MBPTA does not need a model of the platform and of its interactions [7], but it infers empirical worst-case models from the measures of the actual behaviors.

MBPTA aims at reducing pessimism of classical deterministic WCETs with models that are closer to the normal/expected system behaviors [8]. That goal is achieved with probabilistic models (probabilistic Worst-Case execution times - pWCETs) that are worst-case distributions able to upper bounds task execution time under every possible execution conditions. The main problem of MBPTA remains guaranteeing exploration of every possible execution conditions such that the estimated WCETs can be guaranteed safe worst case models.

Contributions: With this paper we propose a timing analysis, based on measurements and statistics, which is able to develop average and worst-case models for task execution time. The use of statistics is to have a rational and formal set of metrics to derive statistical models for task executions; the statistical representations developed are to enhance knowledge of multi-core system dynamics. We focus on the effects that shared memories have on execution time within multi-core platforms, i.e. interference. The statistical analysis proposed quantifies the interference effects under different execution conditions, and the interference representations provided can be later applied into development guidelines

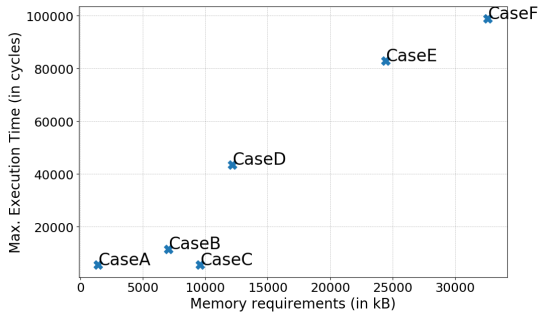


Figure 1: Task maximum execution time vs memory requirements.

to enforce determinism as well as to improve performance. The statistical approach we propose does not rely on system models, as for static timing analyses [10]; thus, it will never be able to provide safety guarantees to the representations obtained. Instead, it relies on easy-to-obtain measurements, and it builds accurate average/nominal and worst-cases statistical models of tasks executions that can be applied during the design and development of multi-core embedded systems. For average modeling, this paper proposes the theory which is then implemented within the MBPTA tool DIAGXTRM; for the worst-case modeling, the paper applies the theory already developed in [11], and also implemented in DIAGXTRM. The analysis is validated with an experimental setup on the NXP T4240 multi-core running the PikeOS hypervisor. Interference are examined by characterizing the impact of tasks placement e.g., use of multiple CPUs, and partitioning capabilities. Also, a comparison with related work is carried out.

Organization of the paper: Section II presents the statistical analysis we propose with notions and the metrics applied. Section III specifies hardware and software set up for the multi-core case study considered. Section IV shows some modeling results on interference generated by the concurrent use of shared resource. It is detailed the statistical analysis for memory utilization under different execution conditions. Section V compares our results with related work. Section VI is for conclusions and future work.

II. Statistical Analysis

Today's multi-core and many-core real-time systems exploit variable behavior due to complex interactions happening at runtime, and due to different possible input conditions. The task execution time can vary from one execution to another, and the objective of MBPTA, and more in general of statistical approaches to timing analysis, is to model such epistemic variability [5].

The execution of a real-time task τ can be modeled with the Execution Time (ET) which can be expressed by the number of processor cycles from the beginning to the end of its execution. Each instance of τ can take its own ET, depending on the events happening at runtime before and during the task execution.

A trace with $\mathcal{T} = \{C_j | j \in [1, n]\}$ is a collection of ET measurements C_j; the Execution Time Profile (ETP) $\mathcal{C}$ of a task is the discrete random variable defined on the finite support $\Omega_{\mathcal{C}}$ of all possible ET values $C_{(k)}$ for the task and task instances executions. $\Omega_{\mathcal{C}} = \left(C_{(k)}\right)_{k \in [1,N]}$ and $C_{(k)} \in \mathcal{T}$. n is the size of the trace while N is the number of different measured ETs.

Assuming the ETP $\mathcal{C}$ as discrete distribution, the probability distribution function (pdf) $\mathsf{pdf}_{\mathcal{C}}$ describes the probability of happening of certain events C_i from the random variable $\mathcal{C}$, $P(\mathcal{C} = C_i)$. $\mathsf{cdf}_{\mathcal{C}}$ denotes the cumulative distribution function (cdf) representation of $\mathcal{C}$, $\mathsf{cdf}_{\mathcal{C}}(C_i) = P(\mathcal{C} \leq C_i) = \sum_0^{C_i} \mathsf{pdf}_{\mathcal{C}}(x)$, while the inverse cumulative distribution function (icdf) $\mathsf{icdf}_{\mathcal{C}}(C_i)$ outlines the exceedance thresholds, $\mathsf{icdf}_{\mathcal{C}}(C_i) = P(\mathcal{C} \geq C_i)$ as the probability of having execution time greater than C_i, $\mathsf{icdf}_{\mathcal{C}}(C_i) = 1 - \sum_0^{C_i} \mathsf{pdf}_{\overline{\mathcal{C}}}(x)$. As a reminder, the definition of a distribution is such that $\sum_0^{\infty} \mathsf{pdf}_{\overline{\mathcal{C}}}(x) = 1$.

The pWCET $\overline{\mathcal{C}}$ of a task τ is defined as the worst-case estimate distribution that upper-bounds any possible execution time the task can exhibit [12]. $\overline{\mathcal{C}}_i$ is composed of multiple values, each with a probability associated[1]. Those values are WCET thresholds, and the probability associated to each of them is the probability/confidence for that value of being task WCET: the probability for that value not being overcome at runtime.

pWCETs and WCET thresholds are generalization of classical deterministic WCETs. Having multiple WCET thresholds does not mean that a task has or need multiple WCETs; instead, they allow a more fine grained representation of what happens to task worst-case execution conditions: the thresholds parametrize execution conditions, while probabilities quantifies impacts of such execution conditions on task behavior.

Assuming the pWCET $\overline{\mathcal{C}}$ as continuous distribution, the pdf $\mathsf{pdf}_{\overline{\mathcal{C}}}$ it is such that $P(C_1 \leq \overline{\mathcal{C}} \leq C_2) = \int_{C_1}^{C_2} \mathsf{pdf}_{\overline{\mathcal{C}}_i}(x)dx$ and $\int_0^{\infty} \mathsf{pdf}_{\overline{\mathcal{C}}_i}(x)dx = 1$; the cdf representation is such that $\mathsf{cdf}_{\overline{\mathcal{C}}}(C) = P(\overline{\mathcal{C}} \leq C) = \int_0^{C} \mathsf{pdf}_{\overline{\mathcal{C}}}(x)dx$; the icdf representation is $\mathsf{icdf}_{\overline{\mathcal{C}}}(C) = P(\overline{\mathcal{C}} \geq C) = 1 - \int_0^{C} \mathsf{pdf}_{\overline{\mathcal{C}}}(x)dx$.

To note that ETPs are obtained measuring task ET, thus are discrete empirical distributions from system that have discrete time evolution, i.e. ticks. Instead, pWCETs are estimated from MBPTA approaches and Extreme Value Theory (EVT) statistics, thus are continuous distributions. In this work, the MBPTA version applied is the one from DIAGXTRM [11].

[1] In the following, calligraphic letters are used to represent distributions or traces, while non-calligraphic letters are for scalars or deterministic values.

A. Average modeling

Some statistics are used to characterize the average behavior of task executions; they apply to $\mathcal{T}$ and $\mathcal{C}$.

Expected behavior. The mean $\mu(\mathcal{T}) \stackrel{def}{=} \frac{1}{n}\sum_{j=1}^{n} C_j$, is the mean value of a trace $\mathcal{T}$ and its associated discrete empirical random variable $\mathcal{C}$. μ contributes to define the nominal behavior of a trace. The moving average $\overline{\mu}(\mathcal{T}, k)$ applied here is defined as an equally weighted average of the sequence of k values, with $k \in [1, n]$. Mode $\bar{m}(\mathcal{T})$ is the value that appears most often in $\mathcal{T}$; median $\tilde{m}(\mathcal{T})$ is the value separating the higher half from the lower half of a data sample $\mathcal{T}$.

Variability. the standard deviation $\sigma(\mathcal{T}) \stackrel{def}{=} \sqrt{\frac{1}{n}\sum_{j=1}^{n}(C_i - \mu)^2}$, contributes defining measurements variability around the nominal behavior μ.

Extreme behaviors. The minimum $min(\mathcal{T}) \stackrel{def}{=} \min_{1\leq j\leq n}(C_j)$ defines the minimum measurement in $\mathcal{T}$; the maximum $max(\mathcal{T}) \stackrel{def}{=} \max_{1\leq j\leq n}(C_j)$ defines the maximum measurement in $\mathcal{T}$. min and max are extreme cases of the measured behavior. Also, we apply the quantiles in order to have a better representation of the worst-case measurements. In particular, the quantiles aims at detailing the trace in specific range of behaviors toward the tail of the $\mathcal{C}$ distribution. We use the quantile at 0.8, $D8$, and the quantile at 0.9, $D9$ to focus on the ETP tail. Together with σ, $D8$ and $D9$ help describing ET variability.

The analysis is conceived in a compositional manner such that other metrics can and will be added in future work to help better defining average models. All the metrics proposed are implemented into DIAGXTRM. Figure 2 is an example of average statistics applied to characterize a distribution of measurements.

B. Trends with time series and worst-case modeling

Traces of measurements can be seen as time series since the ETs taken and kept in time order. From ET time series it is possible to extract properties which tells about the behavior of the system under specific execution conditions. Those properties are used by DIAGXTRMfor the applicability of the EVT, and in this work, they are applied for the first time to represent real-time systems dynamics.

Stationarity and identical distribution hypothesis - $h_{1.1}$ and $h_{1.2}$. A trace $\mathcal{T}$ is strictly stationary if every subset of ETs has an ETP which follows the same probabilistic law. Statistical tests and moving average metrics are applied for checking if $\mathcal{T}$ has not a non-null deterministic trend and has not a non-null random walk, thus it follows a well defined reference distribution law. Stationarity is also a necessary condition for the identical distribution hypothesis: measurements are identically distributed from the same probabilistic law. For the stationary and identical distribution hypotheses, the KPSS test [11] is applied.

Independence - $h_{2.1}$ and $h_{2.2}$. The notion of statistical dependence could instantiate into short range dependence or long range dependence. The short range independence hypothesis $h_{2,1}$, equivalently local independence, focuses on the relationship between close-in-time measurements, i.e. close-by within the trace. It is for verifying if there are dependence effects which shortly affect task executions e.g., locality from cache memories. A valuable short range dependence test applied here is the Brock Dechert Scheinkman test [11].

The long range independence $h_{2,2}$, equivalently extremal independence, focuses on the relationship between far-in-time measurements: it is used to verify if there exist trends between measurements separated in time. Statistical estimators exist to define how far away are the dependent measurements or the distance between dependent execution times measurements. Possible estimators comes from [11].

Maximum domain of attraction (MDA) - h_3. MDA applies to evaluate if the input ETPs – $\mathcal{T}$ or $\mathcal{C}$ – belong to the domain of attraction of the EVT possible resulting distributions. This is applied to validate the reliability of the resulting pWCET model of the worst-case behavior. The Cramer Von Mises criterion (CVM) detects whether the measurements come from a chosen distribution, thus it verifies the validity of the MDA hypothesis [11]. The MDA will be accurately developed in the future to derive from it, and from the distribution representing ETPs, assumptions on the system behavior.

Convergence - h_4. The convergence hypothesis h_4 defines if the input trace $\mathcal{T}$ is representative of all the events that can happen to the system under the scenario considered. In particular, it defines if the n measurements taken are enough, and that by adding more measurements the empirical distribution ETP does not change; the estimator used for h_4 is the continuous rank probability score [11].

Figure 3 exhibits three examples of properties for ET time series which does not allow EVT application, but also that describe, and quantify, the system execution behaviors. In Figure 3a is represented the non-stationarity case with changing moving average (continuous red line); in Figure 3b the local dependence with clusters of large values; in Figure 3c the extreme dependence with far away clusters of large values that repeats.

The EVT allows estimating uncertainties on the rare events, i.e. where the worst-cases are. It is then the statistic applied to compute pWCET estimates $\overline{\mathcal{C}}$ [6], [7], [8].

The MBPTA version that DIAGXTRMapplies is the so called "generalized EVT" or "weak independence EVT" [11], [6]. In it, the EVT is applicable also in case of certain cases of light dependence (local or extreme dependence) within the time series; the hypotheses that applies are: $\{h_{1,1}, (h_{2,1} \vee h_{2,2}), h_3\}$.

In DIAGXTRM, the fuzzy logic defines the confidence level (cl) of each hypothesis [11], [6]; the EVT is applicable

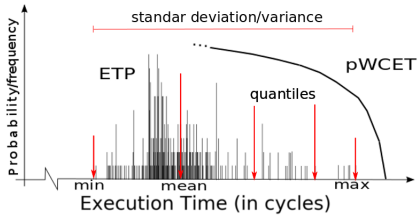
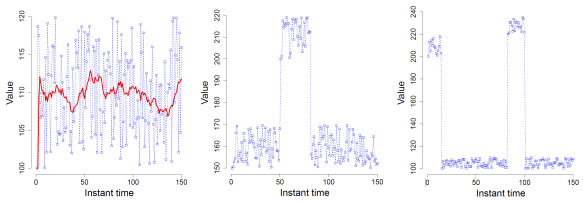


Figure 2: ET measurements and statistics that apply to those for average modeling.



(a) $h_{1,1}$ (b) $h_{2,1}$ (c) $h_{2,2}$

Figure 3: Time series and patterns: non-stationarity, clustering and extreme dependence.

iff:

$$\{cl_{1,1} \geq 1\} \wedge (\{cl_{2,1} \geq 1\} \vee (\{cl_{2,2} \geq 1\}) \wedge \{cl_3 \geq 1\};$$

the greater each confidence level, the more confidence there is on the obtained pWCET to be a safe pWCET estimate. $h_{1,2}$ (more constrained version of stationarity $h_{1,1}$) and h_4, hence $cl_{1,2}$ and cl_4, are not in the classical nor in the generalized EVT definition. They are used hereby to extract properties from inputs time series.

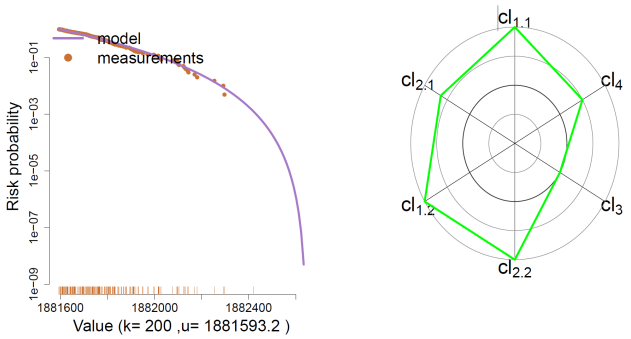
In Figure 4 is an example of pWCET estimates from the input measurements, and the hypothesis testing for EVT applicability with the fuzzy logic. In Figure 4a it is shown that the MBPTA is able to accurately upper bound the ETPs under certain execution conditions; values are the ET in cycles (with k the number of values above a threshold u) and the risk probability is the inverse cumulative probability. In Figure 4b all the hypotheses tested passes, thus the pWCET obtained has good quality and good confidence to be the pWCET.

To note that the pWCET obtained with MBPTA approaches is the probabilistic worst-case execution time only for the scenario considered by the measurements.

III. Experimental Setup

The test case we apply is composed of the NXP T4240 multi-core platform running the PikeOS operating system.

The NXP QorIQ® T4240-QDS is composed of twelve cores grouped into three clusters [13]. The T4240 has three hierarchical levels of cache such that each processor access its own cache level L1 data of 32 KB size and instruction of 32 KB size, and the processors within the same cluster share cache levels L2 and L3, of size 2MB and 512KB respectively and RAM memory controller. The RAM is DDR of size



(a) Worst-case modeling in icdf representation (b) Trend verification and hypothesis testing with the spyderplot representation

Figure 4: Worst-case modeling from measurements and trend verification with DIAGXTRM.

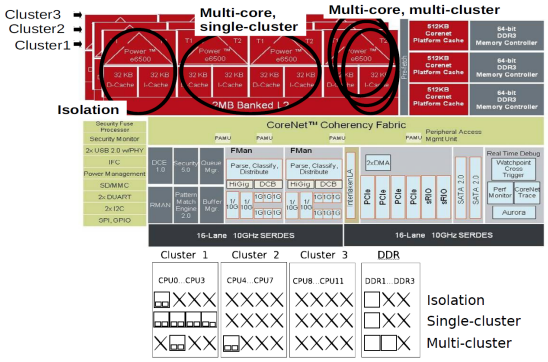


Figure 5: Three scenarios – isolation, single-cluster, and multi-cluster – with shared resource utilization and and interference sources.

8GB per cluster, and is required for memory needs higher than about 2.5MB per processor (at most for full L1 and L2 caches usage). A central bus Corenet connects each cluster's L2 cache to its related L3 cache; this last being connected to one of the three DDR controller.

PikeOS 4.2 is the real-time operating system running onto the platform considered here. PikeOS is based on a preemptive hypervisor which offers a separation kernel. Among many key features of PikeOS, one interesting is the configuration of partitions with time and hardware resources, in order to separate different tasks in time, and resources in space. PikeOS version 4.2 applied here, compared to version 4.1, offers a multi-processor schedule table able to ensure parallelism between processors with a different time partition for each. The time partitions considered have period of $10ms$. L3 cache is inhibited by default with PikeOS 4.2, so we do not consider L3 cache interference effects. In addition, PikeOS offers an Eclipse-based integrated development environment named CODEO, the version of which is 6.2 for PikeOS 4.2. The development language applied is C.

With PikeOS, a high priority partition `Service` is executed on all cores by default. The partition is required to execute drivers and different tools, such as target monitoring, remote application deployment, or timing analysis. Target monitoring allows to obtain an history/log comparable to in-circuit debuggers, but with intrusion from read/write memory and core registers.

Limitations from PikeOS 4.2 multi-core schedule table are that in order to partition core m, it requires to partition each core with at least one task. To comply with that, we have created a task `MX`, named "gap task" which has a minimal footprint in memory consisting on calling the sleep function `sleep()`.

A. Real-time application

In the real-time application implemented, we distinguish between Tasks (`T`) and Interfering (`I`) tasks. `T` task is the part of the application under observation; I tasks are concurrently run with `T`, they are not observed, and they make interference to `T` tasks with shared memory.

To implement `T` and `I` tasks we use the code of an algorithm which relies on writing and reading into two arrays of equal size repeated in a loop, i.e. periodic tasks. The arrays size is the parameter through which changing the memory requirements of the tasks, and thus of the application. The memory footprint of each task is mostly due to data: data cache is mostly solicited.

Each task is allocated on a different core, thus preventing preemption. At each loop of writing and reading of arrays, the ET of `T` is measured via a `gettime()` function. The measurement is a sample of the trace $\mathcal{T}$ printed on the serial port output. 5000 samples/measurements per trace is enough to capture all the behaviors. This has been proved by the convergence hypothesis testing h_4. After each trace, configuration is changed through modification of T and/or Is arrays size. So as to guarantee a homogeneous interference from `I` tasks, they always start executing the algorithm before `T`, and always end after `T`, which meanwhile waits with a `sleep()` function. Start/end synchronization is done through a global variable flag. The first measurements are filtered out of the trace in order to ensure that each sample is measured in similar condition, thus enforcing representativity.

B. Memory requirements and execution scenarios

The interference between tasks from shared memory is obtained changing memory requirement and triggering data transfer through all the memory hierarchy: data are sometimes only transferred throughout the core – L1 caches, sometimes across the same cluster (intra-cluster) – L2 cache, and sometimes across the platform (inter-cluster) – Corenet bus, DDR controller.

We define memory requirements cases with two parameters NumT and NumI, CaseNumT-NumI: NumT corresponds to the size of arrays of `T` task, while NumI correspond to the size `I` tasks composing the application. The correspondence between num (numT or NumI) and the real size s in kB of the arrays is such that $s = 2^{1+\mathtt{num}}$. Thus, the size of each array ranges from $4kB$ (Case1) to $4Mb$ (Case11), in order to cover different cache levels. For instance, case Case6-8 refers to $128kB$ for T, and $512kB$ for the single or multiple `I`s.

Together with memory requirements, we impose execution scenarios. They define task placement, and together with memory requirements cases, they define which part of the memory is impacting interference.

We investigate scenarios: 1) `SM1` in which only task `T0` executes on core 0 - baseline scenario; 2) `SM2` in which task `T1` executed on core 1 and task `I0` executes on core 0; 3) `SM3` with three tasks executing - `T1` on core 1, `I0` on core 0, and `I2` on core 2; 4) `SM4` with four task executing in the 4 core composing a cluster, `T1` `I0`, `I2`, and `I3`.
The scenario for multi-cluster case investigated is `SMC1`, which has 2 different tasks on two core of two different clusters.

With cache disabled, inhibited by PikeOS, we study scenarios: 1) `SMnC1` as the scenario equivalent of `SM1`; 2) `SMnC2` with two tasks executing in separate cores `T1` and `I0`, equivalent to `SM2`; 3) `SMnC3` for three tasks executing, equivalent to `SM2`; 4) `SMnC4` with four tasks executing, each in a separate core of the same cluster, equivalent to `SM4`. Also, the multi-cluster scenario investigated is `SMC1nC1` with 2 different tasks each in different core of two different clusters. Figure 5 depicts 3 scenarios with task mapping and resource usage for cores, clusters and DDR. In it: isolation which is `SM1` or `SMnC1` depending on the cache being enabled or not; single-cluster for `SM2`, `SM3`, `SM4`, `SMnC2`, `SMnC3`, or `SMnC4`; multi-cluster for `SMC1` or `SMCnC1`.

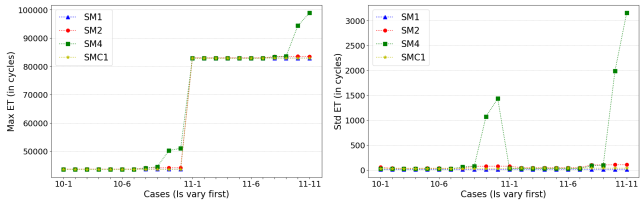
The task chosen are representative of a realistic real-time application with strong memory usage; the scenarios considered, and the platform are representative of realistic today's real-time system implementations with multi-core and with multiple execution conditions.

IV. Memory effect results

A. Memory interference with cache enabled

The effects of increasing memory requirements are decomposed considering a single-core scenario (isolation), multi-core scenarios (single-cluster), and multi-cluster scenarios.

Single-cluster scenarios. This part is about adding I tasks to the same cluster where one T task is running, and to analyze the evolution of T task's ETs. With it, we investigate interference from L1 and L2 cache, i.e. intra-cluster interference. In particular, we consider scenarios: 1) `SM1`, `SM2`, `SM3`, and `SM4`; `SM1` (isolation) is the baseline for scenario comparisons.
The results of average statistical analysis show that with low memory requirement there is no variation of the maxi-



(a) max (b) σ

Figure 6: ET max and σ vs. memory requirements in SM2 from Case9-1 on.

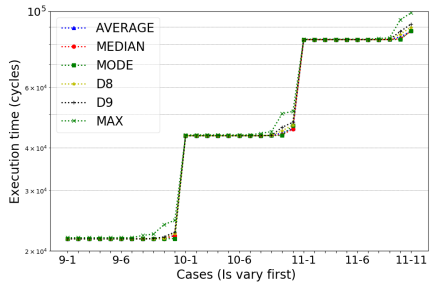


Figure 7: max, $D8$, $D9$, median, μ, and mode for SM4 and from Case9-1 on.

mum measured nor variability. The determinism is intrinsic to the behavior of the task, since there is no significant interference. Nonetheless, the statistical analysis with its metrics, guarantees such determinism with accurate models. By increasing memory requirements, some effects are captured by the metrics proposed. In Figure 6 scenarios are compared with respectively max and σ metrics applied. Some variability manifests for large memory requirements, while max increases are step-wise, as the memory requirements reach the size of cache levels. In here there is not intrinsic determinism, but the statistical analysis makes it with average and worst-case representations.

The cache never compromises determinism since: 1) for low memory requirements the system behavior is intrinsically deterministic, i.e. it has no variability or very low variability; 2) for large memory requirements in which variability appears, it is possible to derive accurate average and worst-case probabilistic models which guarantee determinism.

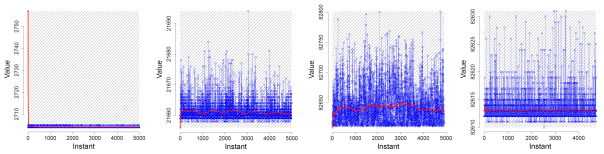
How the metrics for average analysis react to memory requirement changes is represented in Figure 7. A future analysis of that will contribute to define metrics most sensitive to changes, thus creating monitoring tools.

In Figure 8a is represented the trace for Case6-1 (all the cases with low memory requirement have similar behavior): for Case6-1 the EVT does not apply because there is not enough variability. But as soon as the variability increases (sufficient level of variability or EVT applicability evaluated with $\{h_{1,1}, h_{1,2}, h_{2,1}, h_{2,2}, h_3, h_4\}$), i.e. Figure 8b, Figure 8c and Figure 8d, the EVT applies with large confidence, and the pWCET estimates obtained are accurate in modeling tasks worst-case behaviors. The results of EVT applicability with DIAGXTRM, accurate pWCET models and good confidence are visible on Figure 9 and Figure 10. The pWCET representation is plotted with the icdf and logarithmic scale. As an example, with Case10-11 and Figure 10b, the stationarity and identical distribution is largely guaranteed ($cl_{1,1} = 4$), the independence is minimal but enough to be guaranteed ($cl_{2,2} = 1$ – only extremal independence exists), and the MDA is maximal ($cl_3 = 4$).

Multi-cluster scenarios. Here, the scenarios at study are: SMC1, SMC2, SMC3, and SMC4; SM1 (isolation) remains the baseline for comparisons. The separation between L2 caches, DDR memory and controller of the different clusters is now observed and analyzed. With multi-cluster scenarios, the additional interference applied through the use and bandwidth limitation of the Corenet, i.e. inter-cluster interference, is quantified.

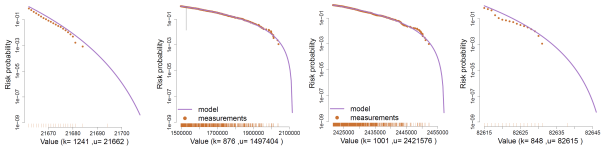
Figure 6 shows the comparison of SMC1 with other single-cluster scenarios; the results, both in terms of max and σ are comparable with SM2 and SM3.

Superposition effect. With SMC4, we also study the superposition of intra- and inter-cluster interference effects, see Figure 6a, and Figure 6b. SMC4 is built such that intra-cluster interference is similar to SM4 and inter-cluster interference is similar to SMC3. Intra- and inter-cluster interference are observable by the increase in ET they produce. We verify if increases from both interference sum up as if they were independent. With max, this is true for some cases e.g., Case9-9 on, but still a bit inferior to the independent summation. The total effect is on average 11.3% less the independent summation of effects. This is also the case for σ. We prove that intra- and inter-cluster interference are not independent; this is explained by the fact that intra-cluster interference requires access to DDR via Corenet.



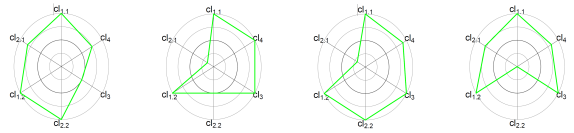
(a) Case6-1 under SM2 (b) Case9-10 under SM2 (c) Case11-11 under SM2 (d) Case11-10 under SMC1

Figure 8: Traces from different cases and different scenarios.



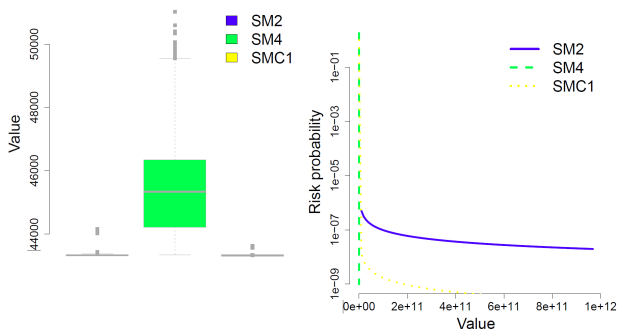
(a) Case9-10 (b) Case10-11 (c) Case11-5 (d) Case11-10

Figure 9: pWCET of different cases under SM2.



(a) Case9-10 (b) Case10-11 (c) Case11-5 (d) Case11-10

Figure 10: pWCET reliability different cases under SM2 scenario.



(a) Average model comparison (b) pWCET comparison

Figure 11: Average models and pWCET models comparison of Case11-10 under different scenarios.

Figure 11 compares average and worst-case models for Case10-11 under scenarios SM2, SM4, and SMC1. The results from SM2 and SMC1 are comparable, thus the interference is comparable; for SM4 there is more variability, larger maximum, and all the other metrics increase due to a larger interference from larger memory requirements. To note that the SM2 resulting distribution is a "heavy tail" distribution. This result from the input best fit procedure that the EVT applies; the MDA hypothesis supports this and the resulting distribution best represents the behavior of the system under the scenario. The average comparison is made with a box plot representation, while the pWCET representation is with the icdf and logarithmic scale.

Interference effects from cache memory have to be investigated, since they affect task behavior (on average and for worst-cases) even with hypervisors executing, see Figure 1 which is an example of the investigation made here, Figure 6, and Figure 9. In certain cases, the effects can be neglected, in other cases they need to be taken into account, and different task placements can reduce them.

B. Memory interference with cache disabled

The statistical analysis is also applied with cache memories disabled. In these scenarios, the CPUs directly access the DDR memory, and interference effect of the Corenet and of DDR controllers are expected to happen with less memory requirements than in case of cache memories. We study scenarios SMnc1, SMnc2, SMnc3, and SMnc4, with SM1 used as reference.

Single-core scenario: SMnC1. As expected, the single-core scenario without cache memory has worse metrics than with cache memories activated, see Figure 12 with the comparisons of max and σ of SMnC1 with other scenarios and different memory requirements. Also, the results are generally worse than scenarios SM3 and SM4, see Figure 6 for comparisons. Comparing SM1 and SMnC1 in terms of σ shows that for high memory requirements e.g., larger than Case7-1, the difference is of only 33 times in favor of SM1. On the contrary, with light memory requirements (smaller than Case6-1), the difference is quickly growing up to 366 more σ without cache. The reason is that with cache memories, theses cases were benefiting a lot from the use of L1 and L2. Such benefit has disappeared here.

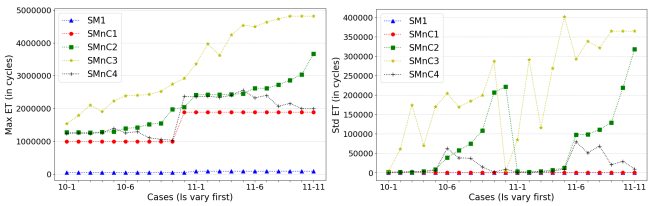
Multi-core scenarios: SMnC2, SMnC3. As expected, the more I tasks concurrently run with T, the worse the metrics are, as seen on Figure 12 and Figure 13. In particular, max is greatly impacted by memory needs, and that depends on both on T and Is needs. This was not the case with cache memory, where increases of 100% in Is requirements would only grow max of 19% for scenario SM4 and Case11-10, for example.

Also, adding I tasks while inhibiting cache memories largely increases the variability, primarily checked with σ. The exponential increase of σ with memory requirements, in particular with I tasks, can be interpreted as the effect of both intra-cluster and inter-cluster interference.

Multi-cluster scenarios: SMnC4. SMnc4 represents the multi-cluster case. Figure 12 shows the metrics for SMnC4 compared to SMnC2 and SMnc1. max, and σ are growing together with T memory requirements.

When T memory requirements increase, the decrease occurs for smaller I memory requirements. This means that for small T and I memory requirements, T and I are already interfering even on different clusters. Thus, they both try to access DDR and experience delay caused by cache inhibition. We interpret this delay as attempts to access to inhibited cache (equivalent to cache misses). Also, frequently accessing small memory region in DDR, i.e. small memory requirements, may be more expensive than rarely accessing big region. On the other hand, when the T memory requirements increase, then it switches to a default use of the DDR with less of those two possible sources of delay. There is also not enough Corenet use to produce notable inter-cluster interference. Finally, scenario SMnC4 is reaching scenario SMnC1 because there is no intra-cluster interference in both scenarios.

With multi-core platforms and cache disabled, the system is not necessarily deterministic: the variability is evident due to interference and concurrent access to shared memory. The system becomes deterministic with accurate pWCET models, which is possible with the analysis we propose.



(a) max (b) σ

Figure 12: ET max, μ, and σ vs. memory requirements in different scenarios from Case9-1 on.

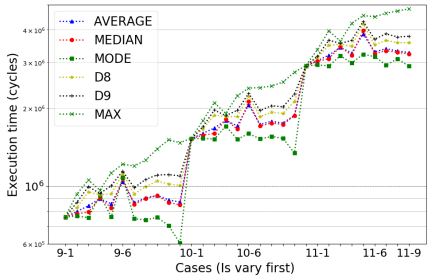
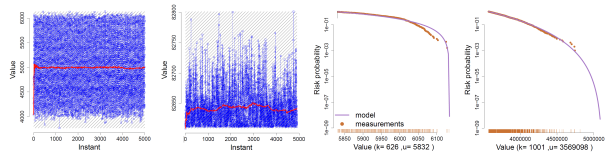


Figure 13: max, $D8$, $D9$, median, μ, and mode for `SMnC3`.

In Figure 14 two cases are represented under the same scenario `SMnC2`, in which the EVT has been applied and accurate pWCET estimate has been obtained. The pWCET representations are in the icdf form and logarithmic scale. Contrary-wise to the cases with cache memory, already with lower memory requirements e.g., Case2-2, the EVT can be applied: there is enough variability for that, and all the applicability hypotheses successfully pass their tests. Without cache, the system becomes intrinsically non-deterministic already with low memory requirements. Nonetheless, the statistical analysis we propose is able to infer average and worst-case models and to make the system deterministic out of those.



(a) Case2-2 (b) Case11-10 (c) Case2-2 (d) Case11-10

Figure 14: Trace and pWCET estimates of different cases under `SMnC2` scenario.

As verified by the statistical analysis, not using the cache does not increase system determinism. This is especially true with application heavily memory oriented and with multi-core platform. Also in this case the statistical analysis is required to have accurate average and worst-case models in order to make determinism.

C. Comparing cache and no cache performance

With the statistical analysis we propose, it is relatively easy to extract accurate average and worst-case models with and without cache. With it, it is possible quantify performance changes from the presence of cache to the absence of it. Also, we verify that in both cases, determinism can be achieved, and how.

The average models show that without cache, the variability becomes important since low memory requirements; with cache there are shaping and locality effects in place which reduce variability; low memory requirements enforce intrinsic determinism e.g., low variability, strong dependence. The stationarity is always ensured, not because of the application implemented, but because of the unpredictability in multi-core runtime executions.

With cache, few are the cases with variability, but for those the EVT can be applied and pWCETs can be computed: determinism can be achieved with them. Without cache, to all cases and scenarios it is possible to apply the EVT and to derive accurate pWCETs: determinism can be achieved with them.

In Figure 15 and Figure 16 are compared cases with and without cache by increasing memory requirements. With both max and σ statistics, there are evident linear correlations between both max and σ and the memory requirements. In particular, with and without cache there are two linear correlation behaviors: for CasesX-11 and for the others. Future work will be devoted to accurately quantify the correlation with case-by-case investigations, correlation tests, and adding other parameters to be measured.

Also, we detail a limit case, i.e. Case7-11 under scenario `SM2` and `SMnC2`. Under `SM2` there is not enough variability (intrinsic determinism), thus the EVT cannot apply. In particular, it cannot apply because of a strong dependence between ET measurements ($h_{2,1}$ nor $h_{2,2}$ satisfied). The statistical analysis quantifies and guarantees that. Under `SMnC2` instead, there is enough variability and independence such that the EVT can be applied, and an accurate pWCET model can be obtained.

Much more experiments and results are available in a technical report and at `https://forge.onera.fr/projects/diagxtrm2`.

V. RELATED WORK

The way storage resources e.g., cache, and bandwidth resources e.g., buses, can improve or degrade multi-core performance is studied in [14]. In [3] it is studied maximum interference in buses, depending on the bus arbiter which goes along with our notion of intra- and inter-cluster interference. [15] targets the decrease of performance caused by co-running tasks, and how to compute safe bounds on this decrease. This goes in line with our analysis of interference effect from different scenarios, and the need to account for parameters such as placements of `I` tasks.

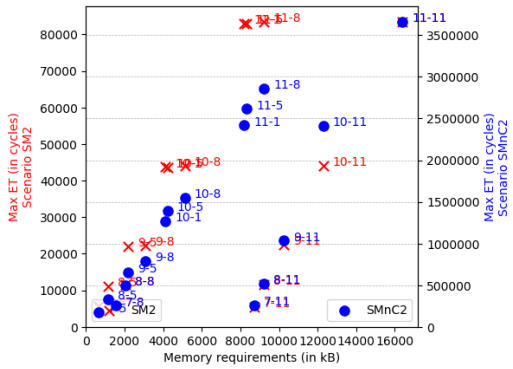


Figure 15: ET max vs. memory requirements for `SM2` and `SMnC2`.

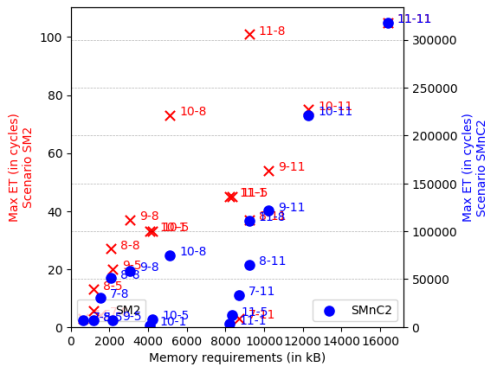
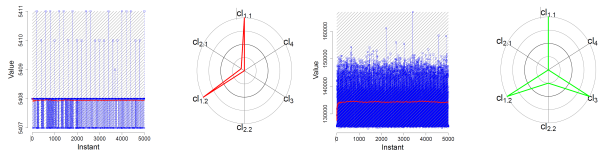


Figure 16: ET σ vs. memory requirements for `SM2` and `SMnC2` from Case9-1 on.



(a) `SM2` (b) `SM2` (c) `SMnC2` (d) `SMnC2`

Figure 17: Traces and spyder plots for Case7-11 in different conditions.

[16] is a work about contention-mitigating techniques for shared resources only using scheduling. In our case, scheduling is not part of the study but it will be addressed in future works by executing more than 1 task per core and including multi-core scheduling in order to identify and investigate other sources of interference. Recent research trends are extending timing analysis with incremental hypotheses [17], and interference analysis for certification purposes [18]. Our work differentiates from that since we propose first to quantify scenarios and interference effects, and if not possible to derive reliable models. Then countermeasures can be taken in order to simplify system behavior and reduce interference.

Previous researches agree that cache memories have much faster access time than main memories. Cache performance measurement and metrics such as in [19], [20], [21] indicate bandwidth improvement of about 2 times with L1 and L2 caches. In our case, we were able to prove that ET is about 23 times smaller for single-core scenario with L1 and L2 than without cache. Moreover, our analysis is able to extract efficient average and worst-case models in every configuration tested.

In [22] it is studied the notion of hypervisor on embedded avionic systems to improve determinism; shortcomings of virtualization are addressed in [23]. [24] presents a safety process to analyze multi-core interference, based on a partitioning process. Our work does not go along that direction yet. We decided to first develop a statistical analysis framework based on measurements, then to apply it for modeling interference effects. We plan to apply the same framework to study hypervisors and real-time operating systems and verifying their safety in partitioning.

Measurement-based approaches such as [25] or [26] focus on guaranteeing worst-case models with input coverage. In particular, [26] is about improving execution time bounds of multi-core platform, by taking into account contention requests of co-running tasks on a shared resource. Instead, our objective is to establish confident average models and confident pWCETs only for the measurements obtained, thus under the conditions considered and not others. Recent timing analysis approaches are mixing measurement-based and static timing analysis approaches to leverage qualities of both and to derive more accurate worst-case models for multi-core platforms [27]. Our approach can perfectly cope with the new hybrid timing analysis paradigm, and will benefit from it: basic block can be inferred and more accurate measurements of those can be carried out.

VI. CONCLUSION

This work defines the basics for the statistical analysis of task execution time measurements. Statistical metrics (for average and worst-case) are applied on trace of execution time measurements in order to characterize task execution behavior under different conditions. The probabilistic representations are used to characterize interference effects from shared memory. An exhaustive experimental evaluation is made together with some initial guidelines for predictable development of multi-core real-time embedded systems.

Future work will consist in developing a measurement framework to be minimal intrusive. It will be used in the future to measure other system parameters e.g., cache misses, and include them all in the statistical analysis. Moreover, we intend to apply the statistical analysis to other realistic real-time applications, to study other-than-memory interference sources, to characterize cache-coherence methods, or to study different bus access policies: interference effects and

policies to mitigate them will be investigated. Future work will also include the comparison between real-time operating system/hypervisors to investigate their isolation properties or the interference they bring to task executions.

References

[1] M. K. Gardner, "Probabilistic analysis and scheduling of critical soft real-time systems," Ph.D. dissertation, Champaign, IL, USA, 1999, aAI9953022.

[2] R. Pellizzoni and M. Caccamo, "Toward the predictable integration of real-time cots based systems," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, Dec 2007, pp. 73–82.

[3] D. Dasari, V. Nélis, and B. Akesson, "A framework for memory contention analysis in multi-core platforms," *Real-Time Systems*, vol. 52, no. 3, pp. 272–322, 2016.

[4] D. Dasari, B. Akesson, V. Nélis, M. A. Awan, and S. M. Petters, "Identifying the sources of unpredictability in cots-based multicore systems," in *SIES*. IEEE, 2013, pp. 39–48.

[5] R. I. Davis, A. Burns, and D. Griffin, "On the meaning of pwcet distributions and their use in schedulability analysis," in *In Proceedings Real-Time Scheduling Open Problems Seminar at ECRTS*, 2017.

[6] L. Santinelli, F. Guet, and J. Morio, "Revising measurement-based probabilistic timing analysis," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2017, Pittsburg, PA, USA, April 18-21, 2017*, 2017, pp. 199–208.

[7] S. J. Gil, I. Bate, G. Lima, L. Santinelli, A. Gogonel, and L. Cucu-Grosjean, "Open challenges for probabilistic measurement-based worst-case execution time," *Embedded Systems Letters*, vol. 9, no. 3, pp. 69–72, 2017.

[8] F. Cazorla, E. Quinones, T. Vardanega, L. Cucu-Grosjean, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim, "PROARTIS: Probabilistically analysable real-time systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2, pp. 1–26, 2013.

[9] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *the 24th Euromicro Conference on Real-Time Systems ECRTS*, 2012.

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

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

[12] R. I. Davis, L. Santinelli, S. Altmeyer, C. Maiza, and L. Cucu-Grosjean, "Analysis of probabilistic cache related pre-emption delays," *Proceedings of the 25th IEEE Euromicro Conference on Real-Time Systems (ECRTS'13)*, 2013.

[13] "Qoriq® t4240/t4160/t4080 multicore communications processors." [Online]. Available: https://www.nxp.com

[14] A. Abel, F. Benz, J. Doerfert, B. Dörr, S. Hahn, F. Haupenthal, M. Jacobs, A. H. Moin, J. Reineke, B. Schommer, and R. Wilhelm, "Impact of resource sharing on performance and performance prediction: A survey," in *CONCUR*, ser. Lecture Notes in Computer Science, vol. 8052. Springer, 2013, pp. 25–43.

[15] J. Reineke and R. Wilhelm, "Impact of resource sharing on performance and performance prediction," in *DATE*. European Design and Automation Association, 2014, pp. 1–2.

[16] S. Blagodurov, S. Zhuravlev, and A. Fedorova, "Contention-aware scheduling on multicore systems," *ACM Trans. Comput. Syst.*, vol. 28, no. 4, pp. 8:1–8:45, 2010.

[17] P. Raymond, C. Maiza, C. Parent-Vigouroux, E. Jahier, N. Halbwachs, F. Carrier, M. Asavoae, and R. Boutonnet, "Improving WCET evaluation using linear relation analysis," *LITES*, vol. 6, no. 1, pp. 02:1–02:28, 2019.

[18] I. Agirre, J. Abella, M. Azkarate-askasua, and F. J. Cazorla, "On the tailoring of CAST-32A certification guidance to real COTS multicore architectures," in *12th IEEE International Symposium on Industrial Embedded Systems, SIES 2017, Toulouse, France, June 14-16, 2017*, 2017, pp. 1–8.

[19] Visakhr, "Cache memory performance gain," Gate Overflow, 2016. [Online]. Available: https://gateoverflow.in

[20] P. Bureau, "High-speed-cache-memory," PCQuest, 2003. [Online]. Available: https://www.pcquest.com

[21] Hannu, "How much faster is memory ram compared to ssd for random access," SuperUser, 2017. [Online]. Available: https://superuser.com

[22] X. Jean, "Maîtrise de la couche hyperviseur sur les architectures multi-coeurs COTS dans un contexte avionique. (hypervisor control of COTS multi-cores processors in order to enforce determinism for future avionics equipment)," Ph.D. dissertation, Télécom ParisTech, France, 2015.

[23] R. Kaiser, "Complex embedded systems - A case for virtualization," in *WISES*. IEEE, 2009, pp. 135–140.

[24] G. Berthon, M. Fumey, X. Jean, H. Misson, L. Mutuel, and D. Regis, "White Paper on Issues Associated with Interference Applied to Multicore Processors," 2016. [Online]. Available: https://www.faa.gov/aircraft/air_cert/

[25] S. Law and I. Bate, "Achieving appropriate test coverage for reliable measurement-based timing analysis," in *ECRTS*. IEEE Computer Society, 2016, pp. 189–199.

[26] J. Jalle, M. Fernandez, J. Abella, J. Andersson, M. Patte, L. Fossati, M. Zulianello, and F. J. Cazorla, "Bounding Resource Contention Interference in the Next-Generation Microprocessor (NGMP)," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01259133

[27] T. Huybrechts, Y. De Bock, H. Li, and P. Hellinckx, "Hybrid approach on cache aware real-time scheduling for multi-core systems," in *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, F. Xhafa, L. Barolli, and F. Amato, Eds. Springer International Publishing, 2017, pp. 759–768.